ECE-250 – Algorithms and Data Structures (Winter 2012)

Practice Problems for the Final

Disclaimer: Please do keep in mind that this problem set does not reflect the exact topics or the fractions of each of the topics or the amount of work that will appear in the final. It is supposed to help you prepare for the final, but I'm including here mostly the second part (after the midterm). For the first part, you have assignments, practice problems, and you have the midterm with posted solutions to review the material.

Disclaimer 2: I may or may not post solutions.

Sorting algorithms

1 – Using the median-of-three with first, last, and middle values, run the first round (the first instance of the recursive procedure) of quick sort on the following sequence of values: $10, 28, 15, 31, 47, 91, 5, 37, 23$

2 – Suppose we're using quick sort to process data that we're receiving from a connection in a networked system. We want to cover our system from the possibility of being "sabotaged" by hostile connections — we could receive data that is specifically crafted to cause quick sort to have its worst-case performance and thus make our system consume excessive resources and time (rendering it unable to efficiently respond to other connections).

(a) Assuming that quick sort simply chooses the first element as the pivot (instead of the median of first, last, and middle), what is the arrangement of data that produces the worst-case performance in quick sort? (that is, if you were the attacker trying to sabotage the system, what data would you have to send?)

(b) Suggest a simple strategy (hopefully requiring no more than linear time) to avoid the problem. That is, a strategy to guarantee that quick sort will run in $O(n \log n)$ most of the time, regardless of input data, even if this input data is maliciously created. Notice that in the context of this question, you're not allowed to change the way quick sort selects the pivot (in fact, you will hopefully suggest a strategy that works regardless of how quick sort selects the pivot).

Priority queues and Heaps

3 – Insert the following values into an initially empty min heap, ensuring that we always maintain a complete binary tree: $15, 17, 9, 11, 5, 19, 7$

4 – From the resulting min heap, remove all the elements, in order, ensuring that at all times (that is, once each removal is completed) we have a complete binary tree.

5 – Suppose that we do a depth-first traversal in a max heap, stopping the recursion when we either

reach a leaf node or when we reach a node with a negative value.

Prove that the run time is $O(p)$, where p is the number of positive values in the max heap.

Graphs

6 – Prove that a Directed Acyclic Graph (DAG) must have at least one vertex with in-degree 0.

7 – Given a DAG where vertex v_j is adjacent to vertex v_i if v_j is divisible by v_i, and vertices 3 to 15:

(a) Without running a topological sort, indicate what could be a possible outcome of a topological sort.

(b) Run a topological sort on this graph, with the additional constraint that whenever more than one vertex can be output next, you must choose the highest value among the candidate vertices.

8 – Why is Dijkstra's algorithm a greedy algorithm? Does it have characteristics that fall in some other algorithmic paradigm? If so, why?

9 – Suppose a DAG is implemented with a class template with the following declaration:

```
class Dag
{
public:
    Dag();
    void insert (int u, int v);   // inserts an edge from u to v
    bool adjacent (int u, int v); // checks if v is adjacent to u
    bool connected (int u, int v); // checks if there is a path from u to v
};
```

Assuming that we're using an adjacency matrix, show an implementation (it can be in pseudo-code — but at least make it C++-like pseudo-code) of the insert method, ensuring that no cycles are created.

10 – For the graph from question 1 in assignment 5, assuming that the weights are given by the two-digit value obtained placing the source vertex and then the target vertex (for example, the edge from 2 to 3 has weight 23, the edge from 1 to 4 has weight 14), run Dijkstra's algorithm to find the shortest path from 1 to 8.

11 – For the same graph as the previous question, assuming it is undirected:

(a) Use Prim's algorithm to find a minimum spanning tree.

(b) Use Kruskal's algorithm to find a minimum spanning tree.

Algorithm design techniques

12 – (a) Show that a naive recursive implementation of a function to obtain the n-th Fibonacci number (as $\text{fib}(n) = \text{fib}(n-1) + \text{fib}(n-2)$ for $n > 2$, 1 otherwise) has non-polynomial run time.

(b) Write a C++ function that uses dynamic programming to make this run time linear (with respect to the value of n), using the top-to-bottom approach.

(c) Write a C++ function that uses the bottom-to-top approach and achieves linear run time.

13 – Write a C++ function that solves the eight queens puzzle in a reasonably efficient way (i.e., using backtracking). You can assume that there is a function that checks whether the current configuration is valid (i.e., if in the given configuration, no queen is under threat of capture) — this function could have prototype `bool valid ( ··· )`, where the parameters are whatever you decide works well with your approach.

NP-completeness

14 – Explain why the only reductions that we are interested in when dealing with NP-completeness are polynomial time reductions.

15 – Show that the computational version of subset-sum reduces in polynomial time to the decision subset-sum problem. That is, show that obtaining a subset that adds to the given parameter can be implemented in terms of an algorithm that simply determines whether such a subset exists, with a polynomial time reduction.