CS 341, Fall 1999 due Friday Nov. 19

A. Lubiw, D. Schuurmans

ASSIGNMENT 4

Do not copy from others, and acknowledge your sources.

1. [10 points] Give a backtracking algorithm to compute the permanent of an $n \times n$ matrix $M = [m_{i,j}]$. The *permanent* of such a matrix is the sum over all permutations $a_1, \ldots, a_n$ of $1, \ldots, n$ of $\Pi_i m_{i,a_i}$, that is $\sum_{(a_1,\ldots,a_n) \in perms(1,\ldots,n)} \prod_i m_{i,a_i}$. [It is a curious fact that the closely related *determinant* can be computed efficiently, though no one knows how to compute the permanent efficiently.]

2. [10 points] Consider the problem of finding the length of the shortest path between all pairs of vertices in a directed graph. Let problem A be the version of this problem where we are given weights on the edges of the graph, and the length of a path is the sum of the weights of the edges in the path. We saw a dynamic programming algorithm for version A. Let problem B be the variant where we are given weights on the *vertices* of the graph, and the length of a path is the sum of the weights of the vertices in the path. Give an algorithm to solve problem B by giving a reduction from problem B to problem A. Analyze the worst case running time of your algorithm.

3. [10 points] Give an algorithm for finding the max and min of an unsorted array $A[1...n]$ that uses at most $3n/2$ comparisons. [Note that finding the max and min separately uses $n-1+n-2$ comparisons.]

4. [10 points] Prove via an adversary argument that any algorithm which finds the min and max of an unsorted array $A[1...n]$ must make at least $3n/2-2$ comparisons in the worst case (assuming that the algorithm can only operate on array elements by making pairwise comparisons). [Hint: For an algorithm to know that an element x is the max and the element y is the min, it must know that every element other than x has lost some comparison and every element other than y has won some comparison.]

5. PROGRAMMING. [30 points]
   (A) Program a backtracking algorithm for the following problem. Input is an undirected graph $G = (V, E)$, weights $w : E \rightarrow$ R, and specified vertices s and t. Output is the minimum weight of a simple path from s to t, and a path realizing this minimum. Note that the graph is permitted to have negative weight cycles. Recall that a path is *simple* if it never repeats vertices. Analyze the worst case running time of your algorithm.
   (B) Making use of your program from part (A), write a program for the following problem. Input is an undirected graph and specified vertices s and t. Output is a *Hamiltonian path* from vertex s to vertex t, if one exists. A *Hamiltonian path* is a simple path that includes all vertices. You should not rewrite any of your code from part (A), just use it

as a subroutine. In other words, you will be reducing this problem to the problem in part (A).

6. BONUS QUESTION. [5 points] Suppose you are given two sorted lists of length n. Give a comparison based algorithm to find the median of the $2n$ elements. Analyze the number of comparisons used, and give a matching lower bound. The analysis and the lower bound should be exact; do not use O or Ω.