

### Hamiltonian graphs

A Hamiltonian cycle is a closed walk which visits each vertex *exactly* once. (Contrast this with Eulerian cycles which visit each edge exactly once.) To prove that a graph is Hamiltonian we only need to show the closed walk visiting each vertex exactly once; on the other hand to show that a graph is not Hamiltonian takes an argument. The graph  $K_n$  is Hamiltonian for all  $n$ ;  $K_{m,n}$  is Hamiltonian if and only if  $m = n$ ; the Petersen graph is not Hamiltonian.

The hypercube  $Q_n$  is Hamiltonian for all  $n$ . A Hamiltonian walk in a hypercube is a Gray code (an arrangement of the binary words of length  $n$  so that any two consecutive words differ in exactly one entry).

### Adjacency matrix

We can use the *adjacency matrix*  $A$  to represent a graph. The rows and columns are indexed by the vertices and the entries indicate adjacency relationships. Namely we have

$$A = i \begin{pmatrix} & j & & \\ & \vdots & & \\ \cdots & a_{i,j} & \cdots & \\ & \vdots & & \end{pmatrix}$$

where  $a_{i,j}$  counts the number of edges joining the  $i$ th vertex to the  $j$ th vertex (for simple graphs this will be a 0-1 matrix). The graph  $A$  is not unique since reordering vertices can give a different matrix.

If the graph is undirected the matrix  $A$  is symmetric; the row sums and the column sums give the degrees. If the graph is directed the matrix  $A$  does not have to be symmetric; the row sums give the out-degrees and the column sums give the in-degrees. If the matrix is of the form

$$A = \left( \begin{array}{c|c} O & B \\ \hline B^T & O \end{array} \right),$$

where  $B^T$  is the transpose of  $B$ , then the graph is bipartite.

Adjacency matrices are useful to count the number of walks joining two vertices. Namely we have

$$(A^n)_{i,j} = \# \text{ walks joining } i \text{ and } j.$$

The diagonal of  $A^2$  counts the degree. The trace of  $A^3$  is equal to six times the number of triangles.

A related matrix is the *incidence matrix*  $Q$  which is a 0-1 matrix with the rows indexed by the vertices and columns indexed by the edges. The entry  $q_{i,j}$  is 1 if and only if the corresponding vertex is incident to the corresponding edge. If the graph is a simple graph then  $QQ^T = D + A$  where  $D$  is the diagonal degree matrix while  $Q^TQ = 2I + A_{L(G)}$  where  $L(G)$  is the line graph of  $G$ .

---

### Isomorphism of graphs

One graph can be represented in different ways (i.e., as different drawings) so given two representations of graphs how do we decide if they are the same graph or a different graph?

We say that two graphs  $G_1$  and  $G_2$  are isomorphic if there are 1-1, onto functions  $f : V(G_1) \rightarrow V(G_2)$  and  $g : E(G_1) \rightarrow E(G_2)$  so that edge  $e$  is incident to vertex  $v$  if and only if  $g(e)$  is incident to  $f(v)$ . If the graph is simple we only need the function  $f$  and show that  $v$  is adjacent to  $u$  if and only if  $f(v)$  is adjacent to  $f(u)$ .

In other words two graphs are isomorphic if we can relabel the vertices of one graph to get the other. So if two graphs are isomorphic we only need to find a relabeling. To show that two graphs are *not* isomorphic we need to find a property which is not shared by the two graphs but should be if they were isomorphic (these properties are called *invariants*). Examples of invariants include number of vertices, number of edges, degree sequences, Hamiltonian, Eulerian, planar, connected, having a simple cycle of length  $k$ , and so on.

### Planar graphs

A graph  $G$  is planar if it can be drawn in the plane without edges crossing (but not all ways we draw it in the plane have to avoid crossing edges). The graphs  $K_{3,3}$  and  $K_5$  are *not* planar. Further, any graph is planar if and only if it does not contain a subgraph homeomorphic (i.e., by a series of edge contractions) to  $K_{3,3}$  and  $K_5$ .

A connected planar graph divides the plane up into faces. If we were to draw the planar graph on a piece of paper with no two edges crossing and then cut along the edges the number of faces is the number of pieces of paper we have after cutting. An important formula for planar graphs is Euler's formula

$$V - E + F = 2,$$

where  $V$  is the number of vertices,  $E$  the number of edges and  $F$  the number of faces. This can be used to show, for example, that  $K_{3,3}$  and  $K_5$  is not planar and also that the number of Platonic solids is five.

The chromatic number of a graph  $G$ , denoted  $\chi(G)$ , is the minimum number of colors needed to color the vertices so that no two adjacent vertices have the same color. We have  $\chi(K_n) = n$  and  $\chi(K_{m,n}) = 2$  (more generally if  $G$  is bipartite then  $\chi(G) = 2$ ). A famous result in graph theory is that if  $G$  is planar then  $\chi(G) \leq 4$ .

---

### Trees

A tree  $T$  is a special type of graph. There are several ways to characterize a graph on  $n$  vertices which is a tree including:

- Between any two vertices there is a *unique* simple path connecting them.

- The graph is connected and acyclic (i.e., no cycles).
- The graph is connected and has  $n - 1$  edges.
- The graph is acyclic and has  $n - 1$  edges.

The name “tree” comes from the idea of branching. Because of this trees are useful in representing hierarchical structures (such as files on a computer) or modeling sequence of decisions.

A rooted tree is a tree with a special vertex designated as its root. The level of a vertex in a rooted tree is the distance of the vertex to the root (distance measured in number of edges in the unique simple path connecting them). The height of a tree is the largest level.

All trees are planar and bipartite (to see that they are bipartite we lump the vertices into two groups, namely those at an even level and those at an odd level). Every tree on 2 or more vertices will always have at least 2 vertices of degree 1. Such vertices are called leaves, or sometimes terminal nodes.

In a rooted graph if  $u$  is adjacent to  $v$  and  $v$  is at a “lower” level (actually at a level one greater than  $u$ , this is because we tend to put the root vertex at the top and grow the tree down). Then we say that  $u$  is a parent of  $v$  or that  $v$  is a child of  $u$ . A vertex with children is an internal node. A subtree of a rooted tree is taken by taking the tree consisting of that vertex and all of its “descendants”.

---

### Spanning trees

Given a graph  $G$  a spanning tree  $T$  is a subgraph which is a tree and uses all of the vertices of  $G$  (i.e., it spans the vertices). A graph has a spanning tree if and only if it is connected.

Given a graph there are several ways to form a spanning tree.

1. Continue removing edges from graph as long as that edge does not disconnect the graph. Resulting graph is a spanning tree.
2. (Breadth first search) Pick an initial vertex  $v$  which forms tree  $T_0$ . Now given  $T_i$  we grow  $T_{i+1}$  by finding all vertices incident to  $T_i$  but not in  $T_i$ , add these vertices and edges connecting them to  $T_i$  to form  $T_{i+1}$ . Continue until all vertices are found.
3. (Depth first search) Pick an initial vertex  $v$  which forms tree  $T_0$  and we note that  $v$  is the active vertex of this tree. Given a tree  $T_i$  and an active vertex  $u$  on that tree we look to see if any vertices adjacency to  $u$  are not in the tree, if so we add that vertex and edge to form  $T_{i+1}$  and make the new vertex the active vertex. If not, we move up to the parent of the active vertex and similarly see if any of its adjacent vertices are not in the

tree. If not, we continue moving up through the ancestors. If no ancestor is adjacent to a vertex not in the tree then we are done.

The idea of depth first search (badly explained here) is related to the idea of backtracking in trees. Namely we can model a problem as a sequence of decisions and then we search the tree looking for a valid solution. We do this by systematically going down the branches, if we get stuck we go back far enough to find a different branch to try. We continue this process until we have either found a solution or examined the whole tree and can conclude no solution exists.

---

### Minimal spanning trees

A *weighted* graph is a graph where each edge has been given a weight. The minimum spanning tree problem is to find the spanning tree where the sum of the edge weights is as small as possible.

This problem has many nice solutions (and many nice applications), all of them based on *greedy algorithms*. A greedy algorithm is one which at each stage we do the current best possible option. (Note that in some problems being greedy at one decision can later on force you to make a bad decision so being greedy is not always the right thing to do.)

Below are some algorithms for finding the minimal spanning trees.

1. Start with no edges in the “tree”. Among all edges in the graph not currently in the tree find the one with lowest weight which does not create a cycle in the tree and add that edge to the tree. Repeat until the resulting graph spans all the vertices.
2. Start with a tree consisting of a single vertex  $v$ . We now grow our tree one edge at a time by adding the edge incident to exactly one vertex in our current tree with lowest weight. Repeat until the resulting graph spans.
3. Start with all of the edges of the graph. We continue at each stage to remove the edge with the largest weight whose removal does not disconnect the graph. Repeat until resulting graph is acyclic.

(The difference between the first two is that the first one always looks in the whole graph for the smallest edge to add next while the second one only looks near what we already have. So in some sense one is global and the other is local.)

---

### Isomorphism of trees

Two trees are isomorphic if they are isomorphic as graphs. Two rooted trees are isomorphic if the graphs are isomorphic and further the function which maps the vertices of the first tree to the vertices of the second tree maps the root of the first tree to the root of the second tree.

The number of labeled trees on  $n$  vertices with labels  $\{1, 2, \dots, n\}$  is  $n^{n-2}$ . To do this we can use Prüfer codes which takes every tree on  $n$  vertices to a list of  $n - 1$  numbers by the following rule: Find the lowest number among all leaves in the current tree, erase that leaf and write down the label of its neighbor, continue until the tree has no leaves (i.e., consists of a single vertex). The first  $n - 2$  number can be any of  $1, \dots, n$  but the  $(n - 1)$ th number *must* be  $n$ . In particular there are  $n^{n-2}$  possible Prüfer codes. On the other hand each Prüfer code corresponds to a unique tree. Given Prüfer code  $a_1 a_2 \dots a_{n-1}$  form the sequence  $b_j$  by

$$b_j = \min \{k \mid k \notin \{b_1, \dots, b_{j-1}, a_j, \dots, a_{n-1}\}\}.$$

Then the tree is formed by the edges  $\{a_i, b_i\}$ .

### Binary trees and traversal of trees

A binary tree is a tree where every vertex has 0, 1 or 2 children. Further each child is labeled either left or right (but not both). A full binary tree is a tree where every vertex has exactly two children. In a full binary tree with  $i$  internal vertices there are  $i + 1$  leaves.

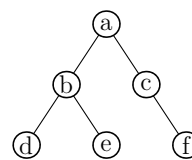
A binary tree of height  $h$  has at most  $2^h$  vertices.

There are several ways to systematically visit the vertices of a tree (known as traversing the tree).

- *Preorder traversal*  
Visit the root.  
Do a preorder traversal on the subtree of the left child.  
Do a preorder traversal on the subtree of the right child.
- *Inorder traversal*  
Do an inorder traversal on the subtree of the left child.  
Visit the root.  
Do an inorder traversal on the subtree of the right child.
- *Postorder traversal*  
Do a postorder traversal on the subtree of the left child.  
Do a postorder traversal on the subtree of the right child.  
Visit the root.

We can also find preorder by starting at the root and walking along the graph clockwise and listing vertices in order of the first time we encounter them. Similarly the postorder can be found by starting at the root and walking along the graph clockwise and listing vertices in order of the last time we encounter them.

For the following graph the preorder is “abdefc” the inorder is “dbeafc” and the postorder is “debfc”.



Merge-sort

Given a list of elements there are several ways to arrange them into increasing order (i.e., to sort). One method of sorting is MERGE-SORT which does the following.

Given a list  $L$  of elements to sort, if  $|L| = 1$  then we are done since it is already sorted. Otherwise divide the list into two parts  $L_1$  (of size  $\lfloor |L|/2 \rfloor$ ) and  $L_2$  (of size  $\lceil |L|/2 \rceil$ ). Sort the elements of  $L_1$  and  $L_2$  using MERGE-SORT, finally merge the two (now sorted) lists together.

This method uses order  $\Theta(n \ln n)$  in the worst case, and in some sense is optimal in that any sorting method requires using at least  $\Theta(n \ln n)$  in the worst case. Of course there are other sorting methods which achieve this bound and deciding on which sorting method is best is a non-trivial problem.

### Probability

Probability is a measurement of how likely certain events are to occur. We start with a *probability space*  $X$  whose elements consist of all possible outcomes and associated with each element  $x$  is a probability  $P(x)$  where  $0 \leq P(x) \leq 1$  and further we have that  $\sum_{x \in X} P(x) = 1$  (i.e., we have portioned up 1 among the elements of  $X$ ). An *event* is a subset  $A \subseteq X$  of the outcomes. We then have that  $P(A) = \sum_{x \in A} P(x)$  is the probability that an event in  $A$  occurs.

Unless specified we will assume that each outcome is equally likely, or that it is a uniform distribution. This is sometimes denoted using the word “fair” as in “a fair dice” or “a fair coin”. In this case, probability reduces down to counting. Namely we have

$$P(A) = \frac{|A|}{|X|}.$$

We have that the probability that an event both in  $A$  and in  $B$  occurs is  $P(A \cap B)$  while the probability that an even in either  $A$  or in  $B$  occurs is  $P(A \cup B)$ .

Conditional probability looks at the probability that an event  $B$  happens given the you know even  $A$  happened. This is written  $P(B|A)$  and read “the probability of  $B$  given  $A$ ”. In this setting we now restrict our probability space to  $A$  and so to find  $P(B|A)$  we simply need to find “how much” of  $A$  is  $B$ , i.e.,

$$P(B|A) = \frac{P(A \cap B)}{P(A)}.$$

On the other hand we also have

$$P(A|B) = \frac{P(A \cap B)}{P(B)}.$$

If we combine these two we get Bayes Theorem

$$P(B|A) = \frac{P(A|B)P(B)}{P(A)}.$$

(Note that using a decision tree can be useful to help us keep track of these numbers.)

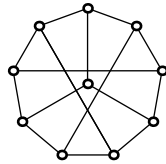
If we can partition the probability space into disjoint events  $A_1, A_2, \dots, A_k$  then

$$P(B) = P(B|A_1)P(A_1) + \dots + P(B|A_k)P(A_k).$$

We say that two events  $A$  and  $B$  are independent if knowing the outcome of  $A$  does not effect the outcome of  $B$ , i.e.,  $P(B|A) = P(B)$ . Putting this into the equation above this translates into saying that two events are independent if  $P(A \cap B) = P(A)P(B)$ .

### Sample questions

1. A Hamiltonian path is a path which visits each vertex exactly once, further if the graph is directed we always go in the direction indicated by the edge. A tournament is a directed graph where between two vertices  $u$  and  $v$  we have either  $u \rightarrow v$  or  $v \rightarrow u$  (but not both). Use induction to show that every tournament on  $n \geq 1$  vertices has a (directed) Hamiltonian path.
2. A simple graph is said to be *strongly regular* if it is regular of degree  $k$ ; if  $u$  and  $v$  are adjacent then there are exactly  $\mu$  vertices adjacent to both  $u$  and  $v$ ; and if  $u$  and  $v$  are not adjacent then there are exactly  $\nu$  vertices adjacent to both  $u$  and  $v$ . If  $A$  is the adjacency matrix of a strongly regular graph, find the entries of  $A^2$  in terms of  $k, \mu, \nu$ .
3. Show that the Petersen graph is strongly regular.
4. Give an example of two graphs on six vertices which are regular of degree 3 (i.e., each vertex has degree 3) and are *not* isomorphic. Give an invariant that one graph has that the other does not.
5. Show that the following graph is isomorphic to the Petersen graph by some appropriate relabeling.



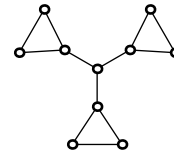
6. True/False: The chromatic number of a graph is an invariant. Justify your answer.
7. Find the chromatic number of the Petersen graph.
8. A planar graph is triangulated if every face is a triangle. Show that in a connected planar triangulated graph

$$F = 2V - 4.$$

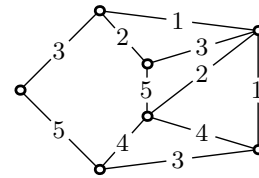
9. A planar graph is cubic if every vertex has degree three. Show that in a connected planar cubic graph

$$V = 2F - 4.$$

10. The preceding two problems are related by considering “dual graphs”. Given a connected planar graph  $G$  the dual graph  $H$  is the graph formed by replacing every face of  $G$  by a vertex and connecting two vertices if the corresponding faces share an edge. (So for example connected planar triangulated graphs become connected planar cubic graphs and vice-versa) Show that if  $H$  is the dual of  $G$  then  $H$  and  $G$  have the same number of edges.
11. The crossing number of a graph is the minimum number of edges that must cross in a drawing of the graph. If the graph is planar the crossing number is zero. Show that the crossing number of  $K_5$  and  $K_{3,3}$  is one by giving drawings where there is only one edge crossing.
12. Given a tree let  $\kappa$  be the length of the longest simple path in the tree. Show that if we root the tree at some vertex then the height  $h$  of the tree satisfies  $\lceil \kappa/2 \rceil \leq h \leq \kappa$ .
13. Given a rooted tree  $T$  we can define a relation on the vertices by saying  $u$  is related to  $v$  if  $u$  is a descendant of  $v$ . Is this relationship symmetric, anti-symmetric, transitive, and/or reflexive?
14. How many different (labeled) spanning trees does the following graph have?



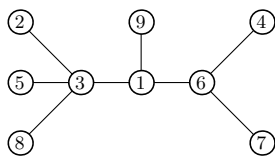
15. Find a minimal spanning tree and its corresponding total edge weight for the following graph.



16. Instead of finding a minimal spanning tree we can find a maximal spanning tree, namely the spanning tree with maximum edge sum (for example a corrupt official might have been bribed to choose the most expensive option in a construction project). The same greedy approach works as in the case of the minimal spanning tree problem. Outline an algorithm that will find a maximal spanning tree of a graph. Use the algorithm on

the graph given in the previous problem to find the maximal spanning tree and its corresponding total edge weight.

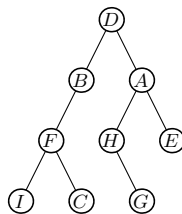
17. Find the Prüfer code for the following tree.



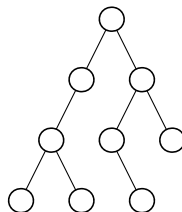
18. Construct the tree corresponding to the following Prüfer codes (on 9 vertices):

- (a) 81854629.  
 (b) 36527519.

19. Find the preorder, inorder and postorder of the following tree.



20. Label the vertices of the following tree so that the preorder of the vertices will be *ABCDEFGHI*.



21. Repeat the previous problem for the inorder and the postorder

22. Let reverse-inorder be the traversal which visits vertices by the following rule: First do a reverse-inorder traversal on the right tree; second visit the root; third do a reverse-inorder traversal on the left tree. Find the reverse-inorder traversal for the tree in problem 19.

What is the relationship between inorder and reverse-inorder.

23. If the preorder *and* the inorder for a tree are both *ABCDEFGHIJKLMN*, what is the postorder?

24. You flip seven fair pennies. What is the probability that *at most* two of the coins are heads?

25. Twenty cookies are distributed among five people. What is the probability that every person got *at least* two cookies.

26. Four boys and three girls sit down on a row of chairs. What is the probability that no two of the girls sit next to each other?

27. Suppose that you have an urn which initially has two blue marbles and one red marble. You draw out a marble. If it is blue then replace it, if it is red replace it and add three more red marbles. Now draw a second marble.

- (a) What is the probability that the first marble you drew was red given that the second marble is red?  
 (b) What is the probability that the first marble you drew was red given that the second marble is blue?  
 (c) What is the probability that the first marble you drew was blue given that the second marble is red?  
 (d) What is the probability that the first marble you drew was blue given that the second marble is blue?

28. You roll a fair die with sides labeled 1, 2, 3, 4, 5, 6. Whatever number is rolled you then pick up that number of fair pennies and flip them.

- (a) What is the probability that you flip exactly two heads among the pennies you flipped?  
 (b) Given that you flipped two heads, what is the probability that you rolled a 4 on the die?

29. A new disease has been discovered called *senioritus* which causes people in their last year of college to stop working hard, this disease afflicts 2% of all seniors. A new test has been discovered to help detect the affliction. The test is remarkably good in that 100% of people afflicted with senioritus will test positive while 95% of people not afflicted with senioritus will test negative.

You have just taken the test and were tested positive. What is the probability that you have senioritus?

30. Suppose that you have a biased coin that with probability  $p$  comes up head and with probability  $1 - p$  comes up tails.

- (a) What is the probability that if you flip the coin three times that heads will come up exactly twice?  
 (b) Find the value  $p$  so that the probability in part (a) is maximized.

31. You roll two fair six-sided die.

- (a) Are the events that the first die is a 2 and that the sum of the two die is 7 independent?  
 (a) Are the events that the first die is a 2 and that the sum of the two die is 8 independent?