Sample Final Exam 1

Instructions:

Congratulations! You've made it through a tough quarter and there's but one final task awaiting you. This is an open book, open notes exam.

Please write clearly and succinctly;

1 Algorithm Usage [16 points]

For each of the following settings specify which of the algorithms we have studied you would use to solve it. You should use the most efficient algorithm (considering both time and space), and you should choose an algorithm which needs minor modifications to work. Briefly justify your choice, indicate how the problem would be solved, and give the running time of your algorithm.

1.1. a) (11 points)

You are given a program which has $p$ procedures $P_1, P_2, \ldots, P_p$. For each procedure $P_i$ you have a list of the procedures which can be called from it.

i) (6 points) Assume the program always starts executing in $P_1$.

You want to know for each procedure $P_i$: A) whether it can ever be called (i.e. is there some sequence of calls starting with $P_1$ which reaches $P_i$), and B) if it can be called, what is the shortest calling sequence (the one with the fewest calls ending at $i$).

Note that for a procedure $P_i$ we want not only the length of the shortest calling sequence, but the actual procedures in that sequence.

ii) (5 points)

Now suppose that we want to do an initial computation so that when we are done we can quickly answer the following type of question: given $i, j$, if the program starts in procedure $P_i$ can we reach $P_j$? We don't care if the initial computation is fairly slow.

b) (5 points) You want to build a computer network which connects $n$ sites. For each pair of sites you have a known cost $c(i, j)$ for building a two-way communication line between $i, j$ (so $i$ can send messages to $j$ and $j$ can send messages to $i$). You want to choose a set of lines which allows any pair of computers to communicate (either directly or indirectly: if there are lines $i, j$ and $j, k$ then $i$ can send a message to $k$). Your goal is to minimize the total cost of the lines built.
2 Short answers

2.1. (5 points) The subset sum problem is defined as follows: given a list of numbers $x_1, \ldots, x_n$, and a target $B$, can you find a set of items whose total value is exactly $B$?

Show that the subset sum problem is in $NP$. NOTE: you are NOT being asked to show it is $NP$-complete.

2.2. (8 points) For which of the following settings is the all-pairs shortest path problem solvable in polynomial time (you should assume that P is not equal to NP in your answer, and that we always want the best simple path)? Briefly justify your answer

(a) Arbitrary graph, one edge has length -5 the rest have length 1.
(b) Arbitrary graph, but all edges have length -1.
(c) Arbitrary edge weights, but graph is acyclic.

2.3. (25 points) For each of the following problems, write true or false. Briefly justify your answer. For the purpose of answering this question you should assume that $P$ is not equal to $NP$.

(a) If $A \leq_p B$, $B \leq_p C$ and $A$ is NP-hard then $C$ must be NP-hard also.
(b) The traveling salesman problem (TSP) is hard even if the graph is complete (there is an arc between every pair of vertices) and all arcs have weight 1.
(c) Prim’s MST algorithm works correctly even on a graph with negative cycles.
(d) For a directed graph $G$ with $m$ edges and $n$ nodes, stored in adjacency list form, we can check if $G$ has a cycle in $\Theta(m + n)$ time.
(e) In the union-find algorithm using balancing and path-compression we can do a union of the set containing $i$ and the set containing $j$ in ($\Theta(\log n)$ time even if $i$ and $j$ are not set names.

3 Data Compression

$S = DEBCAAACABBCAACCAACCCACACCCAA$

(a) Give a huffman code for the string $S$ (describe both your tree and the actual binary encoding of the first five characters of the above string).
(b) Give the "high-level" Lempel-Ziv encoding of the above string where each match of length 3 or longer is represented by a length, pointer pair, and otherwise by the actual input character. Assume that your window goes all the way back to the start of the string.

4 String Matching

For the Pattern BBBABBBBD
and an alphabet $= \{A, B, C, D\}$

Draw the finite state machine (FSM) which allows you to search for this pattern. Show your steps in constructing the FSM.
5 Aproximate String Matching (NOT covered)  [10 points]

Suppose we have a new cost measure for errors when comparing a pattern to a text. Our user has a bad keyboard which sometimes doesn’t register a character typed (so you might type an ‘x’ and have nothing appear). Thus we change our cost measure:

- matching a text character to nothing costs 1.
- a mismatch costs 3,

matching a pattern, character to nothing costs 3.

For example, I could match \( P = \text{srt} \) to \( T = \ldots \text{smart} \) by pairing t-t (cost 0), r-r (cost 0) nothing-a (cost 1) nothing-m (cost 1) and s-s (cost 0) for a total cost of 2. But pairing s - a would cost 3 which is worse.

We want to modify the dynamic programming algorithm to find \( T[i,j] \) the least cost way to match the first \( i \) characters of a pattern \( P \) with the text ending at (and using) \( t_j \).

(a) Describe how to compute \( T[1,1] \).

(b) Describe how to compute \( T[1,j] \), for \( j > 1 \).

6 Shortest Paths  [12 points]

Consider the airline problem we looked at on the problem set: given a list of flights out of each city, find the fastest route from Sacramento to Shangrila. However, we now assume that if for each city \( i \) there is a transfer delay \( t_i \). Thus we need at last \( t_i \) minutes between an arriving flight and a departing flight at city \( i \) (note that the \( t_i \) values are different for different cities). For example, if we arrive at 2PM into city \( i \) and \( t_i = 10 \), we can only take departing flights at 2 : 10 or later.

(a) Describe how to find the fastest time to reach Shangrila. In particular, focus on any changes from the solution without transfer delays.

(b) What is the running time of your algorithm if there are a total of \( c \) cities and \( a \) flights?

(c) How would you modify your algorithm to find the actual route (that is the actual sequence of flights to take)? Does this change the run time?