WHEN AND HOW n CHOOSE k

Igor Pak

Abstract. We discuss different ways of generating k-subsets of an n-set, as well as of permutations, matrices and flags over the finite field $\mathbb{F}_q$. Connections with the Markov Chain approach and Bruhat decomposition are established. The black box groups approach is defined. No previous knowledge of the subject is assumed.

Introduction

The problem of choosing one or few objects among many has a long history and probably existed since the beginning of the human era (e. g. *"Choose twelve men from among the people,..."* Joshua 4:2). Historically this choice was mostly rational and random choice was considered to be a bad solution. Times have changed, however. Problems have become enormously complex and computers have intruded into virtually every part of human activity. In many cases random solution has become desirable, if not the only possibility. Which means that it's about time we understand the nature of a random choice.

The ultimate goal of this paper is to explain the meaning of the words *"pick at random a k-subset on an n-set"* and *"pick at random a k-dimensional subspace"*. Basically this paper is about details of the procedure of actually choosing this subset or subspace in real time. Somewhat unexpectedly these very details lead to some interesting and nontrivial questions.

For the past several years the author has discussed results in this paper with people working in different areas such as Combinatorics, Algebra, Probability Theory and Computer Science. It was quite striking to see that very few people actually understood the existing connections. Finally the author decided to write a self-contained review paper which would provide a basis for the mutual understanding. We restrict ourselves to study of the *classical cases* such as symmetric group S_n, k-subsets of $\{1, \dots, n\}$, full linear group $GL(n, \mathbb{F}_q)$ and k-dimensional subspaces of $\mathbb{F}_q^n$. In each of these cases we try to develop and compare various generating algorithms. While most of the results are well known, some are new and may be of interest.

Here is a general setup of a problem. We are interested in generating a uniformly chosen element from some finite set $\mathcal{S}$. If $\mathcal{S}$ has a global description, we can use it to get the desired random elements. One can think of it as reducing our randomization procedure to several coin flips. The problem is to optimize the number of steps in

1991 *Mathematics Subject Classification.* Primary 20-04, 05-04; Secondary 20B40, 20G40, 60G40, 60J15, 68R05.

Supported by NSF Postdoctoral Research Fellowship.

Typeset by $\mathcal{A}_{\mathcal{M}}\mathcal{S}$-TEX

the algorithm as well as the number of coin flips required (which are also called *random bits*). The first half of the paper describes how to do that in the classical cases.

Suppose now that the global structure of our set $\mathcal{S}$ is too complicated to be used efficiently in order to generate random elements. The usual way to proceed then, is to define a set of local transformation, each of them easy to describe, and then run a random walk by choosing a local transformation at random, independently at each step. For example, consider a group G and its set of generators S. We can start at the identity and run a random walk using multiplications by a random generator.

Under mild conditions the random walk will converge, often extremely fast, to a uniform distribution. Thus after walking long enough we get a nearly uniform random elements in $\mathcal{S}$. Now the problem is to find how long is "long enough", which is to find a mixing time for our random walk. Unfortunately even in nice cases this becomes a hard problem.

There is another way to look at this problem. Consider a stopping rule which observes the walk and stops it based on the previous observations. If the stopping state of this rule is uniform, this will give us a new algorithm for generating random elements in $\mathcal{S}$. On the other hand, if the stopping state is uniform even conditioned on the stopping time, the expected stopping time is roughly the mixing time (see §6.1 the precise formulation). Such a stopping time is called a *strong uniform time.* Now, given such a strong uniform time we solve both problems at once (of generating a random element and of finding the mixing time of a random walk). In section 6 we present several constructions of strong uniform times.

Finally, suppose we are given only the local structure of the set $\mathcal{S}$ and little or no global description. For example, suppose our group G is defined by its set of generators S lying in a bigger group $G_0 \supset G$. This situation is known as a *black box group.* Now we have to run some kind of random walk to achieve near-uniformity. The theoretically important algorithm due to Babai shows that this can be done in $O(log^5|G|)$ steps. Another important question to ask is whether a black box group is isomorphic to a given group. While this question cannot be answered efficiently in general, it has a fast solution in nice cases.

The paper is constructed in the following way. We do not highlight the results into theorems or lemmas but rather divide the body of the paper into sections, subsections and subsubsections, each with a certain purpose. The sections and subsections inside the sections are made to be increasing in difficulty. The remarks and some references are provided at the end of the sections.

The paper is written in a manner in which there is less emphasis on the results and more on explanation of how one should approach the problem. Some of the technical details and generalizations were left out or just briefly mentioned. The idea was to make this paper "easy reading" rather than overwhelm it with results.

The first section of the paper is introductory; it contains several basic definitions, notations and some elementary computations. In the second section we discuss several elementary generating algorithms. In particular, we present a "smart" way of choosing a k-subset in an n-set without keeping too much data in the memory. Although virtually all the results in this section are pure classics these days, this section cannot be skipped without loss of understanding.

In section 3 we show how to choose a permutation or a k-subset from a particular non-uniform distribution. The results of this section will be used later for random sampling of complete flags and of k-dimensional subspaces. In Section 4 we introduce the concept of *Schubert cells* and prove several basic results about their structure. Later, in section 5 we use Schubert cell decomposition to generate random flags and subspaces over a finite field.

In section 6 we study some natural random walks on permutations, k-subsets and k-dimensional subspaces. We show that the mixing time for these walk is of the same order (up to a logarithmic factor) as the complexity of the algorithm to generate the corresponding combinatorial objects. This confirms the general rule that natural random walks converge extremely fast.

Finally, in section 7 we briefly describe Babai's algorithm and give a heuristic for solving recognition problem for black box groups isomorphic to symmetric group and to special linear group.

Acknowledgments

I would like to express my gratitude to Persi Diaconis for introducing me to the subject and giving me a feel for randomness. I am indebted to David Aldous, Jim Fill, Michael Rabin, Ronitt Rubinfeld, Gian–Carlo Rota, Richard Stanley, Herb Wilf and Peter Winkler for the interesting and helpful conversations. Special thanks to Sergey Bratus, Gene Cooperman, Jason Fulman and Nathan Lulov for remarks on the manuscript.

This research started when the author was a graduate student at Harvard University and finished when the author was an NSF postdoctoral fellow at MIT.

Notation

$\mathbb{N} = \{1, 2, \dots\}$

$\mathbb{Z}_+ = \{0, 1, 2, \dots\}$

$[n] = \{1, 2, \dots, n\}$

$\begin{bmatrix} n \\ k \end{bmatrix}$ - k-subsets of an n-set $[n]$

$n! = 1 \cdot 2 \cdot \ldots \cdot n$, $n \in \mathbb{N}$

$\binom{n}{k} = \frac{n!}{k!\,(n-k)!} = \left| \begin{bmatrix} n \\ k \end{bmatrix} \right|$

$(n)_q = 1 + q + q^2 + \cdots + q^{n-1}$, $n \in \mathbb{N}$

$(n!)_q = (1)_q \cdot (2)_q \cdot \ldots \cdot (n)_q$, $n \in \mathbb{N}$

$\binom{n}{k}_q = \frac{(n!)_q}{(k!)_q\,(n-k)!_q}$

$\mathfrak{h}_n = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}$ - partial harmonic sum

$\mathbb{Z}$ - group of integers

$\mathbb{Z}_m \simeq \mathbb{Z}/m\mathbb{Z}$ - cyclic group with m elements

$\mathbb{F}_q$ - finite field with q elements

$GL(n, \mathbb{F}_q)$ - full linear group over the field $\mathbb{F}_q$

$B(n, \mathbb{F}_q)$ - Borel subgroup over the field $\mathbb{F}_q$

$U(n, \mathbb{F}_q)$ - unipotent group over the field $\mathbb{F}_q$

$Fl(n, \mathbb{F}_q)$ - set of complete flags over the field $\mathbb{F}_q$

$Gr(n, k; \mathbb{F}_q)$ - set of k-subspaces of $\mathbb{F}_q^n$

S_n - group of permutations of n elements

A_n - alternating group of permutations of n elements

$B_n = S_n \ltimes \mathbb{Z}_2^n$ - group of symmetries of an n-dimensional cube

$\daleth(p)$ - coin with probability p of heads, $0 \leq p \leq 1$.

1. How many are there?

1.1 Permutations

By the *symmetric group* S_n we mean a group of all permutations of an n-set $[n] = \{1, 2, \dots, n\}$. The number of all permutations i.e. the order of S_n is $n!$. Indeed, there are n ways to choose position of 1, there are $n-1$ ways to choose position of 2 among remaining places, ... , finally there is only one remaining place left for n.

We use the standard notation to denote permutations: $(3, 2, 1) \in S_3$ is a permutation $1 \to 3$, $2 \to 2$ and $3 \to 1$.

1.2 k-subsets of n-set

By the $\begin{bmatrix} n \\ k \end{bmatrix}$ we denote the set of all k-subsets of n-set $[n]$. The number of k-subsets of an $[n]$ is given by the binomial coefficient:

$$\left|\begin{bmatrix} n \\ k \end{bmatrix}\right| = \binom{n}{k} = \frac{n!}{k!(n-k)!}$$

Indeed, consider a surjective map $\varphi : S_n \to \begin{bmatrix} n \\ k \end{bmatrix}$ which maps a permutation $\sigma \in S_n$ onto the set of its first k elements. Observe that the preimage of a k-subset $A \in \begin{bmatrix} n \\ k \end{bmatrix}$ consists of all permutations $\sigma \in S_n$ with given sets of the first k and last $n-k$ elements regardless of their relative order. Therefore $|\varphi^{-1}(A)| = k!(n-k)!$ and

$$\left|\begin{bmatrix} n \\ k \end{bmatrix}\right| = \frac{|S_n|}{|\varphi^{-1}(A)|} = \binom{n}{k}$$

In other words, the elements in $\begin{bmatrix} n \\ k \end{bmatrix}$ are in a bijection with cosets in G/B, where $G = S_n$, $B = S_k \times S_{n-k}$.

1.3 Nonsingular matrices

By $GL(n, \mathbb{F}_q)$ we denote the group of nonsingular matrices over the finite field $\mathbb{F}_q$ with q elements. We claim that

$$|GL(n, \mathbb{F}_q)| = q^{\binom{n}{2}}(q^n - 1)(q^{n-1} - 1) \cdot \ldots \cdot (q - 1)$$

Indeed, the number of ways to choose the first row vector in a nonsingular matrix is $(q^n - 1)$: all but a zero row; the number of ways to choose the second row vector is $(q^n - q)$: all vectors that are not proportional to the first one, etc. This immediately yields the result.

1.4 Complete flags over $\mathbb{F}_q$

By $Fl(n, \mathbb{F}_q)$ we denote the set of all *complete flags* in $\mathbb{F}_q^n$, i.e. the sequences $\emptyset = V^0 \subset V^1 \subset \cdots \subset V^n = \mathbb{F}_q^n$ where $\dim(V^i) = i$, $i = 1, \ldots, n$. We claim that the number of these flags is given by the formula

$$|Fl(n, \mathbb{F}_q)| = (n!)_q = \prod_{i=1}^{n} (1 + q + \cdots + q^{i-1})$$

Before we prove it, note that $(n!)_1 = n!$, which means that we can think of the complete flags as a *q-analog* of permutations, even though there is no such thing as a finite field with 1 element.

To count the number of complete flags, define a surjective map $\psi : GL(n, \mathbb{F}_q) \to Fl(n, \mathbb{F}_q)$. Take a matrix M with row vectors $v_1, \ldots, v_n$ and define a vector space $V^i = \langle v_1, \ldots, v_i \rangle$, $i = 1, \ldots, n$. As before, let us compute the preimage of a fixed flag $\emptyset = V^0 \subset V^1 \subset \cdots \subset V^n = \mathbb{F}_q^n$. Consider the *Borel subgroup* $B(n, \mathbb{F}_q) \subset GL(n, \mathbb{F}_q)$ of upper triangular matrices $B = (b_{i,j})$, $1 \le i \le j \le n$, $b_{i,j} \in \mathbb{F}_q$, $b_{i,i} \ne 0$. Denote row vectors of $B^T M$ by $\widetilde{v}_1, \ldots, \widetilde{v}_n$ and the corresponding flag $\widetilde{V}^0 \subset \widetilde{V}^1 \subset \cdots \subset \widetilde{V}^n$. By definition,

$$\widetilde{v}_i = b_{1,i} v_1 + \cdots + b_{i-1,i} v_{i-1} + b_{i,i} v_i$$

Observe that since $b_{i,i} \ne 0$, the linear span of V^{i-1} and $\widetilde{v}^i$ is exactly V^i. Therefore by induction we have $\widetilde{V}^i = V^i$ for all $i = 1, \ldots, n$. This means that the whole orbit of B^T lies in the preimage of ψ. Similarly one can show that there is nothing but orbit of B^T in the preimage and

$$|Fl(n, \mathbb{F}_q)| = \frac{|GL(n, \mathbb{F}_q)|}{|B(n, \mathbb{F}_q)|} = \frac{q^{\binom{n}{2}} (q^n - 1)(q^{n-1} - 1) \cdot \ldots \cdot (q - 1)}{q^{\binom{n}{2}} (q - 1)^n} = (n!)_q$$

In other words, the complete flags in $Fl(n, \mathbb{F}_q)$ are in the bijection with cosets in G/B, where $G = GL(n, \mathbb{F}_q)$ and $B = B(n, \mathbb{F}_q)$ as above.

1.5 k-subspaces of $\mathbb{F}_q^n$

Define a *Grassmanian* $Gr(n, k, \mathbb{F}_q)$ to be the set of all k-dimensional subspaces $V^k \in \mathbb{F}_q^n$. We will show that

$$|Gr(n, k, \mathbb{F}_q)| = \binom{n}{k}_q = \frac{(n!)_q}{(k!)_q (n-k)!_q}$$

To prove that, define a map $\varphi : Fl(n, \mathbb{F}_q) \to Gr(n, k, \mathbb{F}_q)$ by forgetting all subspaces of a flag but the k-dimensional one. We have :

$$|Gr(n, k, \mathbb{F}_q)| = \frac{|Fl(n, \mathbb{F}_q)|}{|Fl(k, \mathbb{F}_q)| \cdot |Fl(n-k, \mathbb{F}_q)|} = \binom{n}{k}_q$$

Again, from here k-subspaces in $Gr(n, k, \mathbb{F}_q)$ are in the bijection with cosets in G/B, where $G = GL(n, \mathbb{F}_q)$ and $B = (b_{i,j}) \subset GL(n, \mathbb{F}_q)$, such that $b_{i,j} = 0$ for all $1 \le j < k < i \le n$.

1.6 Remark

As the reader may have noticed, everything in this chapter is very well known and can be generalized in many directions (see e.g. [GR, GJ, H1, H2, NW, P, S1, St]). For example, one may look at other root systems, such as C_n. The analogs of S_n for other root systems are called *Weyl group*, which in the case of C_n is the group of symmetries of the n-dimensional cube. Also, in this case the analog of $GL(n)$ is the *symplectic group* $Sp(2n)$ (see [H1, St, W]).

Note also that if we look at incomplete flags with a fixed sequence of dimensions we get q-multinomial coefficients as their cardinalities (see [GR, GJ, S1]).

2. How do you generate them? (First part)

The mathematical formulation of the question is the following: how do you choose an element of the classical set with equal probability?

2.1 Permutations

2.1.1 Here is a simple algorithm. Take an element 1 and put it on a line. Take 2. We have two possible places to put it: either to the left or to the right of 1. Do it with equal probability. Keep going. At the end we have n places to put n. Pick one at random and place n there. Stop.

2.1.2 There are thousands of others, more or less analogous ways to generate a random permutation. Here is a bit unusual way to do that. We are going to be marking elements of the permutation according to some rule. The rule is designed in such a way that each time we will have a random permutation of the marked elements with the rest of the elements remaining in their initial places.

Start with the identity permutation $id = (1, \ldots, n) \in S_n$ and mark the last element. We will mark a new element at each step. Suppose at step m we have m marked elements. Pick an unmarked element (it doesn't have to be a random choice; say it is the smallest unmarked element). Mark it. Exchange it with a randomly chosen marked element (including itself, in which case the element stays where it is). Stop after $(n-1)$ steps.

Checking by induction that we indeed get a random permutation is easy. Indeed, after m-th step we get a permutation with the new marked element be the smallest, the second smallest, etc., all with equal probability $\frac{1}{m+1}$. Therefore by induction it's a random permutation of these $m+1$ elements. This finishes the proof.

2.1.3 Define $T(i)$, $1 < i \leq n$ to be the following operation on a permutation $\sigma \in S_n$:

- Flip a coin $\beth\left(\frac{i-1}{i}\right)$. If heads, exchange $\sigma(i)$ and $\sigma(i-1)$.

Here by $\beth(p)$ we denote a coin with probability p of heads. Now, denote by $R(i) = T(i) \circ T(i-1) \circ \ldots \circ T(2)$ (means we apply $T(i)$ first, then $T(i-1)$, etc.).

Consider the following procedure. Starting with identity permutation and apply $R(2) \circ R(3) \circ \ldots \circ R(n)$. We claim that this gives us an algorithm for generating a random permutation. Indeed, after $R(2)$ we get either id or $(1,2)$ with equal probability. Suppose after $R(2) \circ R(3) \circ \ldots \circ R(k)$ every permutation of $[k]$ occurs with equal probability. After the first operation $T(k+1)$ the element $(k+1)$ stays with probability $\frac{1}{k+1}$ and the operations $T(k), \ldots, T(2)$ do not move it. Analogously the element $(k+1)$ gets to the place k with probability $\frac{k}{k+1} \cdot \frac{1}{k} = \frac{1}{k+1}$, etc. Thus after $R(k+1)$ the element $(k+1)$ gets to every place with equal probability. Recall that we started with a random permutation of $[k]$. Now use induction. This proves the claim.

2.1.4 Remarks

The algorithm in §2.1.2 was used by the author in [P] to bound the rate of convergence of random walks generated by transpositions (see also section 6 of this paper). The main advantage of this algorithm is that it can be easily generalized in many directions. Some of them can be found below.

The algorithm in §2.1.3 was introduced by Diaconis in [D2]. It is based on the following identity:

$$\overrightarrow{\prod_{k=n-1\ldots 1}}\ \overrightarrow{\prod_{i=1\ldots k}} \left(\frac{1}{i+1}\, e + \frac{i}{i+1}\, (i, i+1) \right) = \frac{1}{n!} \sum_{\sigma \in S_n} \sigma$$

where both sides are considered to be elements in group algebra $\mathbb{C}[S_n]$. Similar identities were obtained earlier by Jucys and Nazarov (see [J, N]) and in a less explicit context for any root system by Demazure (see [De]).

2.2 k-subsets of a n-set

2.2.1 There are also great many ways you can approach this problem. The most trivial way is simply to reduce the problem to the previous one by taking the first k elements of the permutation. Use §1.2 to prove that this works. Unfortunately when k is relatively small compared to n, this algorithm is very excessive.

2.2.2 One can try to use another simple idea. Start with an empty set. Pick an element $a_1 \in [n]$ in random and add it to your set. Pick an element $a_2 \in [n] \setminus \{a_1\}$ etc. until we get a_k. Stop. The disadvantage of this method is that it requires too much memory. It also outputs an unsorted k-subset so if we want it them to be sorted, we need to use an additional sorting at the end. This can be avoided by the tricked we present in the next subsection.

2.2.3 The algorithm works the following way. We will go over all numbers from 1 to n and for each we will decide whether we want to take it in a k-set or not. The only information we need to keep is the number l of elements we still need to choose ($l = k$ at the beginning) and the number m of elements left ($m = n$ at the beginning). When we accept or reject some element, we simply output it or not, change numbers l and m appropriately and move on to the next element. We claim that each time the probability of accepting a number is exactly $P(l, m) = \frac{l}{m}$. Indeed, this probability can be computed as follows.

$$P(l,m) = \frac{\left| \left\{ A \in \begin{bmatrix} m \\ l \end{bmatrix} : 1 \in A \right\} \right|}{\left| \begin{bmatrix} m \\ l \end{bmatrix} \right|} = \frac{\binom{m-1}{l-1}}{\binom{m}{l}} = \frac{l}{m}$$

This gives us a probabilistic rule for accepting/rejecting the next element. Stop when $l = 0$, i.e. when no elements are yet to be accepted.

Proof is clear.

2.2.4 Finally, we would like to present a version of the algorithm in §2.1.2 suited for this particular case. Start with the subset $A = \{1, \ldots, k\}$. At each step we will be marking one element in A, while making some changes in A. Now, suppose after m steps we have a set A with k elements of which exactly m are marked. Choose any unmarked element $a \in A$ (say, the smallest one). Flip a coin $\beth\left(\frac{m+1}{n-k+m+1}\right)$. If heads, mark the element and leave it there. If tails, choose randomly an element $b \in [n] \setminus A$, replace a by b and mark the latter. Stop after k-th step.

In order to show that this algorithm works the way it is supposed to, use the following trick. We can think that at the beginning all the elements $k+1, \ldots, n$ are marked and written to the right of A in some order. Then what we do each time is pick an unmarked element, mark it and exchange it with an unmarked one. What happens is that there is no need to exchange the element with the other one that are in the set A. That's where the number $\frac{m+1}{n-k+m+1}$ comes from. Hence we reduced the algorithm to the one considered in §2.1.2.

2.3 Nonsingular matrices over the finite field $\mathbb{F}_q$

2.3.1 The simplest known way to generate a random nonsingular matrix is just by choosing a random matrix, check whether it's nonsingular and if it is singular, try again. We will show that on average we wouldn't need many tries. Still, this algorithm takes about $O(n^3)$ steps for checking whether the matrix is nonsingular using the standard *Gaussian elimination algorithm.*

The first problem with this algorithm is that it uses on average more random bits than are really necessary. The second problem with this algorithm is that it is unclear how to generalize it for other algebraic groups such as the symplectic or orthogonal group over a finite field etc. (They are very sparsely distributed among all matrices). Later on we will explain the "right" way to generate nonsingular matrices which can be easily generalized in many directions.

2.3.2 **Lemma** *A randomly chosen matrix over $\mathbb{F}_q$ is nonsingular with probability greater than $1/4$ if $q = 2$ and $1/2$ if $q \geq 3$.*

Proof. The desired probability is given by

$$
\begin{aligned}
p(n,q) = \frac{|GL(n,\mathbb{F}_q)|}{q^{n^2}} &= \frac{q^n-1}{q^n}\,\frac{q^n-q}{q^n}\,\cdots\,\frac{q^n-q^{n-1}}{q^n} \\
&= \left(1-\frac{1}{q}\right)\left(1-\frac{1}{q^2}\right)\cdots\left(1-\frac{1}{q^n}\right) \\
&> \prod_{i=1}^{\infty}\left(1-\frac{1}{q^i}\right) = 1-\frac{1}{q}-\frac{1}{q^2}+\frac{1}{q^5}+\frac{1}{q^7}-\ldots
\end{aligned}
$$

The last product can be expanded into a sum using Euler's pentagonal theorem

$$
\prod_{i=1}^{\infty}\left(1-\frac{1}{q^i}\right) = \sum_{m=0}^{\infty}\frac{(-1)^m}{q^{m(3m\pm1)/2}}
$$

Therefore the desired probability can be bounded as

$$p(n,q) > 1 - \frac{1}{q} - \frac{1}{q^2} + \frac{1}{q^5} - \left(\frac{1}{q^6} + \frac{1}{q^7} + \frac{1}{q^8} + \dots\right) =$$

$$= 1 - \frac{1}{q} - \frac{1}{q^2} + \frac{1}{q^5} - \frac{1}{q^6}\frac{1}{1-\frac{1}{q}} = 1 - \frac{1}{q} - \frac{1}{q^2} + \frac{1}{q^5}\left(1 - \frac{1}{q-1}\right)$$

Finally we get

$$p(n,q) > 1 - \frac{1}{q} - \frac{1}{q^2}$$

In particular,

$$p(n,2) > 1 - 1/2 - 1/4 = 1/4$$

Also for all $q \geq 3$ we have

$$p(n,q) \geq p(n,3) > 1 - 1/3 - 1/9 = 5/9 > 1/2$$

This proves the lemma. □

We decided to include the proof of the lemma for the sake of completeness. From here one can see that the algorithm in §2.3.1 does indeed work very fast.

2.4 Complete flags over $\mathbb{F}_q$

2.4.1 We have a small problem here. Namely, we need to decide how we are going to encode the flags. By analogy with the idea described in §1.4 we will simply encode them by the sequence of vectors, which forms a basis $(v_1, \dots, v_n)$, and such that the vector subspaces are given by $V^i = \langle v_1, \dots, v_i \rangle$, $i = 1, \dots, n$.

2.4.2 Now we are ready to reduce the problem of generating a random complete flag to the problem of generating a random nonsingular matrix. Indeed, using the argument of §1.4 we can simply take row vectors of our randomly produced matrix and we are done. The big advantage of this algorithm is that it is very easy to code (assuming that the Gaussian elimination algorithm is available in some kind of mathematical package). The only problem is that one can actually generate a random complete flag much faster, in about $O(n^2)$ steps. You can't beat this bound since $O(n^2)$ is the number of steps required just to write the vectors down. We will come back to this question later in §5.1, where we present generalizations of the algorithm in §2.1.1.

2.4.3 If we were working over $\mathbb{C}$ we also could require our basis to be orthonormal, which would make this basis uniquely defined for a given flag. The only difference is that at the end we will now have to apply orthonormalization algorithm. Generally speaking we can do the same over finite fields, but some of the technical details become more complicated and we decided to omit this case.

2.5 k-subspaces of $\mathbb{F}_q^n$

2.5.1 Again we have a similar problem. We can try to encode k-dimensional subspaces by the k linearly independent vectors and try to generate them as in §2.2.1 by taking the first k vectors or a random matrix. In fact, we can pick a random $k \times n$ matrix and check whether it has the maximal rank (which it obviously does at least as often as is predicted by Lemma 2.3.2). Now use Gaussian elimination algorithm, which takes $O(k^2 n)$ steps in this case.

2.5.2 To conclude, we would like to show how the algorithm in §2.2.4 can be modified to apply in this case.

Start with some k-dimensional linear subspace $V \in \mathbb{F}_q^n$. Mark it. We will be gradually unmarking parts of V, reducing the dimension of the marked subspace by one after each step. We use induction to get k-subspaces which are chosen at random conditioned on having contain the marked subspace. In other words, at the beginning our marked subspace is all of V so there is only one possible k-dimensional subspace containing the marked subspace. After one step we get a $k-1$-dimensional marked subspace and a random k-dimensional subspace containing it. Keep on decreasing the dimension of the marked subspace. At the end we don't have anything marked, so we have a randomly chosen k-dimensional subspace.

Suppose we are given a k-dimensional subspace V with the marked subspace $W \subset V$, $\dim(W) = m$. Choose *any* $(m-1)$-dimensional subspace $W' \subset W$. Unmark everything in $W \setminus W'$. Flip a coin $\beth(P)$, where the probability $P = P(m,k,n)$ of heads is given by

$$P(m,k,n) = \frac{(k-m+1)_q}{(n-m+1)_q} = \frac{1+q+\cdots+q^{k-m}}{1+q+\cdots+q^{n-m}}$$

If it's heads, stay. If it's tails, choose a random $V' \supset W'$, $\dim(V') = k$, $\dim(V' \cap V) = 1$. Go to the next step. Stop when $W = \{0\}$.

To make this algorithm rigorous we need to define how we are going to choose subspaces V' and W' (see §5.2.3), explain how we are going to keep track of the spaces, and prove the correctness of the algorithm (see below).

2.5.3 To show why the above algorithm works, compare it with the algorithm in §2.2.4. Note that the idea is completely analogous except for the technical difference of *unmarking* the "random part" instead of marking it.

We prove the correctness of the algorithm by induction. We claim that after step i we have a random uniformly chosen k-subspace $V \in \mathbb{F}_q^n$ conditioned V contains the marked subspace W, where $\dim(W) = n-i$. The claim is clear at the beginning when $V = W = \mathbb{F}_q^n$. Define the probabilities

$$P'(m,k,n) = \frac{|\{V \in Gr(n,k,\mathbb{F}_q) : V \supset W\}|}{|\{V \in Gr(n,k,\mathbb{F}_q) : V \supset W'\}|}$$

If we assume that $P'(m,k,n) = P(m,k,n)$, then the step of induction is clear. Therefore modulo proof of the identity above, we prove the claim. Thus after k steps we indeed get uniform k-subspaces.

2.5.4 Now in order to finish the proof we need to show that the probabilities

$$P'(m,k,n) = P(m,k,n)$$

for all $0 \le m \le k \le n$.

Define $FL(m,k,n,\mathbb{F}_q)$ to be the set of sequences $\emptyset = V^0 \subset V^m \subset V^k \subset V^n = \mathbb{F}_q^n$, where $\dim(V^i) = i$. It is easy to see that

$$|FL(m,k,n,\mathbb{F}_q)| = \binom{n}{k}_q \binom{k}{m}_q$$

since the number of ways to choose the sequence $V^m \subset V^k \subset V^n$ can be thought as the number of ways to choose $V^k \subset V^n$ times the number of ways to choose $V^m \subset V^k$.

We have a group $GL(n,\mathbb{F}_q)$ acting transitively on $Fl(m,k,n,\mathbb{F}_q)$, which implies that the number of ways to choose a k-dimensional subspace which contain a given m-space is

$$c(m,k,n;q) = \frac{\binom{n}{k}_q \binom{k}{m}_q}{\binom{n}{m}_q} = \binom{n-m}{k-m}_q$$

From here we have

$$P'(m,k,n) = \frac{c(m,k,n;q)}{c(m-1,k,n;q)} = \frac{\binom{n-m}{k-m}_q}{\binom{n-(m-1)}{k-(m-1)}_q} = \frac{1+q+\dots+q^{k-m}}{1+q+\dots+q^{n-m}}$$

which finishes the proof.

We will get back to this algorithm in §5.2.3 with an explanation on how we can actually choose the subspaces.

2.6 Remarks

More about generation of permutations and k-subsets can be found in [K]. Various algorithms for generation of other combinatorial objects can be also found in [NW]. See also [DSh2] for the general treatment of subgroup algorithms.

The result of Lemma 2.3.2 is known, although use of the Euler's identity is not standard. Ironically Euler invented this identity also for computational purposes. He needed to compute the number $p(n)$ of partitions of n for large n ([An, Eu]). Indeed, the identity can be presented as follows

$$p(n) = p(n-1) + p(n-2) - p(n-5) - p(n-7) + p(n-12) + p(n-15) - \dots$$

which gives us a fast recursive algorithm.

Algorithm 2.2.3 can be found in [MR]. Another randomized algorithm was recently found in [Ra]. Algorithm 2.2.4 and its q-version given in §2.5.2 are probably new but not terribly interesting. They will be important to us when we get to the random walks in section 6.

3. What if you like some better than others?

3.1 Permutations

3.1.1 Statistics on permutations

There are great many different statistics on permutations and each of them can be transformed into a question about sampling from a nonuniform distribution associated with this statistics. Here is the definition.

Statistic on a finite set X is a function $\gamma : X \to \mathbb{Z}_+$. Fix some positive real number q. Define $\mathbf{S}(X, \gamma, q) = \sum_{x \in X} q^{\gamma(x)}$. The probability distribution on X associated with γ is defined as

$$P_{\gamma,q}(x) = \frac{q^{\gamma(x)}}{\mathbf{S}(X, \gamma, q)}$$

We needed to normalize it by $\mathbf{S}(X, \gamma, q)$ so that the total sum adds up to 1.

Now the question is: Given q and statistic γ, how can we sample from the distribution $P_{\gamma,q}$?

3.1.2 Inversions

Inversion in a permutation $\sigma \in S_n$ is a pair (i, j), $1 \le i < j \le n$ such that $\sigma(i) > \sigma(j)$. Define *inv* to be a statistic on S_n by the number of inversions. In other words,

$$inv(\sigma) = |\{(i, j) : \sigma(i) > \sigma(j), 1 \le i < j \le n\}|$$

Let us prove that

$$\mathbf{S}(S_n, inv, q) = \prod_{i=1}^{n} (1 + q + \cdots + q^{i-1}) = (n!)_q$$

Indeed, use the algorithm described in §2.1.1 and let t be a formal variable. We are going to compute the polynomial

$$f_n(t) = \sum_{\sigma \in S_n} t^{inv(\sigma)}$$

and then evaluate it at $t = q$.

Recall the algorithm in §2.1.1 for generating a permutation. Let us run this algorithm and keep track of the number of inversions it generates. There is only one way to put 1 on a line so $f_1 = 1$. There are two ways to put 2 : one to the right of 1 without any inversions and one to the left of 1 with one inversion. Therefore $f_2(t) = f_1(t)(1 + t)$ etc. There are n ways to put n : one with 0, 1, ... , $n - 1$ additional inversions. By induction we have that

$$f_n(t) = \prod_{i=1}^{n} (1 + t + \cdots + t^{i-1})$$

and by definition $\mathbf{S}(S_n, inv, q) = f_n(q)$. This completes the proof.

We would like to mention the important distinction here between the *polynomial* $f_n(t)$ and its *value* $f_n(q)$. Note also that $f_n(q) = (n!)_q$ is the number of complete flags over $\mathbb{F}_q$, where q is a power of a prime.

3.1.3 Standard sampling

Observe that the above stated idea can be reduced to an algorithm which generates a permutation σ with probability $q^{inv(\sigma)}/f_n(q)$. Indeed, when placing i simply place it in the right-most position with probability $1/(i)_q$, in the second position from the right with probability $q/(i)_q$, etc., in the left-most position i-th probability $q^{i-1}/(i)_q$, where $(i)_q = (1 + q + \cdots + q^{i-1})$. By definition we get exactly the probability distribution associated with statistic *inv* on S_n (i.e. by the number of inversions). Use induction to finish the proof.

Note also that if were trying to generalize an algorithm in §2.1.2, we wouldn't be able to prove the multiplicative formula for $f_n(t)$ since sometimes when inserting a new number we get a multiple of a different form. For example, if we had a permutation $(1, 2, 3) \in S_3$ with elements $1, 3$ marked, then three permutation occur after we exchanged 2 are $(2, 1, 3)$, $(1, 2, 3)$ and $(1, 3, 2)$ with 1, 0 and 1 inversions respectively, which is different from the needed 0, 1 and 2 distribution.

3.1.4 Product sampling

By analogy with §2.1.3, define $T(i; q)$, $1 < i \leq n$ to be the following operation on a permutation $\sigma \in S_n$.

- Flip a coin $\beth\left(\frac{(i-1)_q}{(i)_q}\right)$. If heads, exchange $\sigma(i)$ and $\sigma(i-1)$.

Denote $R(i; q) = T(i; q) \circ T(i-1; q) \circ \ldots \circ T(2; q)$. The algorithm works as follows. We start with a permutation $(n, n-1, \ldots, 1)$ and apply $R(2; q) \circ \ldots \circ R(n; q)$. We claim that this algorithm outputs a permutation σ with probability $P(\sigma; q) = \frac{q^{inv(\sigma)}}{(n!)_q}$. The proof goes through verbatim of the proof in §2.1.3 with the substitution of q-numbers in appropriate places. Note also that if we start with identity permutation $id \in S_n$ we get a permutation σ with probability $P(\sigma; \frac{1}{q})$.

Now observe that result in this subsection is equivalent to the following identity in $\mathbb{C}[S_n]$ which generalizes an identity presented in §2.1.4:

$$\overrightarrow{\prod_{k=n-1\ldots1}} \overrightarrow{\prod_{i=1\ldots k}} \left(\frac{q^i}{(i+1)_q} e + \frac{(i)_q}{(i+1)_q} (i, i+1) \right) = \frac{1}{(n!)_q} \sum_{\sigma \in S_n} \sigma \cdot q^{\binom{n}{2} - inv(\sigma)}$$

3.1.5 Metric on S_n

Define a *Coxeter graph* to be a graph with vertices in S_n and two permutation σ and σ' connected by an edge if and only if $\sigma = \sigma' \cdot (i, i+1)$ for some $i = 1, \ldots, n-1$. Call a *distance* $d(\sigma, \omega)$ the length of the minimal path between two permutation σ and ω. We call it distance because it is obviously nonnegative, symmetric, and satisfies the triangle inequality. We claim that

$$d(\sigma, \omega) = inv(\sigma^{-1}\omega)$$

First, observe that by the definition of a distance we have

$$d(\sigma, \omega) = d(\tau\sigma, \tau\omega)$$

for any $\sigma, \omega, \tau \in S_n$. Indeed, if we had a path $(\sigma, \sigma', \sigma'', \ldots, \omega)$, we would also have a path $(\tau\sigma, \tau\sigma', \tau\sigma'', \ldots, \tau\omega)$, and vice versa. Therefore it is enough to prove that

$$d(\sigma, id) = inv(\sigma)$$

for all σ (to see that, take $\tau = \sigma^{-1}$). We prove it by induction on the length of σ. Suppose the shortest path from σ to id starts with σ' and we know that $d(\sigma', id) = inv(\sigma') = l$. Hence $d(\sigma, id) = l + 1$ and we are left to prove that $inv(\sigma) = l + 1$.

By definition, there exists some i, such that $\sigma = \sigma' \cdot (i, i+1)$. It is easy to see that when we exchange i and $i+1$ in σ' we can either increase or decrease the number of inversions by exactly one depending on whether $(i, i+1)$ is an inversion or not. By induction it can't be $l - 1$. Therefore $inv(\sigma) = l + 1$ and we proved the result.

3.1.6 Braid Arrangements and Permutohedra

Consider a configuration of hyperplanes $\mathcal{H}_n = \bigcup H_{i,j} \subset \mathbb{R}^n$, where $H_{i,j}$ is defined as $(z_i - z_j = 0)$, $1 \le i < j \le n$. This configuration is usually called *braid arrangement*. The complement $(\mathbb{R}^n \backslash \mathcal{H}_n)$ is a disjoint set of regions which is in a natural bijection with the set of permutations. Indeed, it is not hard to see that each region is a set of a form $x_{\sigma(1)} > x_{\sigma(2)} > \cdots > x_{\sigma(n)}$ which makes the bijection obvious.

Construct a graph with vertices to be these regions and two vertices connected by an edge whenever we can cross only one hyperplane in order to get from one region to another. A little less obvious is the result that this graph is exactly a Coxeter graph after taking inverses of the permutations. Indeed, one can see that the hyperplanes adjacent to the region $x_{\sigma(1)} > x_{\sigma(2)} > \cdots > x_{\sigma(n)}$ are of the form $x_{\sigma(i)} = x_{\sigma(i+1)}$, $i = 1, \ldots n-1$ and therefore after crossing this hyperplane we end up at the region $x_{\sigma(1)} > \cdots > x_{\sigma(i)} > x_{\sigma(i+1)} > \cdots > x_{\sigma(n)}$ i.e. the new region corresponds to a permutation $\sigma' = (i, i+1)\sigma$ which proves the result.

Define a *permutahedron* to be a polytope in $\mathbb{R}^n$ with vertices $\big(\sigma(1), \ldots, \sigma(n)\big)$. This polytope is a translation of the Minkowsky sum of the vectors $e_i - e_j$, $1 \le i < j \le n$ and its 1-skeleton is again a Coxeter graph. The reason is that this polytope is dual to the braid arrangement which follows right from the definition. Another description of this polytope is a convex hull of its set of vertices which form a general orbit of a symmetric group acting on $\mathbb{R}^n$ by permutating coordinates. This is where the name comes from.

3.2 k-subsets of a n-set

3.2.1 Inversions

Define *inversion* in a k-subset $A \in \begin{bmatrix} n \\ k \end{bmatrix}$ to be a pair (i, j) such that $i \in A$, $j \in [n] \setminus A$, and $j > i$. Denote $inv(A)$ to be the number of inversions in A. We can think of inv as statistic on k-subsets of n-set. Here is a somewhat expected result (by analogy with §3.1.2)

$$\mathbf{S}(S_n, inv, q) = \binom{n}{k}_q$$

To show this formula, let us define a polynomial

$$f_{n,k}(t) = \sum_{A \in \begin{bmatrix} n \\ k \end{bmatrix}} t^{inv(A)}$$

We need to prove that

$$f_{n,k} = \frac{f_n}{f_k f_{n-k}}$$

We prove it using bijection φ defined in §1.2. Think about k-subsets of $[n]$ as permutations with increasing sequences of the first k and the last $n-k$ elements. For example, $A = \{1,4,6\} \in \begin{bmatrix} 6 \\ 3 \end{bmatrix}$ corresponds to $\sigma = (1,4,6,2,3,5) \in S_6$. Clearly $\varphi(\sigma) = A$ and by definition $inv(\sigma) = inv(A)$. On the other hand, when permuting first k and the last $n-k$ elements, we can only add the number of inversions in each part. Therefore

$$\sum_{\omega \in \varphi^{-1}(A)} t^{inv(\omega)} = t^{inv(A)} f_k(t) f_{n-k}(t)$$

Summing over all $A \in \begin{bmatrix} n \\ k \end{bmatrix}$ and dividing both sides by $f_k(t) f_{n-k}(t)$ we get the result.

3.2.2 Sampling

Suppose now we would like to sample an element $A \in \begin{bmatrix} n \\ k \end{bmatrix}$ from a probability distribution $P(\cdot)$:

$$P(A) = \frac{q^{inv(A)}}{\binom{n}{k}_q}$$

for some $q \in \mathbb{R}$, $q > 0$. As before, we can sample a permutation in the same manner as it is done in §3.1.3, and then take the first k elements (i.e. use φ). Of course one can do much better than that which we will show in the next subsection.

3.2.3 An algorithm

As in §2.2.3 we go over all numbers from 1 to n and for each of them decide whether we want to take it in a k-set or not. The only information we need to keep is the number l of elements we still need to choose and the number m of elements left, ($l = k$ and $m = n$ at the beginning).

We take an element $(n-m+1)$ and decide whether we would like to take it in a k-set or not. To do that we simply flip a coin $\beth(P)$, where the probability $P = P(l,m)$ is given by the formula

$$P(l,m) = \frac{1 + q + \dots q^{l-1}}{1 + q + \dots + q^{m-1}}$$

If heads, take $(n-m+1)$ in a k-set, and if tails skip this number and move to the next available. Stop when $l = 0$.

We claim that this algorithm does indeed produce a random k-subset A with probability proportional to the $q^{inv(A)}$. The algorithm itself is completely analogous to the one in §2.2.3. We just need to show that the probabilities $P(l,m)$ are chosen in the right way.

Indeed, let us compute the probabilities $P(l,m)$:

$$P(l,m) = \frac{\left|\left\{A \in \begin{bmatrix} m \\ l \end{bmatrix} : 1 \in A\right\}\right|}{\left|\begin{bmatrix} m \\ l \end{bmatrix}\right|} = \frac{\binom{m-1}{l-1}_q}{\binom{m}{l}_q} = \frac{(l)_q}{(m)_q}$$

Now, when we take a new elements into the k-subset we don't force any new inversions in the future since all the elements that can possibly make an inversion with $(n-m+1)$ must be smaller. On the other hand, the probability

$$Q(l,m) = 1 - P(l,m) = \frac{q^l(1+q+\dots q^{m-l-1})}{1+q+\dots+q^{m-1}}$$

has a multiple q^l because if we don't take an element $(n-m+1)$, by doing so we guarantee exactly l future inversions with all the numbers that *will be* added to our k-subset. This finishes the proof.

3.2.4 Metric on $\begin{bmatrix} n \\ k \end{bmatrix}$

There are several ways to define a distance on k-subsets. We briefly mention three of them.

1) Define a graph Γ_1 with all k-subsets as vertices and edges (A,B), where $A, B \in \begin{bmatrix} n \\ k \end{bmatrix}$ such that $|A \cap B| = k-1$. Define $d_1(A,B)$ to be a distance between A and B in this graph. It is easy to see that

$$d_1(A,B) = |A \cap B|$$

This is an interesting notion of a distance and it is symmetric under the natural action of S_n. Unfortunately this distance has nothing to do with inversions. In §6.3 we shall study nearest neighbor random walk on a graph Γ_1.

Here is another place where Γ_1 occurs. Define a *hypersimplex* to be the convex hull of characteristic functions of all k-subsets. It is an $(n-1)$-dimensional polytope which is a simplex when $k=1$. Another way to define this polytope is to take an orbit of the point $(1,\dots,1,0,\dots,0)$ (k ones) of S_n acting on $\mathbb{R}^n$ by permuting coordinates. An easy theorem says that Γ_1 corresponds to 1-skeleton of this polytope (see [EKK, Z]).

2) Define a graph Γ_2 on $\begin{bmatrix} n \\ k \end{bmatrix}$ as follows. Two k-subsets A and B are connected by an edge if and only if $|A \cap B| = k-1$, and the unique elements in $B \setminus A$, $A \setminus B$ are consecutive integers.

By definition Γ_2 is a subgraph of Γ_1, but it is no longer regular and has fewer symmetries. We can still define a distance $d_2(A,B)$ as a distance in this graph. One can easily see now that

$$d_2(J,A) = inv(A)$$

where $J = \{1,\dots,k\}$.

There is one more interpretation of Γ_2. Consider a set of all sequences $(a) = (a_1, \dots, a_k)$ such that $n-k \geq a_1 \geq a_2 \geq \dots \geq a_k \geq 0$. We say that two sequences are *similar* if they differ by 1 at exactly one place. Define a graph on these sequences by connecting pairs of similar sequences by edges. This graph appears to be isomorphic to Γ_2. To prove that we define an intermediate set of combinatorial objects which are easier to deal with and construct two simple bijections between our sets and this new set.

Consider set of all lattice paths in a rectangular $k \times (n-k)$, which start from the southwest corner, go either right or up, and end at the northeast corner. We can think of the area above the path as a diagram (which is called *Young diagram*) with the number of squares in each row corresponding to elements in a sequence. This gives the first bijection.

On the other hand, we can move along the path starting from the southwestern corner and ending at the northeastern corner. We need exactly n moves. Now we can either take an element into a k-subset or not depending whether our move was vertical or horizontal. It gives another bijection and establishes the result.

Although d_2 has some nice properties, we are not going to use this notion of a distance.

3) Denote the $\mathcal{INV}(A)$ to be a set of inversions in a k-subset A (see §3.2.1). Define a new distance as follows:

$$d_3(A,B) = |\mathcal{INV}(B) \setminus \mathcal{INV}(A)|$$

Although it is not true that $d_3(A,B) = d_3(B,A)$, we still have the triangle inequality:

$$d_3(A,B) + d_3(B,C) \geq d_3(A,C)$$

On the other hand there is an obvious connection with inversions:

$$d_3(J,A) = inv(A)$$

where $J = \{1, \dots, k\}$. Moreover, if $\mathcal{INV}(A) \subset \mathcal{INV}(B)$, then

$$d_3(A,B) = d_2(A,B)$$

Therefore it makes sense to think of d_3 as some kind of "oriented distance".

As far as we know d_3 does not correspond to any distance in an oriented graph. It occurs in combinatorics of k-dimensional subspaces. We shall come back to it later in the paper.

3.3 Remarks

The number of inversions in a permutation can be generalized for any Weyl group where it becomes the *length* of an element (see [Bou, H2, W]).

The q-identities similar to the one given in §3.1.4 for all root systems were found in [Ch].

Inversions in permutations and k-subsets have been heavily studied in Combinatorics (see e. g. [GJ, S1, SW]). They will reappear in connection with flags and k-dimensional subspaces. There are many other important combinatorial statistics.

Braid arrangements are very important by themselves. Considered over $\mathbb{C}$ their complement is an Eilenberg-MacLain space $K(\pi, 1)$, where π is a colored braid group (see [Ar, Ca]).

Permutohedron as an abstract polytope is simply a Coxeter complex of S_n and Coxeter generators (see [CM]). Translations of permutohedron can tile the whole $(n-1)$-dimensional space. This is obvious once you consider Coxeter complex of affine Weyl group $\hat{A}_n$.

Both permutahedron and hypersimplex are examples of the *orbit polytopes*, which can be obtained by taking a convex hull of an S_n-orbit of a particular point in $\mathbb{R}^n$.

The distance d_1 is well known and has been studied in numerous papers (see e.g. [Be, DSh3]). Distance d_2 in a different form was also considered in various places (see e.g. [GJ, S1, SW]). Distance d_3 is probably new.

4. What do these flags look like, anyway?

The idea is to break a set of flags into many pieces (cells) with each of them looking like finite vector spaces. Then we can try to work with each of the cells individually.

4.1 Complete flags

4.1.1 Schubert cells

Let us fix a basis $E = (e_1, \dots, e_n)$ in a vector space $V = \mathbb{F}_q^n$ and fix a complete flag $\emptyset = V^0 \subset V^1 \subset \cdots \subset V^n = V$ associated with E, where $V^i = \langle e_1, \dots, e_i \rangle$, $i = 1, \dots, n$ (cf. §1.4). Now with each flag $\emptyset = W^0 \subset W^1 \subset \cdots \subset W^n = V$ associate set of numbers

$$\alpha_{i,j} = \dim(V^i \cap W^j)$$

Define *Schubert cells* to be unions of flags with the same set of numbers $\alpha_{i,j}$, $1 \leq i, j < n$. Of course not all sets of numbers can appear and we will give a complete description of them in the next few subsections.

4.1.2 Unipotent group and its action on flags

Define the *unipotent group* $U(n, \mathbb{F}_q) \subset B(n, \mathbb{F}_q)$ to be the group of upper triangular matrices with 1's on the diagonal. Recall that complete flags correspond to cosets $B^T(n, \mathbb{F}_q) \setminus GL(n, \mathbb{F}_q)$ where $B(n, \mathbb{F}_q)$ is a Borel subgroup of all upper triangular matrices. We claim that Schubert cells correspond to double cosets $B^T(n, \mathbb{F}_q) \setminus GL(n, \mathbb{F}_q) / U(n, \mathbb{F}_q)$. In other words Schubert cells are the orbits of the unipotent group acting on flags through its action on matrices.

First, notice that by associativity of matrix multiplication it doesn't matter whether we first multiply a matrix from the right and then from the left or vice versa. Hence the action of $U(n, \mathbb{F}_q)$ on complete flags is indeed well-defined. Now, take a flag $W^0 \subset W^1 \subset \cdots \subset W^n = V$ associated with the basis $(w_1, \dots, w_n)$, where w_i is an i-th row vector of a matrix M. Take an upper triangular matrix $U \in U(n, \mathbb{F}_q)$. We can think of the complete flag corresponding to $M \cdot U$ as *the same* flag written in different coordinates. Observe that since U is upper triangular, the original basis flag $V^0 \subset V^1 \subset \cdots \subset V^n$ wouldn't change either (see §1.4) and therefore all the dimensions of intersections will remain the same. It proves that the orbits of the unipotent group do indeed lie inside the Schubert cells. The other direction, which claims that the numbers $r_{i,j}$ actually distinguish between the orbits, will be proven later using a combinatorial description of the orbits.

4.1.3 Permutation flags

Since we haven't yet fully established the connection between Schubert cells and orbits of the unipotent group, in this subsection we will continue to analyze the latter.

With each permutation σ associate a *permutation flag* V_σ corresponding to a basis $(e_{\sigma(1)}, \dots, e_{\sigma(n)})$. We claim that each orbit of the unipotent group $U(n, \mathbb{F}_q)$ acting on the complete flags $Fl(n, \mathbb{F}_q)$ contains exactly one permutation flag.

First of all, no two flags can lie in the same orbit since otherwise they would have to be in the same Schubert cell (by the result in §4.1.2) and have the same set of numbers $\alpha_{i,j}$, which is clearly false. Therefore if we prove that each orbit contains at least one permutation flag, we would instantly prove not only this claim but also the result about Schubert cells being the orbits of the unipotent group.

Fix a permutation $\sigma \in S_n$ and consider the orbit $O(\sigma, \mathbb{F}_q)$ of the unipotent group containing the permutation flag V_σ, $V_\sigma \in O(\sigma, \mathbb{F}_q)$. Recall that in a basis E we can think of $O(\sigma, \mathbb{F}_q)$ as an image of the map ψ (see §1.4) of an orbit of the $U(n, \mathbb{F}_q)$ acting on $GL(n, \mathbb{F}_q)$ by the right multiplication. Now we are ready to give an explicit combinatorial description of the orbit $O(\sigma, \mathbb{F}_q)$ (and, as we will see in a moment, of Schubert cells).

4.1.4 Combinatorial construction

By definition a flag V_σ corresponds to a matrix $D(\sigma) = (d_{i,j})$, where $d_{i,j} = 1$, $j = \sigma(i)$ and $d_{i,j} = 0$ otherwise. By definition an orbit of $U(n, \mathbb{F}_q)$ is the set of all matrices that can be obtained by adding columns to the columns with bigger numbers. Since all the ones are in different columns in $D(\sigma)$, it is easy to see that we get a set of matrices $\mathcal{R}(\sigma)$ such that $R = (r_{i,j}) \in \mathcal{R}(\sigma)$ if and only if $r_{i,\sigma(i)} = 1$ and $r_{i,j} = 0$, $j < \sigma(i)$. Indeed, both conditions are necessary since we can't possibly change $r_{i,\sigma(i)}$ and any numbers to the left of it (in standard matrix notations). Suppose now we have a matrix $R \in \mathcal{R}(\sigma)$. There is nothing to the right of $(\sigma^{-1}(n), n)$. There is exactly one matrix element to the right of $(\sigma^{-1}(n-1), n-1)$ and we can make it zero by subtracting the $(n-1)$-th column from n-th column appropriate number of times. Proceed analogously until we are left with matrix $D(\sigma)$. Therefore $\mathcal{R}(\sigma)$ is indeed an orbit of $U(n, \mathbb{F}_q)$ in $GL(n, \mathbb{F}_q)$.

Now let us find the equivalence classes of matrices $R \in \mathcal{R}(\sigma)$ under the action of $B^T(n, \mathbb{F}_q)$ by multiplication from the left. In other words, we would like to find a representative in a preimage $\psi^{-1}(F)$ for each of the $F \in O(\sigma, \mathbb{F}_q)$. As we know (see §1.4) we can add a row of the matrix $R \in \mathcal{R}(\sigma)$ to a row with a bigger number. By an argument similar to the one above we can get all the elements below $(i, \sigma(i))$ to be zeros.

Note that in this construction new zeros are placed in those positions (i, j) which correspond to the inversions in σ (see §3.1.2). Indeed, if we have a matrix element to the right of $(i, \sigma(i))$ and below $(\sigma^{-1}(j), j)$, then we have $i > \sigma^{-1}(j)$ and $\sigma(i) < j$ i.e. pair $(i, \sigma^{-1}(j))$ is an inversion in σ.

Define $\mathcal{S}(\sigma, \mathbb{F}_q)$ to be a set of matrices $M = (m_{i,j})$, $M \in \mathcal{R}$, such that $r_{i,\sigma(i)} = 1$, and $r_{i,j} = 0$ if $j < \sigma(i)$ or $\sigma^{-1}(j) < i$. We just showed that for each $F \in O(\sigma, \mathbb{F}_q)$ there exist exactly one matrix $M \in \mathcal{S}(\sigma, \mathbb{F}_q)$ such that $F = \psi(M)$.

4.1.5 Example

Let $n = 5$, $\sigma = (2,4,3,1,5) \in S_5$. All matrices $R \in \mathcal{R}(\sigma, \mathbb{F}_q)$ and $M \in \mathcal{S}(\sigma, \mathbb{F}_q)$ look as follows.

$$R = \begin{pmatrix} 0 & 1 & * & * & * \\ 0 & 0 & 0 & 1 & * \\ 0 & 0 & 1 & * & * \\ 1 & * & * & * & * \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}, \quad M = \begin{pmatrix} 0 & 1 & * & * & * \\ 0 & 0 & 0 & 1 & * \\ 0 & 0 & 1 & 0 & * \\ 1 & 0 & 0 & 0 & * \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

where by "$*$" we mean any number $x \in \mathbb{F}_q$. Four new zeros in M at places $(3,4)$, $(4,2)$, $(4,3)$ and $(4,4)$ correspond to four inversions in $\sigma = (2,4,3,1,5)$, namely to $(2,3)$, $(1,4)$, $(3,4)$ and $(2,4)$ respectively.

4.1.6 End of the proof

Recall that in §4.1.3 we showed that all we have left to prove is that each orbit of the unipotent group contains at least one permutation flag. We already know all permutation flags lie in different orbits. Now what we can do is compute the number of flags in orbits containing a permutation flag and then compare this number with the total number of complete flags.

By the §4.1.4 we have
$$|O(\sigma, \mathbb{F}_q)| = |\mathcal{S}(\sigma, \mathbb{F}_q)|$$
Recall that matrices $M \in \mathcal{M}(\sigma, \mathbb{F}_q)$ have exactly n ones, $0 + 1 + \cdots + (n-1) = \binom{n}{2}$ zeros to the left of ones and an additional $inv(\sigma)$ number of zeros which are below and to right of ones. Therefore the total number of "stars" in M (see example above) is $\left(n^2 - n - \binom{n}{2} - inv(\sigma)\right) = \binom{n}{2} - inv(\sigma)$ i.e.

$$|\mathcal{S}(\sigma, \mathbb{F}_q)| = q^{\binom{n}{2} - inv(\sigma)}$$

Summing over all permutations we get

$$\sum_{\sigma \in S_n} q^{\binom{n}{2} - inv(\sigma)} = q^{\binom{n}{2}} f_n\left(\frac{1}{q}\right) = f_n(q)$$

where $f_n(q) = (n!)_q = |Fl(n, \mathbb{F}_q)|$ is the total number of the complete flags of length n over $\mathbb{F}_q$ (see §1.4 and §3.2.1).

Let us summarize what we did. We proved that all the orbits of the unipotent group containing at least one permutation flag do not overlap and cover the whole set of complete flags. This implies that all orbits must contain a permutation flag which respectively implies that these orbits are indeed the Schubert cells. This completes the proof of the Schubert cell description.

4.2 k-subspaces of $\mathbb{F}_q^n$

The case of k-subspaces of $\mathbb{F}_q^n$ is almost exactly similar to the case of complete flags. So we will just state what Schubert cells look like in this case and skip much of the proof.

4.2.1 Schubert cells

As in §4.1.1, fix a basis $E = (e_1, \dots, e_n)$ in a space $V = \mathbb{F}_q^n$ and an associated with E complete flag $\emptyset = V^0 \subset V^1 \subset \cdots \subset V^n = V$ defined as $V^i = \langle e_1, \dots, e_i \rangle$, $i = 1, \dots, n$.

Now with each k-dimensional subspace $W \subset V$ we associate a set of numbers

$$\beta_i = \dim(V^i \cap W)$$

Define *Schubert cells* to be unions of subspaces with the same set of numbers β_i, $1 \le i < n$.

Rather than the whole unitary group in this case one should consider its subgroup $U(n, k, \mathbb{F}_q) \subset U(n, \mathbb{F}_q)$ such that $U = (u_{i,j}) \in U(n, k, \mathbb{F}_q)$ if and only if $u_{i,j} = 0$, $1 \le i < j \le k$ or $k < i < j \le n$.

The role of permutation flags would play *basis k-subspaces* i.e. subspaces generated by any subset of k basis vectors. Basis k-subspaces are in bijection with k-subsets $A \in \begin{bmatrix} n \\ k \end{bmatrix}$, $A = \{i_1, i_2, \dots, i_k\}$, $1 \le i_1 < i_2 < \cdots < i_k \le n$. They correspond to $k \times n$ matrices $D(A, \mathbb{F}_q)$ which have ones in places $(1, i_1)$, $(2, i_2)$, $\dots$, (k, i_k), and zeros elsewhere.

As in the case of complete flags, we claim that each Schubert cell contains exactly one basis k-subspace and can be obtained as an orbit of $U(n, k, \mathbb{F}_q)$ acting on $Gr(n, k, \mathbb{F}_q)$. These orbits $O(A, \mathbb{F}_q)$ can be lifted up to a set of matrices $\mathcal{S}(A, \mathbb{F}_q)$ which can be obtained from matrices $D(A, \mathbb{F}_q)$ by putting "stars" on places to the right of ones and with no 1 in the same column. One can observe that

$$|O(A, \mathbb{F}_q)| = |\mathcal{S}(A, \mathbb{F}_q)| = q^{k \cdot (n-k) - inv(A)}$$

which by analogy with §4.1.6 proves the claim.

4.2.2 Example

Let $n = 9$, $k = 4$, $A = \{1, 4, 6, 7\} \in \begin{bmatrix} 9 \\ 4 \end{bmatrix}$. All matrices $M \in \mathcal{S}(A, \mathbb{F}_q)$ look as follows:

$$M = \begin{pmatrix} 1 & * & * & 0 & * & 0 & 0 & * & * \\ 0 & 0 & 0 & 1 & * & 0 & 0 & * & * \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & * & * \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & * & * \end{pmatrix}$$

The number of inversions $inv(A) = 8$. Here we have

$$|\mathcal{S}(A, \mathbb{F}_q)| = q^{4 \cdot (9-4) - 8} = q^{12}$$

which is easy to see since M has exactly 12 "stars" in it.

4.3 Remarks

Although the results in this section are known, our elementary approach is not standard. The careful treatment of Bruhat decomposition for all root systems can be found in [Bou, H1]. Lie groups over the finite field are called *Chevalley groups* (see e.g. [Bor, St]). The group $U(n, k, \mathbb{F}_q)$ introduced in §4.2 is an example of a *parabolic subgroup*. These parabolic subgroups can be used in a similar way to study incomplete flag varieties.

Schubert cells by themselves represent a major development. They are heavily used in different branches of mathematics. There is even a *Schubert calculus*, which is a part of Algebraic Geometry (see e.g. [GH, M2]). Quite recently people in Algebraic Combinatorics have also invaded this area (see [LS, M2, S2]).

5. How do you generate them? (Second part).

The idea is to generate flags or subspaces in two steps. First, choose a Schubert cell with probability proportional to the number of elements it contains. Second, choose an element in the chosen cell.

As we already know these Schubert cells look like vector spaces over $\mathbb{F}_q$ of dimension equal to $\binom{n}{2}$ minus the number of inversions. So the first step is basically done in §3 while the second is easy since we have a complete combinatorial description of these cells. Here is a careful explanation of this idea.

5.1 Complete flags

Let q_0 be a power of a prime, and let $q = \frac{1}{q_0}$. We would like to sample complete flags over $\mathbb{F}_{q_0}$ from the uniform distribution. Following §2.4.1 the output must be of a form $(v_1, \dots, v_n)$.

5.1.1 The idea stated above can be reformulated the following way. By §4.1.4 and §4.1.6 choosing a complete flag is equivalent to first choosing the permutation σ and then a matrix $M \in \mathcal{S}(\sigma, \mathbb{F}_q)$. In the notation of §3.1.1 we therefore need to sample σ from the distribution $P_{inv,q}$, which can be done by the algorithm described in §3.1.3. Note that we need to use $q = \frac{1}{q_0}$ since by the result in §4.1.6 we have

$$|\mathcal{S}(\sigma, \mathbb{F}_q)| = q^{\binom{n}{2} - inv(\sigma)}$$

which forces us to take an inverse scale.

The second step is sampling a matrix $M \in \mathcal{S}(\sigma, \mathbb{F}_q)$. But this is easy since the set $\mathcal{S}(\sigma, \mathbb{F}_q)$ has an explicit description (see §4.1.4). Basically all we need to do is replace all the "stars" by the randomly chosen numbers in $\mathbb{F}_{q_0}$. Here is the final version of the algorithm.

5.1.2 Algorithm

```
1)Input $n$, $q_0$; $q := \frac{1}{q_0}$;
2)Sample $\sigma \in S_n$ from the distribution $P_{inv,q}$ using the Algorithm in §3.1.3;
3)Sample $M \in \mathcal{S}(\sigma, \mathbb{F}_q)$ uniformly;
4)Output $M$.
```

5.2 k-subspaces of $\mathbb{F}_{q_0}^n$

5.2.1

Again let q_0 be a power of a prime, $q = \frac{1}{q_0}$. We can work out the generating algorithm of the k-dimensional subspaces by analogy with flags. Following §2.4.1 the output must be of form k independent vectors $(v_1, \dots, v_k)$. Now we can use a very efficient algorithm described in 3.2.3 to generate a k-subset from the nonuniform distribution associated with statistic *inv* and a constant q. Here is the new algorithm.

5.2.2 Algorithm

1)Input n, k, q_0; $q := \frac{1}{q_0}$;
2)Sample $A \in \begin{bmatrix} n \\ k \end{bmatrix}$ from the distribution $P_{inv,q}$ using the Algorithm in §3.2.3;
3)Sample $M \in \mathcal{S}(A, \mathbb{F}_q)$ uniformly;
4)Output M.

5.2.3 Now we can also formalize the idea of an algorithm in §2.5.2. As usual we will encode the k-dimensional subspace V by k linearly independent vectors $v_1, \ldots, v_k$ In addition to that we will also keep in memory $n-k$ vectors $v_{k+1}, \ldots, v_n$ which form a basis together with $v_1, \ldots, v_k$. Since we are allowed to choose marked m-dimensional subspace W in any way we like, we will do it by making it to be a span of $v_1, \ldots, v_m$, $0 \le m \le k \le n$.

Now let us describe what we do at each step. At step $k - m + 1$ first we need to choose an $m-1$-dimensional subspace W' which is going to be a span of $v_1, \ldots, v_{m-1}$. Then we need to choose a new k-dimensional subspace $V' \supset W'$. We flip a coin $\beth(P)$, where $P = P(m,k,n) = \frac{(k-m+1)_q}{(n-m+1)_q}$. If it's heads, let $V' = V$ and keep all the vectors.

If it's tails, we replace v_m by a vector $v'_m \notin V$ in such a way that a subspace $V' = \langle v_1, \ldots, v'_m, \ldots, v_k \rangle$ is randomly chosen k-dimensional subspace in condition $\dim(V' \cap V) = 1$ and $V' \supset W'$.

Let a vector v'_m to be of a form $a_1 v_m + \cdots + a_{k-m+1} v_k + b_1 v_{k+1} + \cdots + b_{n-k} v_n$ Choose a number $1 \le i \le n-k$ with probability

$$p(i) = \frac{q^{i-1}}{1 + q + \cdots + q^{n-m}}$$

Set values $b_{i+1}, \ldots, b_n = 0$, $b_i = 1$ and choose random values $a_1 \ldots, a_{k-m+1}$ and $b_1, \ldots, b_{i-1} \in \mathbb{F}_q$. Now have the subroutine output vectors $v_1, \ldots, v'_m, \ldots, v_k$.

What we basically did here was show how to choose a line in a $(n-k)$-dimensional space by choosing first a *Schubert cell* and then its random element. This establishes a rigorous procedure for the algorithm in §2.5.2.

5.3 Nonsingular matrices over $\mathbb{F}_{q_0}^n$

5.3.1 We can use the algorithm described in §5.1.2 in order to generate random nonsingular matrices. Indeed, as we showed in §1.4, complete flags correspond to the cosets $B^T(n, \mathbb{F}_q) \backslash GL(n, \mathbb{F}_q)$, so all we need to do is choose a random complete flag and multiply it by a random lower triangular matrix. Here is an algorithm formalizing the idea.

5.3.2 Algorithm

1)Input n, q_0; $q := \frac{1}{q_0}$;
2)Sample $\sigma \in S_n$ from the distribution $P_{inv,q}$ using the Algorithm in §3.1.3;

```
3)Sample $M \in \mathcal{S}(\sigma, \mathbb{F}_q)$ uniformly;
4)Sample $B \in B(n, \mathbb{F}_q)$ uniformly;
5)Compute $M' = B^T \cdot M$;
6)Output $M'$.
```

5.4 "Lazy programmer" solution

Sometimes you can see "lazy programmers" who are capable of sacrificing the speed of an algorithm for the sake of writing less of a computer code. We can actually help these people.

5.4.1 Complete flags

Observe that instead of sampling a random matrix $M \in \mathcal{S}(\sigma, \mathbb{F}_q)$ we could have sampled a random unipotent matrix $U \in U(n, \mathbb{F}_q)$ which is easy to do since these are upper triangular matrices with ones on diagonal and then output the product $D(\sigma, \mathbb{F}_q) \cdot U$, where $D(\sigma, \mathbb{F}_q)$ is a permutation matrix corresponding to σ.

Sampling a random unipotent matrix might be considered easier to code especially if it's already done in some kind of math. package. By using it we still get a uniform distribution of corresponding complete flags which is exactly what we need. It may take more time to sample a unipotent matrix since it may have many more random entries to be chosen. On average however it is off only by a factor of 2 since the average number of inversions in a permutation is $\frac{\binom{n}{k}}{2}$, simply because $f_n(t) = t^{\binom{n}{k}} f_n(t^{-1})$.

The constant 2 is so small that the employer of the "lazy programmer" can probably live with it. Here is an algorithm which realizes this idea.

Algorithm

```
1)Input $n$, $q_0$; $q := \frac{1}{q_0}$;
2)Sample $\sigma \in S_n$ from the distribution $P_{inv,q}$ using the Algorithm in §3.1.3;
3)Sample $U \in U(\sigma, \mathbb{F}_q)$ uniformly; $M := D(\sigma, \mathbb{F}_q) \cdot U$;
4)Output $M$.
```

5.4.2 k-subspaces of $\mathbb{F}_{q_0}^n$

Exactly the same thing happens in here. One can come up with the following algorithm.

Algorithm

```
1)Input $n$, $k$, $q_0$; $q := \frac{1}{q_0}$;
2)Sample $A \in \begin{bmatrix} n \\ k \end{bmatrix}$ from the distribution $P_{inv,q}$ using the Algorithm in §3.2.3;
3)Sample $U \in U(n, k, \mathbb{F}_q)$ uniformly; $M := D(A, \mathbb{F}_q) \cdot U$;
4)Output $M$.
```

5.4.3 Nonsingular matrices over $\mathbb{F}_{q_0}^n$

Finally in the case of nonsingular matrices we get an amazingly simple algorithm by choosing two upper triangular matrices. Even High School students can learn this algorithm but only few of them will really understand why it works.

Algorithm

1)Input n, q_0; $q := \frac{1}{q_0}$;
2)Sample $\sigma \in S_n$ from the distribution $P_{inv,q}$ using the Algorithm in §3.1.3;
3)Sample $B_1, B_2 \in B(n, \mathbb{F}_q)$ uniformly;
4)Compute $M = B_1^T \cdot D(\sigma, \mathbb{F}_q) \cdot B_2$;
5)Output M.

5.5 Product sampling

Here we present an analog of the algorithm in §2.1.3 for generating nonsingular matrices over the finite field $\mathbb{F}_q$. The idea is again based on the identity in the group algebra $\mathbb{C}[GL(n, \mathbb{F}_q)]$ (see §2.1.4).

Recall that by $D(\sigma)$, $\sigma \in S_n$ we denote a matrix $D(\sigma) = (d_{i,j})$, where $d_{i,j} = 1$, $j = \sigma(i)$ and $d_{i,j} = 0$ otherwise. Denote by $E(i, j; a)$, where $1 \le i, j \le n$, a matrix with 1-s on diagonal, $a \in \mathbb{F}_q$ in place (i, j) and 0 elsewhere. Define

$$\mathcal{E}(i,j) = \sum_{a \in \mathbb{F}_q} E(i,j;a) \ , \qquad \text{where } i \neq j$$

$$\mathcal{H}(i) = \sum_{a \in \mathbb{F}_q, a \neq 0} E(i,i;a)$$

$$\mathcal{E}_- = \overrightarrow{\prod_{i=2 \dots n}} \ \overrightarrow{\prod_{j=1 \dots i-1}} E(i,j)$$

$$\mathcal{E}_+ = \overrightarrow{\prod_{i=1 \dots n-1}} \ \overrightarrow{\prod_{j=i+1 \dots n}} E(i,j)$$

$$\mathcal{H} = \prod_{i=1 \dots n} \mathcal{H}(i)$$

An easy argument shows that

$$\mathcal{E}_- = \sum_{M \in U^T(n, \mathbb{F}_q)} M$$

and

$$\mathcal{H} \circ \mathcal{E}_+ = \sum_{M \in B(n, \mathbb{F}_q)} M$$

Now consider a product

$$\mathcal{D} = \overrightarrow{\prod_{k=n-1 \dots 1}} \ \overrightarrow{\prod_{i=1 \dots k}} \left(\frac{q^i}{(i+1)_q} \, id + \frac{(i)_q}{(i+1)_q} \, D(i, i+1) \right)$$

Recall that by §3.1.4 we have

$$\mathcal{D} = \frac{1}{(n!)_q} \sum_{\sigma \in S_n} q^{\binom{n}{2} - inv(\sigma)} \cdot D(\sigma)$$

Thus the result in the previous subsection implies that

$$\mathcal{E}_- \circ \mathcal{D} \circ \mathcal{H} \circ \mathcal{E}_+ = \sum_{M \in GL(n,\mathbb{F}_q)} M$$

which leads to another sampling algorithm, too sluggish to be useful.

5.6 Gaussian elimination algorithm

We would like to conclude this section by establishing the connection between Gaussian elimination algorithm and the Bruhat decomposition

$$GL(n) = \coprod_{\omega \in S_n} B(n) w B(n)$$

Indeed, it's a common mistake to think that all the Gaussian algorithm does is invert matrices. The other common mistake is to think that it breaks a matrix into the product of an upper triangular and a lower triangular ones. What it really does, is present a matrix as a product of a lower triangular, permutation matrix and an upper triangular one, which is exactly what the Bruhat decomposition does in case of the $GL(n)$.

To see that, let us recall how Gaussian elimination algorithm works. If the first upper left corner is non-zero, we subtract the first row from each of the rows below it with some coefficients to place zeros in the first column everywhere but in the upper left corner. Then we subtract the second row from each of the consecutive rows, etc. This way we get an upper triangular matrix as a result of the subtractions and a lower triangular matrix indicating what were the coefficients. Note however that if the upper left corner is zero, the conventional wisdom says that it's not a problem since all we need to do is find a row with a nonzero entry in the first column. Let it be the first nonzero entry. The same thing is repeated for the second row etc. and at the end we also get a permutation matrix.

In other words what we just did was show how to break the set of all matrices into cells using the Gaussian algorithm. One can proceed with this idea and prove all the results contained in §4.1. We just think it easier to understand as described here.

5.7 Remarks

Observe that the number of steps needed for algorithm 5.3.2 can be bounded by the number of steps one needs to multiply two matrices. The number of random bits required by the algorithm is about n^2 i.e. the same as in the case of the algorithm in §3.1.2.

Algorithm 5.2.2 originates in [CW]. Unfortunately the authors omitted many of the details.

Algorithms 5.1.2 and 5.3.2 seem to be new. Algorithm 5.4.3 is brand new and should be patented by somebody. Although known, the connection between Gaussian elimination algorithm and the Bruhat decomposition became a part of mathematical folklore nowhere to be found in print.

The subsection §5.1.5 is of theoretical impotence and is meant to answer a question of Diaconis whether there are analogs of the identity in §2.1.4 for the full linear group.

6. How about taking a random walk?

This is another approach commonly used in practice for the purposes of random generation. The idea is to start from somewhere and walk on the objects you wish to generate at random. One may think that after some large number of steps the walk is at a random state. It is actually true indeed and can be made precise and proved in large generality. In this section we will study several interesting random walks and show that up to a logarithmic factor they are almost as fast as the direct generating algorithms (see §2, §5).

6.1 Definitions

6.1.1 Random walks

Define the *nearest neighbor random walk* as follows. Start at any vertex x and flip a fair coin. If it's heads, stay there. If it's tails, move to a randomly chosen vertex adjacent to x. This defines a reversible aperiodic *Markov chain*. Denote $P^k(x)$ to be the probability of a walk being at a vertex $x \in \Gamma$ after k steps. If Γ is regular, the walk has a uniform stationary distribution, i.e.

$$P^k(x) \to \frac{1}{N} \quad \text{as } k \to \infty \quad \text{for all } x \in \Gamma$$

where $N = |\Gamma|$ is the number of vertices in Γ. For the rest of the paper we will consider only regular graphs.

Let G be a finite group, S be its set of generators. We say that S is symmetric if $s \in S$ implies $s^{-1} \in S$. Define a *Cayley graph* $\Gamma(G, S)$ to be a graph with group elements $g \in G$ as vertices and pairs $(g, g \cdot s)$ as edges, $s \in S$.

6.1.2 Separation distance and mixing times

There are several ways to measure how fast the random walk mixes, most notably the *total variation distance*

$$tv(k) = \frac{1}{2} \sum_{x \in \Gamma} \left| P^k(x) - \frac{1}{N} \right|$$

and the *separation distance*

$$s(k) = N \cdot \max_{x \in \Gamma} \left(\frac{1}{N} - P^k(x) \right)$$

For more general definitions, properties of these distances and references see e.g. [AF, D1, P].

Although the total variation distance is used more often than the separation distance, the latter will be better suited for our purposes. Here are a few properties of the separation distance:

$$s(0) = 1$$

$$s(m + k) \le s(m) \cdot s(k)$$

$$s(k) \le tv(k)$$

for all $m, k > 0$.

It is useful sometimes to quantify the rate of convergence by a single number, which is called *mixing time*. Again, several definitions are known, namely

$$n_{1/2} = \min\{i : s(i) \leq \frac{1}{2}\} = \min\{i : P^i(x) \geq \frac{1}{2N} \text{ for all } x \in \Gamma\}$$

and

$$\xi = 1 + s(1) + s(2) + \cdots < \infty$$

The latter quantity is called *the total separation.* It is always finite and is of the same order of magnitude as $n_{1/2}$:

$$\xi \leq n_{1/2} \leq 2\xi$$

(see [P] for proofs and references). Note also that if the starting state of the walk is x_0, then $\xi \geq diam(\Gamma, x_0)$, where $d = diam(\Gamma, x_0)$ is the maximal distance in graph Γ between x_0 and other vertices. Indeed, we have $P^1(x_0) = \cdots = P^{d-1}(x_0) = 0$ so $s(1) = \cdots = s(d-1) = 1$ and therefore $\xi \geq d$.

For the rest of the paper by the mixing time we will mean the total separation. Finally, note that the submultiplicativity of the separation distance and the inequality above gives us

$$s(k) \leq 2^{-\lfloor k/2\xi \rfloor}$$

which is usually a sharp upper bound.

6.1.3 Strong uniform times

Since the probability distribution P^k can be very complicated, it is usually very hard to compute the separation distance directly. Here we briefly describe how to use the strong uniform time technique to get the sharp estimates.

Define a *stopping time* by a stopping rule which observes the walk and tells us when to stop based on the previous observations and additional randomness if necessary. We think of the stopping time of the walk as a random variable which will be denoted T.

The stopping time T is called *uniform* if the probability of stopping at each state is uniform. The stopping time T is called *strong uniform time* if the above probability is uniform and independent on the time the walk stopped:

$$P(x_t = z \mid T = t) = \frac{1}{N}$$

Heuristically, even if we know the time when the walk stopped, we still don't have any information about *where* it stopped.

Let us consider an example. Suppose we walk on C_4, a circle with four vertices. Start at state 1. Define the stopping time as follows: choose a random state and walk until we get there, then stop. It is easy to see that this is not a strong uniform time because if it stopped after one step, it couldn't possibly be the vertex 3, which is opposed to the starting one. On the other hand here is a right strong uniform time: walk till we hit either 2 or 4. Do one more step and then stop.

The theorem due to Aldous and Diaconis (see [AD2,D1]) gives a bound for the separation distance given the strong uniform time:

$$s(k) \le P(T > k)$$

From here for the total separation we have (see [P])

$$\xi \le E(T)$$

A strong uniform time for which the above inequality becomes an equality is called *perfect time*. It is known that for every random walk there exist a perfect time (see [AD2, D1]).

Given the random walk and a stopping time, define a *halting state* to be a vertex x_0 such that whenever the walks hits x_0, it always stops there. In general, there can be no halting states or there can be many of them. It is known (see [DF, P]) that if a strong uniform time has at least one halting state then it is perfect.

For example, recall the random walk on C_4 and a construction of the strong uniform time defined above. Observe that vertex 3 is a halting state, which means the walk never leaves it. Therefore our stopping time is perfect and the total separation is given by $\xi = E(T) = 2 + 1 = 3$ (it takes on average two steps to hit either 1 or 4).

6.1.4 Remarks

While very fast strong uniform times are known to exist in theory, they are usually very difficult to construct in practice. To generate a graph vertex uniformly it is sufficient to consider just uniform stopping times. Thus constructing strong uniform times is generally a harder problem. It turns out, that in the classical cases this problem can be solved in a way similar to generation algorithms.

The method of strong uniform times is originated in works by Aldous and Diaconis (see [AD1, AD2]) and was later developed in [DF, P, AP]. The notion of the separation distance is also due to Aldous and Diaconis, while the total separation was introduced by the author (see [AF, D1, P]). Halting elements were defined by Diaconis and Fill in [DF] (see also [AF, P]).

6.2 Permutations

6.2.1 Random walk

Let $R = \{(1,2), \dots, (1,n)\} \in S_n$ be the set of *star transpositions* in S_n. Denote by $\Gamma(S_n, R)$ the corresponding Cayley graph. Rather than taking the nearest neighbor random walk, take the probability of staying to be $1/n$. We can think of each walk steps as follows:

- Uniformly choose i, $1 \le i \le n$. Exchange elements in places 1 and i.

In other words, multiply a permutation by a random transposition $(1, i)$ where $1 \le i \le n$.

The main result of this subsection is the computation of the total separation ξ for this random walk:

$$\xi < 2\, n\, \ln n + \gamma\, n + \frac{1}{2}$$

where $\gamma \approx 0.5772156649$ is the Euler-Mascheroni constant (see e.g. [BE], §1.1; [WW], §12.1). We use an explicit construction of the strong uniform time to prove this claim.

6.2.2 Strong uniform time construction

Start with the identity permutation. The stopping rule consists of marking permutation elements according to walk movements. Once the element is marked, it stays marked forever. We stop when all the elements are marked. Here is how we mark elements:

- Mark the element n at the beginning.

Suppose the walk is at the state $\sigma \in S_n$ and is moving to $\sigma \cdot (1, i)$, $1 \leq i \leq n$.

- Mark $\sigma(1)$ if it is unmarked and $i = 1$ or $\sigma(i)$ is marked.

Stop when all n elements are marked.

6.2.3 Proof

Let us prove that the algorithm actually works which means that it indeed defines a strong uniform time. Note the similarity of this algorithm with the one in §2.1.2.

We will use the induction on the number k of marked elements. The inductive assumption is that all marked elements are in random order with respect to each other. In other words, even if we know the stopping time, which elements are marked and which k-subset of places they occupy, we still don't have any knowledge of their relative order. When $k = 1$ the condition is trivial.

Now suppose we have a k-subset A of marked elements. If the unmarked element $\sigma(1)$ is either exchanged with any of the marked ones or stays (as in §2.1.2) we can mark this new element. Indeed, the element $\sigma(1)$ after the walk step now has an equal chance to be the first, the second, dots, the last in the relative order of $A \cup \{\sigma(1)\}$.

Suppose now we exchange either two marked or two unmarked elements. In these cases we do not mark a new element. But then, the relative order of marked elements is still equally likely to be any given linear order on A, so we come back to the previous situation.

Recall that we stop when all the elements are marked. This means that the probability of stopping at any permutation conditioned the stopping time is uniform. This proves the result.

6.2.4 Mixing time

In contrast with the algorithm in §2.1.2 the random walk takes longer than just n steps, but only by a logarithmic factor. Indeed, let us compute the average stopping time under this stopping rule.

Observe that once we get an unmarked element in the first place, the only way to get a marked element there is by marking a new element. If we have k marked elements, $1 \leq k < n$, this will take on average $\frac{n}{k+1}$ steps.

On the other hand, after we mark a new marked element we need to wait till we get an unmarked element in the first place. If we have k marked elements, $2 \leq k < n$, this will take on average $\frac{n}{n-k}$ steps. Therefore we can break the stopping time into $2n-2$ periods described above. For the total separation we have

$$\xi \le E(T) = \sum_{k=1}^{n} \frac{n}{k+1} \; + \; \sum_{k=2}^{n-1} \frac{n}{n-k} < 2\,n\,\mathfrak{h}(n) = 2\,n\,\ln n + O(n)$$

where $\mathfrak{h}(n) = 1 + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{n}$. Now the estimate for ξ given in §6.2.1 follows form the asymptotic expansion for $\mathfrak{h}_n$ (see e.g. [BE, WW]):

$$\mathfrak{h}(n) = \ln n + \gamma + \frac{1}{2\,n} + \dots$$

An easy argument shows also that up to a constant this is the best one can get from this random walk. Indeed, it takes on average $n \log n$ steps just to *touch* all the elements. See [D1, P] on how to make this argument precise.

6.3 k-subsets of n-set

6.3.1 Random walk

Consider a graph Γ with all k-subsets of n-set as vertices. Connect two subsets A, A' by an edge if $|A \cap A'| = k-1$. We can think of the nearest neighbor random walk on Γ as follows. Suppose the walk is at state $A \in \begin{bmatrix} n \\ k \end{bmatrix}$. Randomly choose a pair (i,j), such that $i \in A$, $j \notin A$. Flip a fair coin. If it's heads, replace i by j. If it's tails, stay.

Here we will present a simple construction of a strong uniform time for this walk. It will give us the exact value of the total separation

$$\xi = E(T) = 2\,k\,\frac{n-k}{n+1}\,\big(\mathfrak{h}(k) + \mathfrak{h}(n) - \mathfrak{h}(n-k)\big)$$

where $\mathfrak{h}(m) = 1 + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{m}$.

6.3.2 Perfect time construction

Start with some k-subset A. Mark all the elements in $[n] \setminus A$. While walking on Γ we will be marking some of its elements in A according to the following rule. Suppose the walk is at a k-subset B with exactly m marked elements. Assume by induction that all elements in $[n] \setminus B$ are marked. This means we have a total of $m+n-k$ marked elements.

As in §6.3.1 we think of our walk in the following way. Randomly choose a pair (i,j), such that $i \in B$, $j \notin B$. Flip a fair coin.

- If heads, exchange i and j. If i is unmarked, mark it.
- If tails, stay and if i is unmarked, mark it with probability $P = \frac{m+1}{n-k}$.
- Go to the next step.

As before, we stop when all elements are marked.

We claim that this defines a perfect time.

6.3.3 Proof

First let us proof that we have a strong uniform time. The proof is very much similar to the one in §6.2.3 and is based on the same idea as §2.2.4. Some of the numbers are different however.

Our inductive assumption is that at each time the m-set of marked elements in our k-subset is drawn uniformly from the set of all $m+n-k$ marked elements. In other words, we want to prove that all the probabilities of having in our k-subset a particular m-subset of an $(m+n-k)$-set of marked elements are equal even conditioned that we know *which* elements are marked and time of the observation. Of course, when $m = k$ this condition is equivalent to the definition of a strong uniform time.

The base of induction is obvious: we have no marked elements in our k-subset. Consider what happens when the walk is at a k-subset B with m marked elements. In the next step we can either increase the number of marked elements by one or keep it the same. In the latter case we either stay at B or exchange two marked elements i.e. inductive assumption still holds.

The case when we increase the number of marked elements is more complicated. First of all it could happen either when the walk stays or when we exchange a marked and an unmarked element. We use the same trick as in §2.2.4 thinking that what we have is actually a permutation with elements in B placed on the left in their natural order, and elements in $[n] \setminus B$ are on the right. Following the logic of §2.4.4 we can think of our step as picking a random unmarked element and exchanging it with a randomly chosen marked element. Depending on whether this marked element is in our set B or not, we either stay or exchange these two elements (if both chosen elements are in B there is no point in exchanging them). The conditional probability of the marked element being not in B (given the first element is unmarked) is $Q_1 = \frac{n-k}{m+n-k+1}$.

The conditional probability of staying, whether because we want our new element to stay or because we need to exchange it with the marked element in B, is $P_1 = 1 - Q_1 \frac{m+1}{m+n-k+1}$.

Therefore in order to compute the conditional probability of staying and marking an unmarked element i we need to choose the probability of staying and marking to be

$$P = \frac{P_1}{Q_1} = \frac{m+1}{n-k}$$

Let us repeat what we just did. We showed that conditioned we mark a new element, the probability P of staying is chosen in such way that now we have in our k-subset exactly $m+1$ marked elements, which are chosen randomly from the set of all $m+n-k+1$ marked elements. This implies that our time is strong uniform.

We have yet to prove that our time is perfect. Suppose the walk is started at a k-subset A_0. Consider any k-subset $B \in [n] \setminus A_0$. Recall that $k \le n-k$ so $B \cap A_0 = \emptyset$. If the walk gets to B, it must have all the elements in $[n]$ marked. Thus the walks always stops at B and B is a halting state. Therefore our strong uniform time is perfect. This completes the proof.

6.3.4 Mixing time

Let us now compute the expected number of steps for this algorithm to stop. We assume that $k \le \frac{n}{2}$, otherwise we can simply switch the set with its complement.

Suppose we currently have m marked elements in our k-subset. Observe that the probability of marking a new element is equal to

$$\frac{k-m}{k}\left(\frac{1}{2}+P\right)=\frac{k-m}{k}\frac{m+n-k+1}{2(n-k)}$$

Therefore the expected stopping time of this algorithm is equal to

$$\begin{aligned}
E(T) &= (2\,k\,(n-k))\left(\frac{1}{1\cdot n}+\cdots+\frac{1}{k\,(n-k+1)}\right)\\
&= \frac{2\,k\,(n-k)}{n+1}\left(\frac{1}{1}+\frac{1}{2}+\cdots+\frac{1}{k}+\frac{1}{n-k+1}+\cdots+\frac{1}{n}\right)\\
&= 2\,k\,\frac{n-k}{n+1}(\mathfrak{h}(k)+\mathfrak{h}(n)-\mathfrak{h}(n-k)) \;=\; 2\,k\,\frac{n-k}{n+1}\left(\ln\frac{k\,n}{n-k}+O(1)\right)
\end{aligned}$$

Since our stopping time T is perfect we have

$$\xi = E(T) = 2\,k\,\frac{n-k}{n+1}\left(\mathfrak{h}(k)+\mathfrak{h}(n)-\mathfrak{h}(n-k)\right)$$

Note that when the ratio $\frac{k}{n}$ is fixed and k grows, we have $\xi = O(k\log(k))$.

6.4 k-subspaces of $\mathbb{F}_q^n$

6.4.1 Random walk

Consider a graph Γ with all k-dimensional subspaces of $\mathbb{F}_q^n$ as vertices. Vertices corresponding to subspaces V and V' are connected by an edge if and only if $\dim(V\cap V')=k-1$. We can think of the nearest neighbor random walk on Γ as follows. Choose a random $V'\in G(n,k,\mathbb{F}_q)$, $\dim(V\cap V')=k-1$. Flip a fair coin. If heads, stay. If tails, move to V'.

We assume that $2\,k\le n$ since otherwise one can use a duality to walk on $(n-k)$-dimensional subspaces. As before, we would like to analyze this walk by use of an explicit construction of a perfect time.

6.4.2 Perfect time construction

As in §2.5.2, at the beginning we mark our k-dimensional subspace. Following the walk we will be unmarking this subspace, leaving marked a subspace of a smaller dimension.

Suppose the walk is at the k-dimensional space V, and $W\subseteq V$ is its m-dimensional marked subspace. Choose a random $V'\in G(n,k,\mathbb{F}_q)$, $\dim(V\cap V')=k-1$. Denote $W'=W\cap V'$. It is easy to see that the dimension $d=\dim(W')$ can be either m or $m-1$. The rules are different depending on which one of them is the actual value of d. Flip a fair coin.

- If heads and $d=m$, stay.
- If tails and $d=m$, move to V'.
- If heads and $d=m-1$, stay. Unmark everything but W' with probability $P(m,k,n)$ given by

$$P(m,k,n)=\frac{1+q+\cdots+q^{k-m}}{1+q+\cdots+q^{n-m}}$$

- If tails and $d = m - 1$, move to V'. Unmark everything but W'.

Stop when $m = 0$ i.e. everything but 0 is unmarked.

6.4.3 Proof

Let us first prove that the constructed stopping time T is indeed strong uniform. We need to recall what happened in §2.5.2. There we made the assumption that at each time our k-subspace V is uniformly chosen given that it contains marked m-subspace W. Moreover, it is uniformly chosen even conditioned we know W and time of the event. The main difference is that in §2.5.2 we were able to decrease m at each step, so it took us only k steps to get $m = 0$. Here we break the set of all events into several distinct possibilities, some of them exactly like in §2.5.2 and the others are different.

Start with the case when we get tails and $d = m$. It is clear, that if $V \supset W$ is randomly chosen, then so is $V' \supset W$. This means that the assumption remains valid. Similarly to the cases when heads and we don't unmark, we simply stay and nothing changes.

At last, we have two cases when we do unmark and therefore decrease m by 1. We claim that these are exactly the cases for which everything worked in §2.5.2. Indeed, the only difference is that we restrict our attention to the cases when $m = d - 1$. Therefore the inductive assumption should still work, and T is strong uniform.

To prove that our stopping time is perfect we shall exhibit a halting state. Suppose we start at a k-dimensional subspace V. Recall that $2\,k \leq n$. Consider any k-dimensional subspace V' such that $V \cap V' = \emptyset$ By analogy with §6.3.3, whenever the walk gets into V' it must always stop there. Therefore T is perfect and this completes the proof.

6.4.4 Mixing time

Since T is perfect, we need to find the expected stopping time. First, compute the total number of k-dimensional subspaces V', such that $\dim(V \cap V') = k-1$. In other words we would like to compute the *degree* $deg(\Gamma)$, i.e. the number of edges going out of each vertex in Γ.

Denote $U = V \cap V'$. We can think of choosing V' as a two step procedure of choosing $U \subset V$ and then choosing $V' \supset U$, $V' \neq V$. The number of ways to choose U is $\binom{k}{k-1}_q = (1 + q + \cdots + q^{k-1})$ (see §1.5). To compute the number of ways to choose V' given U we can use the result in §2.5.4, which gives us $c(k-1, k, n; q) = \binom{n-k+1}{1}_q = (1+q+\cdots+q^{n-k})$. We also need to subtract 1 from the last expression to satisfy the condition $V' \neq V$. Finally, we get

$$|\{V' \in Gr(n, k, \mathbb{F}_q) : \dim(V \cap V') = k-1\}| = q\,(k)_q(n-k)_q$$
$$= q\,\left(1 + q + \cdots + q^{k-1}\right)\,\left(1 + q + \cdots + q^{n-k-1}\right)$$

Let us fix the m-dimensional marked subspace W. We need to compute the probability of unmarking at the next step. Before we can do that we need to compute the probability that the dimension $d = \dim(W \cap V')$ is equal to m.

Observe that $W \cap V' = W \cap U$ since $U = V \cap V'$. The number of ways to choose U, $W \subset U \subset V$ is equal to

$$c(m, k-1, k; q) = \binom{k-m}{k-m-1}_q = (1 + q + \cdots + q^{k-m-1})$$

Therefore the desired probability is given by the formula

$$P(d = m-1) = 1 - \frac{1+q+\cdots+q^{k-m-1}}{1+q+\cdots+q^{k-1}} = q^{k-m}\,\frac{1+q+\cdots+q^{m-1}}{1+q+\cdots+q^{k-1}}$$

Now, according to §6.4.2, the actual decrease in m happens with probability more than 1/2 given $d = m-1$. Indeed, it happens always when tails and with probability $P(m,k,n)$ when heads. To simplify the calculations we estimate it by 1/2 from the below. This will give us an upper bound. Analogously the lower bound would be less than the upper bound by a factor of 2. We get

$$E(T) < 2\sum_{m=1}^{k} \frac{1+q+\cdots+q^{k-1}}{q^{k-m}\,(1+q+\cdots+q^{m-1})}$$

which finally gives us

$$\xi = E(T) < 2\,\frac{1+q+\cdots+q^{k-1}}{q^k}\sum_{m=1}^{k}\frac{q^m}{(1+q+\cdots+q^{m-1})}$$

The sum on the right hand side can be (and sometimes is, see remarks below) considered as a q-analog of logarithm. When $q = 1$ we get the bound $2\,k\,\log(k)$ which is the upper bound found in §6.3.3. This is not a coincidence. In §6.5 we present a rigorous argument for that.

Now consider what happens when $q \geq 2$, which is very common among powers of primes. First, take $q = 2$. We get

$$\xi < 2\,\frac{2^k-1}{2^k}\sum_{m=1}^{k}\frac{2^m}{2^m-1} < 4\,k$$

On the other hand $\xi \geq k$ since $k = diam(\Gamma, V)$. Thus the total separation $\xi = O(k)$ where the constant varies between 1 and 4 and depends on q, n and k.

Note also that for fixed n and k the mixing time ξ is decreasing as q is increasing.

6.5 Back to k-subsets of n-set

6.5.1 We would like to conclude this section by explaining why the random walk on k-subspaces of an n-space behaves like a q-version of a random walk on k-subsets of an n-set. It turns out that there is a formal argument for that.

Suppose that our random walk on k-subspaces starts at V_0. Notice that by symmetry the only invariant of a k-subspace V is the dimension $V_0 \cap V$. Observe also that this dimension can either remain the same or change by at most 1 at each step. This means that we basically can make this walk into a birth and death chain by simply projecting the walk. This simplifies the problem significantly.

Here is a different way of doing the same kind of procedure. Recall that each k-subspace lies in a Schubert cell corresponding to exactly one of the k-subsets. Suppose our basis $E = (e_1, \ldots, e_n)$ was chosen in such a way that the starting k-subspace V_0 was a span of $(e_1, \ldots, e_k)$. Then by definition all the elements of the Schubert cell should have the same dimension $\beta_k = \dim(V_0 \cap V)$. It implies that at each time the probability of the walk being at V is constant on all V in the same

Schubert cell. Therefore we can project our walk on a set of k-subsets of an n-set. It would not be a random walk of course, but rather some new Markov chain, which can be described.

6.5.2 First observe that since the stationary distribution for the random walk on k-subspaces is uniform, the stationary distribution of the projected chain must be proportional to the size of the Schubert cell. Recall the definitions and results in §4.2. We have the stationary distribution:

$$\pi(A) = \frac{q^{k\,(n-k)-inv(A)}}{\binom{n}{k}_q}$$

Observe also that we can move from any k-subspace in the Schubert cell corresponding to A to at least one k-subspace in the Schubert cell corresponding to A' if and only if $|A \cap A'| = k-1$. One can actually compute the transition probabilities of this Markov chain:

$$p(A \to A') = \frac{q^{d_3(A,B)}}{q\,\left(1+q+\cdots+q^{k-1}\right)\,\left(1+q+\cdots+q^{n-k-1}\right)}$$

where $d_3(A,B)$ is defined as in §3.2.4, 3). From here one can get a combinatorial interpretation for d_3 in terms of the number of k-dimensional subspaces with certain properties.

This gives us a complete description of the projected Markov chain.

6.5.3 Now we can use the following trick. We can now extend the definition of the Markov chain described in §6.5.2 to *any* $q > 0$. Fix n and k and consider $\begin{bmatrix} n \\ k \end{bmatrix}$ as the set of states of the chain. Define transition probabilities using formulas in §6.5.2.

First we need to show that they indeed define transition probabilities for *any* $q > 0$, not just powers of primes. In other words we need to show that they indeed add up to 1 in each state. Multiply these probabilities by the denominator $g(n,k;q) = q\,\left(1+q+\cdots+q^{k-1}\right)\,\left(1+q+\cdots+q^{n-k-1}\right)$. We know that these corrected probabilities are polynomial in q and add up to $g(n,k;q)$ for the infinite number of values of q. Since the degree of polynomials is bounded by n, we immediately get that these polynomials are *equal* to $g(n,k;q)$ and therefore their ratio is 1 for all $q > 0$. Therefore the Markov chain is indeed correctly defined.

Using a similar argument we can prove that the stationary distribution is given by

$$\pi(A) = \frac{q^{k\,(n-k)-inv(A)}}{\binom{n}{k}_q}$$

for any $q > 0$. Moreover, given the right definition of separation distance in case of non-uniform stationary distribution, strong stationary time, etc. (see [AF, DF]), one can show that the bounds found in §6.4.3 will work in this case as well. The reason for that is that the stopping time like everything else in this section is "polynomial in q" which means it can be defined and proved for any q.

6.5.4 Now we can finally formalize the connection between random walk on k-subsets and on k-subspaces announced earlier.

Clearly when $q = 1$ all transition probabilities in the Markov chain defined in §6.5.2 become equal. Also the stationary distribution becomes uniform. In fact our Markov chain becomes a random walk on k-subsets of an n-set (see §6.3.1). But by the argument given in §6.5.3 this means that we can get "for free" both stopping time and bounds for this random walk (see §6.3.2, 3). By simply substituting $q = 1$ in all the formulas we get in §6.4.2, 3. Moreover, one can show that we get precisely *the same* stopping time and bounds. The details are left to the reader.

6.6 Other random walks and remarks

6.6.1 The theory of Markov chains and random walks in particular is very extensive. See [F] for an introduction into the field and [AF] for an up to the date study of the random walks on graphs. For the total variation distance there exist a coupling technique which is another probabilistic argument for working with general Markov chains See [AF, D1] for the definition references and applications.

6.6.2 The case of the random walk on S_n generated by *all* transpositions has been heavily studied. The mixing time $\xi = O(n \log n)$ in this case. See [D1, D3] for the references and history of the subject. In this case there are several known strong uniform time constructions (see [D1, Ma, P]). The walk by star transpositions was earlier studied in [FOW]. The construction in §6.2.2 first appeared in [P] in a more general context. See also [DSa, P] for other random walks on S_n.

Random walk on k-subsets was introduced and analyzed in [DSh3] (see also [Be, D1, Gr]). The strong uniform time construction given here is the only known and was given in [P]. Note that the upper bound found in [DSh3] is asymptotically tight only when $k = \Omega(n)$.

Random walk on k-dimensional subspaces was studied in [Ari]. Again we follow [P] in our treatment.

There are various other random walks on $GL(n, \mathbb{F}_q)$ which have a property of rapid mixing. Most notably, the random walk on trasvections which generate $SL(n, \mathbb{F}_q)$ was analyzed in [Hi]. By transvection we mean the linear transformation that fixes a hyperplane. It turns out that the random walk is rapidly mixes after $O(n)$ steps. Since generation of a transvection is fast and multiplication of a matrix by a transvection takes $O(n^2)$ multiplications, this gives us a random matrix generating algorithm in $O(n^3)$ steps. See [Gl] for walks on other linear groups generated by conjugacy classes.

6.6.3 Although much of the technical details were omitted, we believe that the section §5.5 has a philosophical importance. We showed how to construct a Markov chain which would simultaneously generalize both random walk on k-subsets and on k-dimensional spaces over the finite field. This explains the similarities one can observe throughout the paper. One can also make the same kind of argument for permutations and complete flags.

7. Black box groups

7.1 Definitions

Suppose the elements of a group are given as binary stings of a fixed length, say N, and a "black box", also called an *oracle* performs group operation: multiplication, taking an inverse, and recognition of the identity element. Such a group G

is called a *black box group.* Note that the encoding need not be bijective: different strings may correspond to the same group element while some strings may not correspond to group elements at all. Note also that $|G| \leq 2^N$.

Suppose we are also given a set S of generators of a black group G. The first problem is to find an algorithm in such a generality for generation of the uniform group elements. In general, this must take an exponential time (in N), since the problem is harder than computing the order $|G|$ and the latter was proved to be exponentially hard in [Ba1]. However, if we weaken the problem to finding *nearly-uniform* group elements the problem can be resolved in a polynomial time. This is the result of Babai we describe in §7.2. Thus in a sense the situation is similar to the approximate counting problem.

By nearly-uniform group elements we mean that all group elements are generated with probability at least $1/2|G|$. Of course, if we repeat the algorithm several times and take a product of the resulting group element, we can get as close to the uniform distribution as we need. Formally if we achieve near-uniformity after N steps, then after $k\,N$ steps we get the probability of each element at least $(1-1/2^k)/|G|$ (see §6.1.2 above).

The second problem is the *black group recognition problem.* Given a black box group G we would like to test whether it is isomorphic to a fixed finite group H. Here in §7.3, 4 we describe algorithms for recognition of S_n and $SL(n, \mathbb{F}_q)$ given as lack box groups.

Unfortunately, black group recognition problem in general is again too hard to have a polynomial solution. It is useful sometimes to assume the existence of an oracle which computes the order of group elements. Amazingly, one can tell a lot just by the order of the group elements. We call G a *gray box group* then.

We give a constructive solution for the recognition problem. This means that me not only prove that $G \simeq H$, but also construct such an isomorphism $\phi : G \to H$ explicitly. This approach is called *constructive recognition.*

About the algorithms. It is important to distinguish algorithms that are correct most of the time from algorithms that are correct all the time. The latter are called *Las Vegas* algorithms while various formulations of the former are called *Monte Carlo* algorithms. When appropriate we will specify the type of our algorithms.

Everywhere in this section μ is the time necessary to multiply two group elements. For simplicity, we will suppress most of the constants using notation $O(f(n))$ instead. The right constants can be found in the literature.

7.2 Generating random group elements

7.2.1 Reducing the number of generators

Let G be a finite black box group and let S be its set of generators. At this stage we construct another set of generators S_1 such that $|S_1| = O(\log |G|)$. Of course, any increasing subgroup chain of G must have length at most $l = \log_2 |G|$ so having more than l generators would be redundant. Here is how the this can be done in practice.

Let $S = \{s_1, \dots, s_m\}$. By a *random subproduct* denote a product $w = s_1^{\epsilon_1} \cdot \ldots \cdot s_m^{\epsilon_m}$ where $\epsilon_i \in \{0, 1\}$ are chosen by independent flips of a fair coin.

Lemma Let S generate group G and let $H \subset G$ be an arbitrary proper subgroup of G. Then a random subproduct w on S is not a member of H with probability at least $1/2$.

Proof Consider a largest i such that $s_i \notin H$. Then w can be decomposed into the form $u\, s_i^{\epsilon_i}\, v$, where $v \in H$.

We have two cases. If $u \in H$ then with probability $1/2$, $\epsilon_i = 1$ and $w = u\, s_i\, v \notin H$. If $u \notin H$ then with probability $1/2$, $\epsilon_i = 0$ and $w = u\, v \notin H$. This proves the lemma. □

Now one can show that with a large probability $O(\log|G|)$ random subproducts will generate the whole group G. This will be our new set of generators S_1. See [BCFLS,CF] for details and further references.

Note that the complexity of our subroutine is $O(\mu\, |S| \log|G|)$. We assume that S is reasonably small (e.g. $|S| = O(\log^3|G|)$) which is often the case in practice. Otherwise even reading the set S might take time longer than the rest of the algorithm.

7.2.2 Constructing fast generators

For a finite group G and its set of generators S define the *length* of an element $l = l_S(g)$ to be the minimum length of a decomposition $g = s_{i_1} s_{i_2} \dots s_{i_l}$. The *diameter* $D = diam(S)$ is defined as the maximum length $l_S(g)$ among all group elements $g \in G$.

Here we will present an algorithm which will construct a set of generators S_2 such that $|S_2| = O(\log|G|)$ and $D = diam(S_2) = O(\log|G|)$. The idea is to start with set of generators S_1 and repeatedly take a random walk on g and at the end of each walk add the last element into our set of generators.

For a symmetric set of generators $S = \{s_1, \dots, s_m\}$ (see §6.1) denote by $g \leftarrow rand(G, S, t)$ the state of the nearest neighbor random walk on a Cayley graph $\Gamma(G, S)$ after t steps.

Algorithm

```
1)Input $L$, $N$, $S := S_1 \cup S_1^{-1}$;
2)For $j = 1$ to $N$ Do:
    • Choose $t \in [L]$ uniformly; Let $g \leftarrow rand(G, S)$, $S \leftarrow S \cup \{g, g^{-1}\}$;
 End Do;
3)Output $S_2 := S$.
```

Now take $N = O(\log|G|)$ and $L = O(\log^3|G|)$. A hard and tedious analysis of random walks on finite groups shows that we do get a generating set S_2 such that the diameter $diam(S_2) = O(\log|G|)$ with large probability (see [Ba1]).

Note that the complexity of the algorithm is $O(\mu \log^4|G|)$

7.2.3 Final random walk

Now we have a generating set S_2, $|S_2| = O(\log|G|)$ for which we have an upper bound on diameter $D = diam(S_2) = O(\log|G|)$ (with high probability). Again consider a random walk on Cayley graph $\Gamma(G, S_2)$. It is a known result in the theory of random walks (see e.g. [DSa,Ba1]) that the total separation

$$\xi \le 16\, |S_2|\, D^2 \log|G| = O(\log^4|G|)$$

Recall that $n_{\frac{1}{2}} \le 2\xi$ (see §6.1.2). Then after $O(\log^4|G|)$ steps of the walk we get each group element with probability at least $1/2$.

To summarize, we constructed an algorithm which with large probability will produce the nearly-uniform group elements taking $O(\log^4 |G|)$ group multiplication on each. Also, assuming $|S| = O(\log^3 |G|)$ the preprocessing stage will also take $O(\log^4 |G|)$ group multiplications.

7.2.4 An Erdös-Rényi machine

Here is the last trick one can employ when there is a large number of random group elements needed to be generated. We will show that when $O(\mu \log^5 |G|)$ time used for preprocessing, then one can produce nearly-uniform group elements in time $O(\mu \log |G|)$.

Let $k = O(\log |G|)$ and using the random walk on $\Gamma(G, S_2)$ compute k nearly-uniform group elements $g_1, \dots, g_k$.

Lemma Random subproducts $u = g_1^{\epsilon_1} \cdot \dots \cdot g_k^{\epsilon_k}$ are nearly-uniformly distributed in G.

The Lemma is due to Erdös and Rényi. It implies that having spent $(\mu \log^5 |G|)$ time for finding k random group elements, we get a algorithm for producing nearly-uniform group elements at a cost of k multiplications.

7.2.5 Remarks

the first two parts of the algorithm above are Monte Carlo. We had no way of checking whether it does what it is supposed to. Put it this way, after §7.2.2 the algorithm outputs the fast generators with probability $1 - \delta$ for some δ which can be made as small as desired. On the other hand our objective was not a uniform distribution but a near-uniform distribution. So after part §7.2.2 we get *some* symmetric generating set which is fast with probability $> 1 - \delta$. Thus combined with the part §7.2.2 we *always* get near-uniform random group elements at the end. Thus our algorithm is Las Vegas.

Situation with Erdös-Rényi machine is different. Our machine is either works or it doesn't (say, the random elements we get could generate only a subgroup of G). Thus this is a Monte Carlo algorithm.

In our short presentation of the algorithm above we followed the pioneer work [Ba1] of Babai. The random subproducts were invented by Erdös and Rényi in [ER] where they also discovered the machine. The result of §7.2 originated in [BCFLS, CF].

It might be worth noting that the algorithm takes too much time to be practical. Recently there has been an attempt in [CLMNO] to find a fast alternative approach. See [Ba3, CG] for more on this algorithm.

7.3 Recognition of the black box group isomorphic to S_n

Suppose we have a black box group G. Let μ be the time necessary to multiply two elements, and let ρ be the time to generate nearly-uniform random elements of G. Assume that it takes ν time to find an order of an element on G. In the worst case we can always keep multiplying until get an identity (see also §7.3.3 below). Here we present a constructive recognition algorithm which compares G with S_n.

7.3.1 Goldbach elements

Fix the number n. Observe that if n is a prime, then every element of order n is a long cycle. On the other hand, every n-th element of S_n is a long cycle (so has an order n) so in this case we can get a long cycle just by checking the order of an element. For general n this idea wouldn't work, so we must find an alternative.

Let n be any even number and suppose $n = p_1 + p_2$ where p_1, p_2 are distinct odd primes. Then every permutation $\sigma \in S_n$ of the order $p_1 \times p_2$ is must have cycles of length p_1 and p_2. On the other hand the probability that a random permutation is of this type is equal to $\frac{1}{p_1 p_2} > \frac{4}{n^2}$, so there must be many of such elements in a group. We call them *Goldbach elements.*

Now, let us estimate the number of Goldbach elements in S_n, where n is even. Unfortunately even their existence is an unsolved problem. The Goldbach conjecture states that there must be some, but heuristically it is clear that there must be many. Indeed, if there are about $\frac{n}{\ln n}$ primes less than n they must form $\frac{n^2}{\ln^2 n}$ pairs and therefore if sums of pairs were "uniformly distributed", we would have *Const* $\left(\frac{n}{\ln^2 n}\right)$ solutions of the equation $n = p_1 + p_2$. Sylvester was the first to observe that this must be the case(see [Sy]). Later Hardy and Littlewood proved that this formula is correct for almost all n (see [HL]), so there is a reason to believe that it is true in general. In [BP] the authors computed that the number of such solutions is $> \frac{n}{3 \ln^2 n}$ for all even n, $20 < n < 10^6$. We will assume here that this is generally the case.

From the estimates above we conclude that a permutation is a Goldbach element with probability at least $\left(\frac{4/3}{n \ln^2 n}\right)$.

7.3.2 Constructing an isomorphism

Here is a general description of the algorithm.

Suppose n is even and $G \simeq S_n$. Pick a random element x and compute its order $d(x)$. Repeat the process until $d(x) = p_1 p_2$ where $p_1 + p_2 = n$. Then x is a Goldbach element with cycles of lengths p_1 and p_2. Compute these cycles: $x_1 = x^{p_1}$, $x_2 = x^{p_2}$.

Repeat picking random group elements y until $d(y) = 2\, p_3\, p_4$ where $p_3 + p_4 + 2 = n$. Then y^2 is a Goldbach element in S_{n-2} with cycles of lengths p_1 and p_2. Thus $z = y^{p_3 p_4}$ is a transposition.

Observe that for random $g \in G$ a conjugate gzg^{-1} is a random transposition. Repeat picking random elements g until $b = gzg^{-1}$ does not commute with neither of the cycles x_1 and x_2.

Now observe that $a = x_1 b x_2$ is a long cycle. Then a map $\phi : a \to (1, 2, \dots)$ and $\phi : b \to (1, 2)$ defines an isomorphism.

Compute the average expected time of this algorithm. Since finding a transposition which does not commute with x_1 and x_2 takes on average at most $O(\rho n + \mu n)$ steps, the expected time is bounded by a constant times the expected time to get a Goldbach element. Thus the total expected time is $O\big(\rho + \mu + \nu)\, n \log^2 n\big)$ steps.

7.3.3 Gray $\to$ black

Let us present an argument here that we having G presented as a gray box group was not unreasonable. In other words, we will show that there is a fast way to compute the order of black box group elements.

First, observe that never in the algorithm we needed to compute the order d which was larger than n^2. Therefore a simple idea would be simply keep multiplying until we get the identity and if we don't get it before n^2, move to the next element. There is a trick, however, how one can achieve $O(n)$ multiplications instead of n^2.

Given a black box group element a, compute two sets of elements a^n, a^{2n}, ... , a^{n^2} and id, a^{-1}, a^{-2}, ... , a^{-n}. Now if $a^{kn+i} = id$ then we would have two *identical* elements in both sets. But comparing lists of elements can be done very fast since we do not have to use all the information but just a few randomly chosen bits. If they coincide, we can check them in full then. This hashing idea is now commonly used in Computer Science (see e.g. [MR]). We conclude that in the algorithm one can assume that $\nu = O(\mu n)$.

7.3.4 Miscellanea

Our algorithm is defined when n is even. When n is odd, one can find a Goldbach element and two transpositions that connect the cycles with a fixed point. The details are given in [BP].

Note that we have yet to check that we do indeed have an isomorphism. Simply compute the Coxeter transpositions and check whether the relations are satisfied. There are only $O(n)$ relations. This proves only that $S_n \subset G$.

Now let us find the action of the original generators. First reconstruct the underlying set $[n]$ by acting on transpositions and checking whether they commute or not. Now if all the original generators are mapped to permutations, then $G \simeq S_n$. This makes the algorithm Las Vegas. The cost is $O(|S|\mu n^2)$.

One can also use Goldbach elements in trying to determine n if it is unknown at the beginning. Simply repeat picking elements and adding their prime divisors. After about $O(\rho n \ln^2 n)$ steps we shall get either n or $n-1$ because of the large number of Goldbach elements in S_n. Distinguishing between the two possibilities is easy again. Thus if n is unknown but we have an upper bound N on n (say, the length of the binary string that presents black box group elements) we can get a Monte Carlo algorithm for finding n.

7.4 Recognition of the black box group isomorphic to $SL(n, \mathbb{F}_q)$

Suppose we have a black box group G. As before, we assume that it is isomorphic to $SL(n, \mathbb{F}_q)$. Then, when we find such an isomorphism, check whether our assumption was valid.

Let μ be the time necessary to multiply two elements, and let ρ be the time to generate nearly-uniform random elements of G. Assume that it takes ν time to find an order of an element on G. In [CL] authors present an algorithm for finding the order of elements in linear groups. Their algorithm works in time $\log|G| \log\log|G|$ group multiplications. Thus in our case $|SL(n, \mathbb{F}_q)| \sim q^{n^2}$ and we have $\nu = O(n^2 \log n \log q \log\log q)$ which is sufficient for our purposes.

7.4.1 Finding a transvection and coinciding primitive element

A *transvection* is a nontrivial linear transformation of $\mathbb{F}_q^n$ that fixes an $(n-1)$-dimensional subspace (a hyperplane). The transvections play a role in linear groups that is similar to the role of transpositions in S_n. In this section we will find a transvection in a gray box $G \simeq SL(n, \mathbb{F}_q)$.

For a pair q and m define a *primitive prime divisor* (*ppd*) to be a prime p such that $p|q^m - 1$ and $p \nmid q^i - 1$ for $1 \le i \le m-1$. A known result of Zsigmondy shows that such ppd exist for all pairs $(q, m) \neq (2, 6)$.

Lemma If $x \in SL(n, \mathbb{F}_q)$, $q = p^l$ and the order $d(x) = p \cdot p' \cdot r\, p'$ is a primitive prime divisor for a pair q and $n-2$, then $x^{r\,p'}$ is a transvection. Moreover, the probability that an element of $SL(n, \mathbb{F}_q)$ satisfies the conditions above is $O\left(\frac{1}{n\,q}\right)$.

The proof which is a combination of number theoretic arguments and some linear algebra results (see [BCFL]).

Now the algorithm for finding a transvection follows directly from Lemma. Choose a random element of a gray box group G. Compute the order. Repeat the process until an element x which satisfies conditions of the lemma is found. Output $x^{d(x)/p}$.

By the second part of the lemma the running time of the algorithm is $O((\nu + \rho)\, n\, q)$ steps. We assume here that the decomposition into prime is relatively fast compared to the finding a random group element. The reason lies in a special structure of the numbers $q^m - 1$ which have been studied heavily in computational number theory.

Consider a *coinciding primitive element* $z = x^p$. We will need it in the next section for the final construction. Heuristically it plays the same role as an $(n-2)$-cycle in the case when $(n-2)$ is a prime.

7.4.2 Finding elementary transvections

The idea is to construct an isomorphism which maps certain gray box group elements into elementary transvections. By elementary transvections we mean matrices in $SL(n, \mathbb{F}_q)$ with ones on diagonal and only one nonzero element elsewhere.

The technique presented in [BCFL] is nice but rather laborious so we will just give a short overview.

First, authors observe that by conjugating transvection x with by random element one gets random transvections. Second, a commutator $x\,y\,x^{-1}y^{-1}$ of x with a random transvection with probability about $1/q$ is a transvection which fixes the same hyperplane as x. The last property is also not hard to check. Now, after such a transvection y is found taking commutators of y with powers of the coinciding primitive element z will lead to $n-1$ "linearly independent transvections" that all fix the same hyperplane. The authors in [BCFL] then define a linear algebra technique for working with transvections.

Now the process is repeated for n other hyperplanes and finally authors employ a version of orthogolization algorithm to get the desired elementary transvections. This is just the rough idea of the technique used.

The overall algorithm takes $O(n^3\epsilon + n^2 q\mu + \rho\mu)$ where ϵ is the time for field operations in $\mathbb{F}_q$. The way the algorithm is presented, it is a Monte Carlo algorithm. It can be made Las Vegas by checking the relations at the end. This can be done at low cost. The minimal set of relations for $SL(n, \mathbb{F}_q)$ can be found in [CM].

7.4 Remarks

The recognition problem of the black box groups has been recently studied in a number of papers (see e.g. [CFL,NP1,KS]).

In our presentation of the recognition algorithm of the gray box group isomorphic to S_n we follow the paper [BP]. A similar idea can be used to find a recognition in case of the alternating group A_n.

The sketch of the recognition algorithm of the gray box group isomorphic to $SL(n, \mathbb{F}_q)$ follows the paper [BCFL]. The paper generalizes the approach started in [CFL]. The alternative version of the recognition algorithm for all Chevalley groups was developed in [KS]. Their result is quite general but in case of $SL(n, \mathbb{F}_q)$ the algorithm is significantly slower.

References

[A1] D. Aldous, *Random walks on finite groups and rapidly mixing Markov chains*, Lecture Notes in Mathematics **986** (1983).

[A2] D. Aldous, *Random walks on finite groups and rapidly mixing Markov chains*, J. Theor. Probability **2** (1980), 91–100.

[AD1] D. Aldous, P. Diaconis, *Shuffling cards and stopping times*, Amer. Math. Monthly **93** (1986), 333–348.

[AD2] D. Aldous, P. Diaconis, *Strong uniform times and finite random walks*, Advances in Applied Math. **8** (1987), 69–97.

[AF] D. Aldous, J. Fill, *Reversible Markov Chains and Random Walks on Graphs*, monograph in preparation, 1996.

[An] G. Andrews, *The Theory of Partitions*, Addison-Wesley, New York, 1976.

[Ari] A. D'Aristotile, *The nearest neighbor random walk on subspaces of a vector space and rate of convergence*, J. Theoretical Probability **8** (1995), 321–346.

[Arn] V. I. Arnol'd, *The cohomology ring of the group of dyed braids*, Mat. Zametki **5** (1969), 227–231.

[AP] A. Astashkevich, I. Pak, *Random walks on nilpotent and supersolvable groups*, in preparation (1997).

[Ba1] L. Babai, *Local expansion of vertex-transitive graphs and random geneartion in finite groups*, in Proc 23^{rd} ACM STOC (1991), 164–174.

[Ba2] L. Babai, *Automorphism groups, isomorphism, reconstruction*, in Handbook of Combinatorics (R. L. Graham, M. Groetschel, and L. Lovasz, eds.) (1996), Elsevier.

[Ba3] L. Babai, *Randomization in group algorithms: Conceptual questions*, in Groups and Computation II (L. Finkelstein, W.M. Kantor, eds.) DIMACS Workshops on Groups and Computation (1997), AMS, Providence.

[BCFLS] L. Babai, G. Cooperman, L. Finkelstein, E.M. Lucks, and A. Seress, *Fast Monte Carlo algorithms for permutation groups*, Proc 23-rd ACM STOC (1991), 90–100.

[BE] H. Bateman, A. Erdélyi, *Higher Transcedential Functions, Volume 1*, Mc Graw-Hill, New York, NY, 1953.

[Be] E. Belsley, *Random walks on distance regular graphs* (1996), in preparation.

[Bor] A. Borel et al, *Seminar on algebraic groups and related finite groups*, Springer, Berlin, 1970.

[Bou] N. Bourbaki, *Groupes et algèbres de Lie, Ch I, IV, V, VI*, Hermann, Paris, 1960.

[BP] S. Bratus, I. Pak, *Fast constructive recognition of a gray box group isomorphic to S_n or A_n using Goldbach's Conjecture* (1997), preprint.

[BCFL] S. Bratus, G. Cooperman, L. Finkelstein, and S. Linton, *Constructive recognition of black box groups isomorphic to $SL(n,q)$ or $PSL(n,q)$.* (1997), preprint.

[CW] E. Calabi, H. Wilf, *On the sequential and random selection of subspaces over a finite field*, J. Comb. Theory (A) **22** (1977), 107–109.

[Ca] P. Cartier, *Lecture Notes in Math., 901*, vol. **1980/81**, Springer, Berlin, pp. 1–22.

[CL] F. Celler, C.R. Leedham-Green, *Calculating the Order of an Invertible Matrix*, in Groups and Computation II (L. Finkelstein, W.M. Kantor, eds.) DIMACS Workshops on Groups and Computation (1997), AMS, Providence.

[CLMNO] F. Celler, C.R. Leedham-Green, S. Murray, A. Niemeyer, and E.A. Obrien, *Generating random elements of a finite group*, Comm. Alg. **23** (1995), 4931–4948.

[Che] I.V. Cherednik, *A new interpretation of Gelfand-Tzetlin bases*, Duke Math. J. **54** (1987), 563–571.

[CG] F.R.K. Chung, R.L. Graham, *Random walks on generating sets for finite groups*, The Electronic J. of Comb. **4 No 2.** (1997), #R7.

[CF] G. Cooperman, L. Finkelstein, *Combinatorial tools for computational group theory*, in Groups and Computation I (L. Finkelstein, W.M. Kantor, eds.) DIMACS Workshops on Groups and Computation (1993), AMS, Providence.

[CFL] G. Cooperman, L. Finkelstein, and S. Linton, *Constructive recognition of a black box group isomorphic to GL(n , 2)*, in Groups and Computation II (L. Finkelstein, W.M. Kantor, eds.) DIMACS Workshops on Groups and Computation (1997), AMS, Providence.

[CM] H.S.M. Coxeter, W.O.J. Moser, *Generators and relations for discrete groups (third edition)*, Springer, Berlin, 1972.

[D1] P. Diaconis, *Group Representations in Probability and Statistics*, IMS, Hayward, California, 1988.

[D2] P. Diaconis, *A group theoretic interpretation of the records to cycles map* (1995), unpublished manuscript.

[D3] P. Diaconis, *The cutoff phenomenon in finite Markov chains*, Proc. Nat. Acad. Sci. U.S.A. **93** (1996), 1659–1664.

[DF] P. Diaconis, J. Fill, *Strong stationary times via a new form of duality*, Ann. Prob. **18** (1990), 1483–1522.

[DSa] P. Diaconis, L. Saloff–Coste, *Comarison techniques for random walk on finite groups*, The Annals of Probability **21** (1993), 2131–2156.

[DSh1] P. Diaconis, M. Shahshahani, *Generating a random permutation with random transpositions*, Z. Wahr. verw. Gebiete **57** (1981), 159–179.

[DSh2] P. Diaconis, M. Shahshahani, *The subgroup algorithm for generating uniform random variables*, Prob. in Eng. and Info. Sci. **1** (1987), 15–32.

[DSh3] P. Diaconis, M. Shahshahani, *Time to reach stationarity in the Bernoulli–Laplace diffusion model*, SIAM J. Math'l Analysis **18** (1987), 208–218.

[Dem] M. Demazure, *Désingularisation des variétes de Schubert généralisées*, Ann. Sci. Éc. Norm. Sup. **7** (1974), 53–88.

[EKK] V.A. Emelichev, M.M. Kovalev, M.K. Kravtsov, *Polytopes, Graphs and Optimization*, Cambridge University Press, New York, 1984.

[ER] P. Erdös, A. Rényi, *Probabilistic methods in group theory*, J. d'Analyse Math. **14** (1965), 127–138.

[Eu] L. Eulero, *Introductio in Analysin Infitorum*, Marcum-Michaelem Bousquet et Socios, Lausanne, Switzerland, 1748.

[F] W. Feller, *An introduction to Probability theory and its applications (third edition)*, John Wiley, New York, 1970.

[FOW] L. Flatto, A. M. Odlyzko, D. B. Wales, *Random shuffles and group representations*, Ann. Prob. **13** (1985), 155–178.

[Gl] D. Gluck, *Characters and random walks on finite classical groups*, Advances in Mathematics, to appear (1995).

[GR] J. Goldman, G.-C. Rota, *Recent Progress in Combinatorics*, Acad. Press, New York, 1969, pp. 75–83.

[GJ] I. P. Goulden, D. M. Jackson, *Combinatorial enumeration*, John Wiley, New York, 1983.

[Gr] A. Greenhalgh, *Random walks on groups with subgroup invariance properties, Ph. D. dissertation*, Stanford U., 1987.

[GH] D. Griffiths, J. Harris, *Principles of Algebraic Geometry*, John Wiley, New York, 1978.

[HL] G.H. Hardy, J.E. Littlewood, *Some problems of 'Partitio Numerorum'; III: On the expression of a number as a sum of primes*, Acta Math. **13** (1944), 1–70.

[Hi] M. Hilderbrand, *Generating random elements in $SL_n(\mathbb{F}_q)$ by random transvections*, J. Alg. Combinatorics **1** (1992), 133–150.

[H1] J. Humphreys, *Linear algebraic groups*, Springer, Berlin, 1975.

[H2] J. Humphreys, *Reflection groups and Coxeter groups*, Cambridge University Press, Cambridge, UK, 1990.

[J] A. Jucys, *On the Young operators of the symmetric group* (in Russian), Lietuvos Fizikos Rinkinys **6** (1966), 163–180.

[KS] W. Kantor, A. Seress, *Black box classical groups*, preprint (1997).

[K] D. E. Knuth, *The Art of Computer Programming,* Vol. 2, Seminumerical Algorithms, Addison-Wesley, 1969.

[LS] A. Lascoux, M.-P. Schützenberger, *Polynômes de Schubert*, C. R. Acad. Sci. Paris **294** (1982), 447–450.

[M1] I. G. Macdonald, *Symmetric Functions and Hall Polynomials*, Oxford University Press, London, 1979.

[M2] I. G. Macdonald, *Notes on Schubert Polynomials*, Université du Québec, Montréal, 1991.

[Ma] P. Matthews, *A strong uniform time for random transpositions*, J. Theoretical Probability **1** (1988), 411–423.

[MR] R. Motwani and P. Raghavan, *Randomized Algorithms*, Cambridge University Press, Cambridge, UK, 1995.

[N] M. Nazarov, *Young's Symmetrizers for Projective Representations of the Symmetric Group*, RIMS Preprint No 900, Kyoto University, Kyoto, Japan, 1992.

[NP1] P. Neumann, C. Praeger, *A recognition algorithms for special linear groups*, Proc. London Math. Soc. **65** (1992), 555 – 603.

[NP2] P. Neumann, C. Praeger, *Cyclic matrices over finite fields*, J. London Math. Soc. (2) **52** (1995), 263–284.

[NW] A. Nijenhuis, H. Wilf, *Combinatorial Algorithms*, Academic Press, New York, 1978.

[P] I. Pak, *Random walks on permutation: strong uniform time approach*, Ph.D. Thesis, Harvard U., 1997.

[R] D. Randall, *Efficient random generation of nonsingular matrices*, Random Structures and Algorithms **4** (1993), 111–118.

[RND] E. Reingold, J. Nievergelt, N. Deo, *Combinatorial Algorithms*, Prentice-Hall, Englewood Cliffs, New Jersey, 1977.

[S1] R. P. Stanley, *Enumerative Combinatorics,* Vol. 1, Wadsworth & Brooks/Cole, California, 1986.

[S2] R. P. Stanley, *Some combinatorial aspects of the Schubert calculus, in*, Lecture Notes in Math. **579** (1971), Springer, Berlin, 225–259.

[SW] D. Stanton, D. White, *Constructive Combinatorics,* Undegraduate Texts in Mathematics, Springer–Verlag, Berlin, 1986.

[St] R. Steinberg, *Lectures on Chevalley groups*, Yale U., 1967.

[Sy] J.J. Sylvester, *On the partition of an even number into two primes*, Proc. London Math. Soc. (Series 1) **4** (1871), 4–6.

[W] H. Weyl, *Classical Groups*, Princeton University Press, Princeton, NJ, 1939.

[WW] E. T. Whittaker, G. N. Watson, *A Course of Modern Analysis* (Fourth Edition), Cambridge University Press, Cambridge, UK, 1927.

[Z] G. Ziegler, *Lectures on Polytopes, Graduate Texts in Mathematics 152*, Springer, New York, 1995.

Department of Mathematics, Yale University, New Haven CT 06520
E-mail address : **paki@math.yale.edu**