General Examination in Theory and Algorithms - Spring, 1999

Answer one question from Part I (consisting of two questions) of the test, two questions from Part II (consisting of four questions) of the test, and one additional question from either part.

Part I

1.

a) Draw the finite state machine that would accept the language ba*b+. Give a grammar for this language. Is this language regular, context-free, context-sensitive, or what?

b) Give a regular expression for a language with alphabet {x,y} that consists of all strings with at least two consecutive y's.

c) Consider the grammar below. Is it regular, context-free, context-sensitive, or what? How many terminal strings of length 5 can be generated by this grammar?

```
S --> A 0 B
A --> B B | 0
B --> A A | 1
```

2.

a) Define the pumping lemma for a context-free language. Define a Turing machine and a non-deterministic Turing machine.

b) In some computer languages (e.g., PL/1), there are exception handling mechanisms that allow a user to define what would happen if a given exception (e.g., floating point overflow, division by zero, end of file, or data conversion error) occurs. Would it make sense to have a language allow users to write code to define what would happen if an infinite loop was entered? If so, what would the handling mechanism do? If not, why not?

c) Define what an ambiguous grammar is and why it is a problem.

d) Describe the language generated by the following grammar:

```
S --> a S b | b S a | SS | NULL
```

Part II

3. There are four parts to this problem.

(a). Suppose `nums[1..N]` is an array of N numbers. Find an efficient algorithm to compute max(i, j) = max{`nums[i]`, `nums[i+1]`, ⋯, `nums[j]`} for all $1 \le i \le j \le N$. Give the complexity of your algorithm using the most appropriate notation among $\Theta(\cdot)$, $O(\cdot)$, and $\Omega(\cdot)$. [50]

(b) Give a clear argument to indicate if your algorithm is optimal or not. [15]

(c) Suppose N is large and hence we *do not* want to store max(i, j) for all $1 \le i \le j \le N$. Instead, we want to compute max(i, j) afresh everytime we need it. Give a brief idea how to preprocess the data so that this can be done in $O(log\ N)$ time for each $i \le j$. Expain your idea for computing max(2,6) using the following data. [15]

nums[ ] = [2.1, −3.2, 4.0, 5.2, 3.8, 2.5, 8.1 −2.0].

(d) Suppose an algorithm A for graphs with m nodes and n edges, where $0 \le n \le m(m-1)/2$, has the complexity $O(m+n)$. Is it true that we can write the complexity of A as $O(\max\{m, n\})$? Explain your answer clearly. [20]

4. Formulate the following problem as a shortest path problem or a longest path problem such that a shortest path (or the longest path, as the case may be) in the appropriate digraph will give the solution to the original problem. Your ideas must work for the general case of any set of intervals. Show the digraph, the costs of its arcs, and an optimal path for the specific data given below. What algorithm can you use to solve this problem? (You do not need to give any algorithm.) [80+10+10]

You are given a set of intervals which represent the beginning and ending of certain events. For example, you may have I1 = [1, 5], I2 = [2, 7], I3 = [4, 10], I4 = [8, 18], I5 = [8, 11], I6 = [14, 20], and I7 = [15, 19]. You want to find a subset of these events such that one can attend to maximize the time spent in the events. (You cannot attend two events if they have an intersection other than just at the end points.)

Question 1

(a) Draw the decision tree for the binary search algorithm acting on a list of eight elements

(b) Write and then solve a recurrence relation to get the worst-case behavior for binary search when n is a power of 2. Apply this solution for the above problem of a list of eight elements.

(c) State a theorem on the lower bound for searching problem (for an "n" element list).

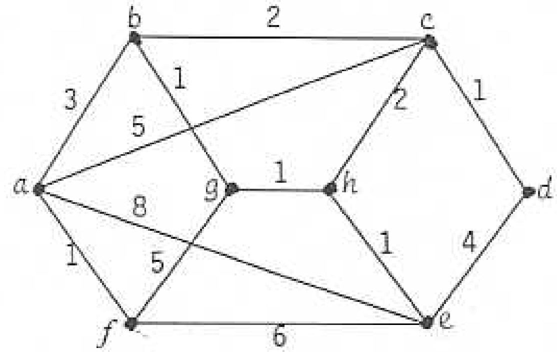
Question 2

Assume that we have a simple, weighted, connected graph, where the weights are positive. Then a path exists between any two nodes x and y. Indeed, there may be many paths. The question is how do we find a path of minimum weight.

(a) Design a Dijkstra's algorithm for the pairs of nodes given in the following graph. Write out the nodes in the shortest path and its distance.

(a_{11}) From b to e

(a_{12}) From c to f



(b) What is the complexity of this algorithm?