Math 118 −− Useful information concerning Midterm #2

The test will be a 70−minute in−class exam held on Monday, November 11. You may not use any notes or books. You will need a calculator! Please come to class a few minutes early, so that you don't forfeit any of your 60 minutes.

Make up tests will be allowed only if pre−arranged. If you are ill, please email or call me in my office (632−8358) **BEFORE** the test.

The exam will cover sections 1,2(no house odds and fair bets),3,4,5 of chapter 4 and sections 1,2 (only till the Greedy Algorithm) of chapter 6.

The following is a partial list of concepts and "skills" you may need during the exam:

· A "**sample space**" (denoted **S**) is a list of all possible **outcomes** of an **experiment**. The favorable "**events**" (denoted **E**) form a subset of the sample space. For example, in the experiment of rolling a die, the sample space is **S**={1,2,3,4,5,6}, and the collection of favorable events that the die rolls an even number is **E**={2,4,6}.

· The most important rule of probability is this: if all outcomes in a sample space are equally likely, then **P(E) = n(E)/n(S)**. Continuing the previous example, the probability of rolling an even number is 3/6.

· The probability that something does not happen is 1 – the probability that it does happen. For example, since there is a 1/6 chance that a die will roll "3", there must be a 5/6 chance that the die will roll something other than a "3". If you are having trouble calculating the probability that something happens, ask yourself whether it is easier to calculate the probability that the opposite thing happens!

· Two events are called **mutually exclusive** if they cannot both happen; in other words, if they are disjoint subsets of the sample space. For example, if I roll two dice, then the event that the dice sum to 6 and the even that the dice sum to 7 are mutually exclusive (they can't both happen at the same time).

· The **Addition Rule** states that, if A and B are events in a sample space, then **P(A or B) = P(A) + P(B) – P(A and B)**. Notice that when A and B are mutually exclusive, P(A and B) is zero, so we just have that **P(A or B) = P(A) + P(B)**.

· To find the chance that two **independent things** will both happen, you should **multiply** their chances. For example, what is the chance that a coin flips heads AND a die rolls 6? P(heads and 6) = P(heads).P(6) = (1/2).(1/6) = 1/12. In this example, the die roll and the coin flip are independent because they have nothing do with each other; knowing how the coin flip turns out doesn't affect the outcome of the die roll.

· Even when two things are dependent, you can still multiply their chances. The **multiplication rule**, which involves conditional probabilities, says that the chance that two things will both happen is the chance that the first thing will happen times the chance that the second thing will happen, given that the first happened. For example, if I take two cards from a deck, what's the chance they are both hearts? P(both are e) = P(first is e) . P(second is e | first is e) = (13/52).(12/51) = 5.88%.

· In the last example, P(second is e | first is e)= 12/51 is called a **conditional probability**, and means the chance that the second card is a heart given that the first card was a heart. More precisely, if A and B are any events in a sample space S, then the conditional probability is **P(B|A)=n(A and B)/n(A)**. Think of P(B|A) as telling you the chance that B is true given that you know A is true (so only the outcomes in A are relevant for figuring the probability, hence the formula). The events A and B are called **independent** if **P(B|A) = P(B)**, which says that "knowing A is true doesn't change the odds at which you're willing to bet that B is true".

The precise version of the **multiplication rule** says that **P(A and B) = P(A) P(B|A)=P(B) P(A|B)**.

· The **Basic Counting Law** says that if there are M possible ways a first task can be performed and, after the first task is complete, there are N possible ways for a second task to be performed, then there are M.N possible ways for the two tasks to be performed in that order. For example, there are 52.51 = 2652 outcomes that the experiment of drawing two cards from a deck could turn out.

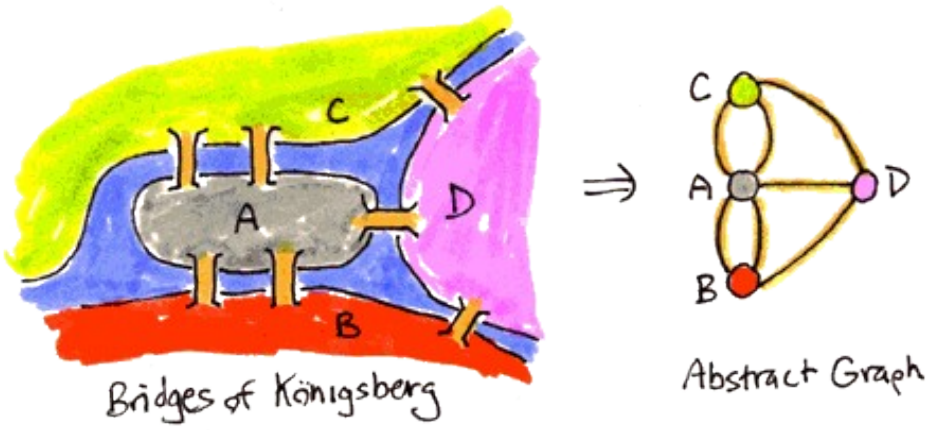
· **n!** denotes the product of all the numbers between 1 and n. There are **n!** possible ways to **permute** (rearrange) a list of n objects. For example, there are 5! = 120 words that can be spelled by rearranging the letters A,B,C,D,E.

· The number of **permutations of n objects taken k at a time** is denoted ${}_nP_k$, and is given by the formula $\mathbf{{}_nP_k = n!/(n-k)!}$ For example, ${}_5P_2 = 20$, which means that there are 20 different ways that you could form an ordered list of 2 of the 5 letters A,B,C,D,E. Can you list them all?

· The number of **combinations of n objects taken k at a time** is denoted ${}_nC_k$, or C_n^k, and is given by the formula $\mathbf{{}_nC_k = n!/k!(n-k)!}$ For example, ${}_5C_2 = 10$, which means that there are 10 different ways that you could form an unordered list of 2 of the 5 letters A,B,C,D,E. Can you list them all?

· In certain probability questions, when using the formula P(E)=n(E)/n(S), the sample space and favorable events are way too big to list. In these problems, it may still be a good idea to describe the sample space in words and begin to list the outcomes in it, as starting to list outcomes in the sample space may help you figure out how to use the combinations/permutation formulas above to calculate n(E) and n(S).

· An undirected **graph** G consists of a collection of **vertices** and of segments or arcs starting and ending at vertices which are called **edges**. An edge may connect two different edges or it may start and end at the same vertex; in this last case it is called a **loop**. Here is the graph G modeling Euler's problem whether or not it is possible to stroll around Königsberg in Prussia (nowadays Kaliningrad in Russia) crossing each of its bridges across the Pregel (nowadays called Pregolya) exactly once:



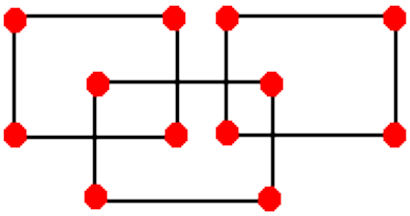
The above graph has four vertices (A,B,C,D). Vertices A and B are joined by two edges, vertices A and C are also joined by two edges, while A and D, B and D, C and D are each joined by only one edge. There are altogether 7 edges, each corresponding to a bridge over the Pregel.

· The **degree** or **valency** of a vertex is the number of edges that end (or start) at that vertex (a loop at that vertex is counted twice). For instance the degree/valency of the vertex A in Euler's above graph is 5. The degree of vertex D is 3. A vertex whose degree is even is called **even**; a vertex whose degree is odd is called **odd**. For instance all vertices in Euler's graph are odd. Check this!

· Every graph has an even number of odd vertices!

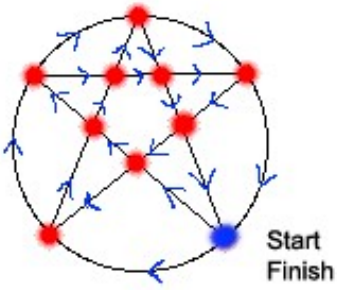
· A **path** in a graph G is a finite sequence of edges in which any two consecutive edges are adjacent. For instance a path of length four:

We say a path is a **circuit** if the initial vertex is also the finial vertex. A graph is **connected** if and only if there is a path between each pair of vertices. It is called **disconnected** otherwise. Is the Königsberg's bridges graph connected or not? The following graph is disconnected (it has three components):



· A path that runs exactly once over each edge of a graph is called an **Eulerian path**. An Eulerian path which is a circuit (that is it is closed) is called an **Eulerian circuit**. A graph that possesses an Eulerian circuit is called an **Eulerian graph** (named in honor of Euler's problem about Königsberg's bridges, which required to check if the bridges' graph is Eulerian or not).

· **Euler's theorem**: A graph has an Eulerian circuit if and only if it is connected and all its vertices are even. A graph has an Eulerian path if and only if it is connected and has either no odd vertices or exactly two odd vertices. If two of its vertices are odd, then any Eulerian path must begin at one these vertices and end at the other one. The Königsberg's bridges graph is not Eulerian since it has four odd vertices! Here is an example of an Eulerian circuit:



· A **bridge** in a graph is an edge whose removal disconnects the remaining graph. Eulerian graphs have no bridges! An **isolated vertex** in a graph is a a vertex with no edges connected to it.

· If G is an Eulerian (connected) graph, then the following algorithm (**Fleury's algorithm**) always produces an Eulerian circuit in G:
Start at any vertex of the graph and walk along the edges of the graph arbitrarily but subject to the following two rules:

1) Erase edges as they are traversed, and the isolated vertices resulted (if any)

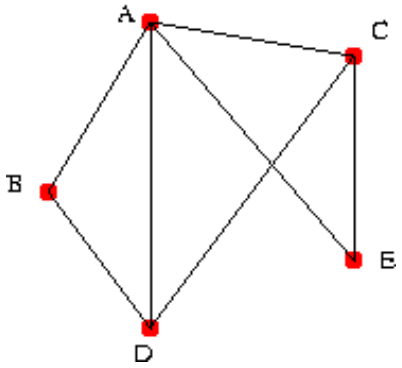
2) Walk along a bridge only if there is no alternative.

· If a graph does not have an Eulerian circuit or an Eulerian path, we might still be interested in knowing how it could be traveled with as few retraced edges as possible (starting and ending at the same vertex if interested in a circuit, or starting and ending at different vertices if just interested in a path). This process is called an **Eulerization** of the graph. Here are the guidelines mentioned in your textbook to achieve that:

− Circle all odd vertices and pair each of them with another odd vertex that is close to it

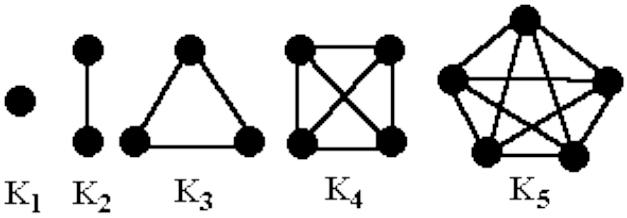
− For each pair of odd vertices, find a path with the fewest edges in the original graph connecting them and then duplicate all edges along this path.

· A connected graph G is said to be **Hamiltonian** if there exists a circuit (called a **Hamiltonian circuit**) passing through each vertex of G. Hamiltonian circuits, like Eulerian circuits, aim to traverse the "whole graph", but by whole they mean "visit each vertex once" (as opposed to Euler's "walk each edge once"). Here is one example of a Hamiltonian graph:

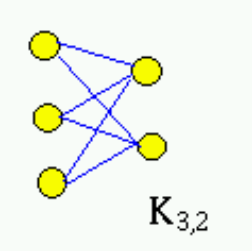


Can you find a Hamiltonian circuit in it ?

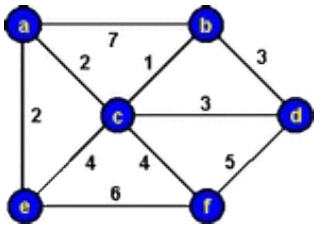
· A graph is called **simple** if there is at most one edge between any two of its vertices. A graph is called **complete** if there is exactly one edge between any two vertices. The complete graph with n vertices is denoted K_n. Here are the first five complete graphs:



A graph is called **complete bipartite**, and denoted $K_{n,m}$, if its vertices can be partitioned into two disjoint subsets A and B, one of size n and the other of size m, such that every vertex in A is connected to every vertex in B, and these form all edges of the graph. For instance



· A **weighted graph** is a graph for which one has associated a number (the **weight**) to each edge. For instance:



· The **Traveling salesman problem** is to find a Hamiltonian circuit in a complete weighted graph for which the sum of the weights of the edges is minimal. The sum of the weights of the edges of a circuit is called the **weight of the circuit**. More generally the Traveling salesman problem asks to find in a weighted (simple) graph a Hamiltonian circuit of least total weight.

· A **Brute Force Algorithm** approach to the Traveling salesman problem is: Check all circuits and select the one with the least total weight! In other words:

− Make a list of all possible Hamiltonian circuits of the graph

− For each Hamiltonian circuit, calculate its total weight by adding the weights of all the edges in the circuit.

− Find the circuit(s) with the least total weight.

Pros: Guaranteed to find the most (cost/weight)−efficient circuit! Cons: Enormous amount of work to carry out the algorithm!
(each increase in the number of vertices increases the work by a factor equal to the number of vertices in the graph)

· The **Nearest Neighbor Algorithm** is an approximate algorithm for the Traveling salesman problem in a complete weighted graph:

− Start at the vertex where the circuit is supposed to begin

− From the starting vertex, go to the vertex for which the corresponding edge has the smallest weight (= **nearest neighbor**).

− Build the circuit, one vertex at a time, by always going from a vertex to the nearest neighbor of that vertex from among the vertices that haven't been visited yet. If there is a tie choose any of the nearest vertices. Keep doing this until all the vertices have been visited.

− Once all the vertices have been visited, from the last vertex, return to the start.

Pros: Much less work then the brute force approach! Cons: It doesn't necessarily produce the optimal Hamilton circuit (so it is a compromise answer).