Complex Systems 535/Physics 508: Homework 5

1. **Generating a random permutation:** Assume that you know how to generate random integers uniformly in the range $0 \ldots k-1$. Assume also that you have an array of n elements that currently stores n integers in ascending order.

   (i) Describe an algorithm that will generate a random permutation of the integers in the array in time $O(n)$, assuming it takes $O(1)$ time to generate a random number. You algorithm should generate the permutation *in place*, that is, it should do it by rearranging items in the array, not by making use of another array. (You may however make use of a small number (order a constant) of other "registers" to store integers temporarily, if you wish.)

   (ii) Prove that your algorithm does indeed generate all permutations with equal probability.

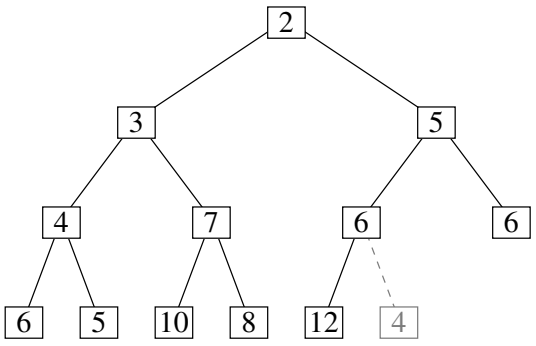
2. **Choosing a vertex in proportion to its degree:** Later in the course we will study "preferential attachment" models of growing graphs. The archetypal such model, proposed by Price in 1976, is a model of a citation network in which newly written papers cite equal numbers of pre-existing papers with a probability proportional to those pre-existing papers' in-degrees. That is, the probability that your paper gets cited by a new one, is proportional to the number $k^{(\text{in})}$ of citations you already have. (This is not a complete description of Price's model—actually in his model it was proportional to $k^{(\text{in})} + c$, where c is a constant.)

   (i) Assume you have a way to generate random numbers r that fall uniformly in the range $0 \leq r < 1$. Devise and describe a computer algorithm making use of these numbers to choose vertices in proportion to their in-degrees. You can assume you already have the in-degrees. You need not give pseudocode (or code) unless you want to—a description in clear English is fine.

   (ii) Assuming the generation of random numbers takes negligible time, what is the leading-order time complexity of your method? That is, how long, in terms of the numbers m and n of edges and vertices, will it take to choose a vertex?

   (iii) **Extra credit:** Probably you will find your method takes time $O(n)$ to make a choice. Extra credit goes to whoever can find a method that does the same thing in time $O(m/n)$. You can assume you have the complete graph in adjacency list form. (If, by skill or chance, you found a method that does $O(1)$ in part (ii), then you automatically get the extra points.)

3. **The binary heap:** In class we studied Dijkstra's algorithm, and mentioned that efficient implementations make use of the binary heap. A heap is a partially ordered binary tree in which numbers are stored at the vertices of the tree in such a way that all numbers are greater than their "parent" in the tree:



(Ignore for the moment the extra item at the bottom with the dashed line, which is obviously not greater than its parent.)

(i) If there are n numbers in the tree, what is the depth of the tree (i.e., the number of levels)?

(ii) The process of adding an item to the heap involves "sifting" the elements. First, you add the item at the bottom of the heap, as with the "4" at the bottom of the picture. Then you compare it to its parent, and if it is smaller, you exchange parent and child, and you do that repeatedly until the new item finds a level at which it is, correctly, greater than its parent. Draw a picture of the heap above after the item "4" has been sifted up the tree.

(iii) What is the time complexity in terms of n for adding an item to a binary heap and why?

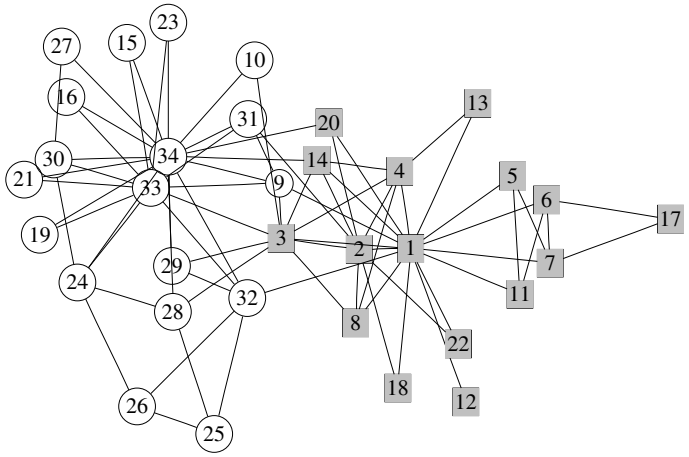
(iv) Removing an item is similar. You remove the item in question leaving a hole in the tree. You fill the hole by looking at the two "children" of the hole and moving the smaller of the two into the hole. This of course leaves another hole, so you repeat the process, until the hole falls off the bottom of the tree. Draw the heap above after the item "3" has been removed from it and the tree sifted to get rid of the hole.

(v) What is the time complexity for removing an item from a heap and why?

(vi) What is the time complexity for finding the smallest item in a heap and why?

(vii) If we use a binary heap to store the estimated distances of vertices from the source in Dijkstra's algorithm, what will be the overall time complexity of the algorithm in terms of the number of vertices n and the number of edges m in the graph?

4. **Spectral bisection:** Here is a famous graph from the social networks literature:



It represents friendships between students at a karate club at a US university (W. W. Zachary, An information flow model for conflict and fission in small groups. *Journal of Anthropological Research* **33**, 452–473 (1977)). You can download the adjacency matrix for this network from here:

`http://www-personal.umich.edu/~mejn/courses/2004/cscs535/karate.adj`

(i) Carry out a spectral bisection of this network. Give the algebraic connectivity of the graph and the corresponding eigenvector. Thus determine the bisection of the graph.

(ii) During the course of Zachary's study of the karate club, a dispute arose between the members and the club eventually split in two. The two factions are shown by the squares and circles in the figure above. How well do these correspond to the results of your bisection?