COT 5405 Analysis of Algorithms, Spring 2010.
Midterm 3

Name: ____________________

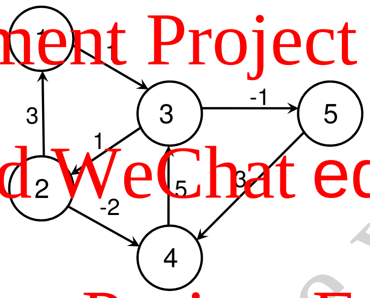
UFID: ____________–____________

Notes

- This is a closed-book exam. No calculator.
- You have 100 minutes for the exam.
- If the problem necessitates writing an algorithm, you must first informally describe the algorithm, in brief, in a paragraph. Yo algorithm. We will per
- Write your name on the last word in your name.
- If you are designing an algorithm, you must write a formal proof of corr
- Please write legibly.

1. [1 page][33points] The following graph with weights on edges is given. Apply Floyd-Warshall algorithm to compute shortest path between all pairs of vertices (only the distance of the path, not the path itself). Provide a weight matrix at each iteration. Floyd-Warshall recurrence relation is as below.

Let $D^{(k)} = (d_{i,j}^{(k)})$ be the weight matrix after kth iteration,
$d_{i,j}^{(k)} = min(d_{i,j}^{(k-1)}, d_{i,k}^{(k-1)} + d_{k,j}^{(k-1)}),\ k \geq 1$
$0 \qquad if\ i = j$
$d_{i,j}^{(0)} \qquad /$



Solution:

$$D = \begin{array}{lllll} 4 & 1 & 0 & & \\ 9 & 6 & 5 & 0 & 4 \\ 12 & 9 & 8 & 3 & 0 \end{array}$$

2. [1 page][33points] You need to go over a river by canoe and there are n trading posts along the river. We use $f_{i,j}$ to denote the fee from post i to post j. These fees are arbitrary. For example it is possible that $f_{1,3} = 10$ and $f_{1,4} = 5$. You begin at trading post 1 and must end at trading post n (using rented canoes). Your goal is to minimize the rental cost. Design an efficient algorithm for this problem. Be sure to prove that your algorithm yields an optimal solution and analyze the time complexity.

Solution:
Let m[i] be the rental cost for the be $\leq i \leq n$. The final answer is in m[1]. We can recurs

$$m[i] = min_{i \leq j < n}(f_{i,j} + m[j]) \text{ otherwise}$$

We now prove this is correct. The canoe must be rented starting at post i (the starting location) and then returned next at a station among i + 1, . . . , n. In the recurrence we try all possibilities (with j being the station where the canoe is next returned). Furthermore, since dent from how the subproblem of going from post j, . . . , n is solved, we have the optimal substructure p For the time complexity there are n subproblems to be solved each of which tak subproblems can be computed in the order m[n],m[n-1], . . . ,m[1]. Hence the ov is O(n^2).

3. [2 page][(8 + 3) + (8 + 3) + (8 + 3) + 1 = 34 points] **[(8 + 3) + (8 + 3) + (8 + 3) + 1 = 34 points]** You are given coins of n different integer denominations $x_1 < x_2 < ... < x_n$. Suppose you are asked to determine whether coins of these denominations can be used to make change for an amount of exactly V (where V is an integer). Consider the following different conditions: (A) You are allowed to use at most one coin for each denomination, (B) You are allowed to use as many coins as required for each denomination, (C) You are allowed to use as many coins as required for each denomination but you are allowed to use at the most K coins in total.
Note: When we say 'as many c s are used.
For **each** condition, you , explain the recurrence clearl algorithm. Which of your algorithms (if any) qualifies as being a polynomial time algorithm? Why or why not?
Examples: Let $x_1 = 9$, $x_2 = 11$ be the two denominations available to you. (Part A) You can create change for $V = 20$ but not for $V = 29$ as the latter requires two coins of value 9. (Part B) You can create change for $V = 29$ but not for $V = 12$ as the latter amou integer combination of 9 and 11 even if you had an unlimited supply of coins of both v C) You can create change for $V = 29$ using $K = 3$ coins, but not fo ires 4 coins.

Solution:
Part (A): You are allowed to use at most one coin for each denomination.
Recurrence: $M(n, V) = M(n-1, V - x_n)|M(n-1, V)$ where $|$ refers to a logical OR operator.
Let $M(i, v) = 1$ if you can make change for an amount of v using denominations $x_1, x_2, ..., x_i$ else it is 0. The logic for this recurrence is this: either you can use a coin of value x_n for making the change, or you don't. In the former c amount $v - x_n$ usi $-1, V - x_n)$). In the latter case, our algorithm sl v using the first $n-1$ denominations (hen ur algorithm outputs 1 and hence the logical OR operator in the recurrence. $O(nV)$ as you have to build a table of n rows and V columns.

Part (B): You are allowed to use as many coins of a given denomination
Recurrence: $M(V) = \max_{\{i: x_i \leq V\}} M(V - x_i)$
(another way of writing: $M(V) = \delta(x_1 \leq V)M(V - x_1)|\delta(x_2 \leq V)M(V - x_2)|...|\delta(x_n \leq V)M(V - x_n)$).
where $\delta(x_i \leq V) = 1$ if $x_i \leq V$ else $\delta(x_i \leq V) = 0$. Here $M(V) = 1$ if you can use coins of these values to make change for V, else it is 0. **Complexity:** $O(nV)$ as you have to build a 1D array of V entries but the initial recurrence requires you to examine n possibilities.

Part (C): You are allowed to use as many coins of a given denomination as required but you can use at the most K coins.
Recurrence: If $k > 1$, $M(k, V) = \max_{\{i: x_i \leq V\}} M(k-1, V - x_i)$ else $M(k, V) = (x_1 == V)|(x_2 == V)|...|(x_n == V)$
where $M(k, v) = 1$ if you can make change of value v using k coins at the most. **Complexity:** $O(nVK)$ as you have to build a table of K rows and V columns, but the initial recurrence requires you to examine n possibilities.

All three algorithms are pseudo-polynomial (not polynomial) because their running time is directly proportional to the value V which requires $O(log_2 V)$ bits of storage. The value is always exponential in the number of bits for storage. Hence, all three algorithms are exponential time.