CS570 Spring2024: Analysis of Algorithms Exam II

	Points		Points
Problem 1	20	Problem 5	18
Problem 2	9	Problem 6	20
Problem 3	16	Problem 7	17
	Total	**100**	

Instructions:

1. This is a 2-hr exam. Closed book. No electronic devices or internet access. One 8.5x11 cheat sheet allowed.
2. If a description to an algorithm or a proof is required, please limit your description or proof to within 150 words, preferably not exceeding the space allotted for that question.
3. No space other than the pages in the exam booklet will be scanned for grading.
4. If you require an additional page for a question, you can use the extra page provided within this booklet. However please indicate clearly that you are continuing the solution on the additional page.
5. Do not detach any sheets from the booklet. Detached sheets will not be scanned.
6. If using a pencil to write the answers, make sure you apply enough pressure, so your answers are readable in the scanned copy of your exam.
7. Do not write your answers in cursive scripts.
8. This exam is printed double sided. Check and use the back of each page.

1) 20 pts

Mark the following statements as **TRUE** or **FALSE** by circling the correct answer. No need to provide any justification.

[TRUE/FALSE]

In a flow network, the maximum flow is unique if all edge capacities are unique AND the min cut is unique.

[TRUE/FALSE]

In a dynamic programming solution, the value of the optimal solution should always be saved for all unique sub problems.

[TRUE/FALSE]

The dynamic programming algorithm for the Knapsack Problem runs in weakly polynomial time in the size of the input.

[TRUE/FALSE]

Given a flow f in a flow network G, if there exists a directed path from the source node s to the sink node t in the residual graph G_f, then f is not a max-flow in G.

[TRUE/FALSE]

Let (A, B) be a minimum *s-t* cut in a flow network. If we strictly increase the capacity of every edge that crosses the (A,B) cut, then the value of a maximum flow of the network must strictly increase.

[TRUE/FALSE]

During the execution of the Ford-Fulkerson algorithm, if all augmenting paths are simple (acyclic), then the resulting max flow will have no flow cycles.

[TRUE/FALSE]

If an iteration of the Ford-Fulkerson algorithm on a flow network assigns a flow of 1 to an edge (u, v), then after every subsequent iteration, the flow assigned to (u, v) must have a value greater than or equal to 1.

[TRUE/FALSE]

It is possible for a correctly implemented dynamic programming algorithm to have an exponential running time with respect to the input size.

[TRUE/FALSE]

Any dynamic programming solution with n^2 unique subproblems will run in $O(n^2)$ time.

[TRUE/FALSE]

It is possible for a circulation network to not have a feasible circulation even if all edges have unlimited capacities.

2) 9 pts (3 pts each)
Circle ALL correct answers (no partial credit when missing some of the correct answers). No need to provide any justification.

i- Given a flow network G with n nodes and m edges, with a maximum flow of value k, which of the following statements are true for the running time of some flow algorithm:

a) A running time of $O(mnk)$ is weakly polynomial

b) A running time of $O(m \log k)$ is pseudo-polynomial

c) A running time of $O(n\, m^2)$ is strongly polynomial

d) A running time of $O(k^2)$ is pseudo-polynomial

ii- Which of the following holds for the scaled version of the Ford-Fulkerson algorithm? Assume C is the maximum capacity of any edge outgoing from the source, and m is the number of edges in the network.

a) The number of Δ-scaling phases is $O(\log C)$

b) The number of Δ-scaling phases is $O(C)$

c) The number of augmentations in a Δ-scaling phase is at most m

d) The number of augmentations in a Δ-scaling phase is at most $2m$

iii- Bellman-Ford algorithm can be used to find.

a) Flow cycles in a flow network

b) Flow cycles in a circulation network

c) Negative cycles in a directed graph

d) Shortest paths in a directed graph

3) 16 pts

The Miso Good food truck produces many different lunch menu items. Suppose that, on a given day, Miso Good has produced m types of food items $b_1, \ldots, b_m$, and the quantity of each type of food item b_j is exactly q_j. Unfortunately, due to the limited quantities, they often run out of popular items, making customers sad. As a solution, Miso Good is implementing a sophisticated lunch-ordering system as follows. Suppose there are n customers $a_1, \ldots, a_n$. The customers are required to text in their acceptable choices for food items before lunchtime - each customer a_i submits a set A_i of one or more acceptable food items. Then, Miso Good will try to assign ONE food item to each customer, and the customers who cannot be assigned a food item from their acceptable set, will be given a \$10 voucher as compensation for their inconvenience. Miso Good would like to minimize the number of these vouchers they give out.

a- Given the input as described above, design an efficient network flow based algorithm for Miso Good to output the optimal assignment of lunches to customers (and vouchers as necessary). (10 pts)

See next page for part b

b- Prove the correctness of your algorithm. (6 pts)

4) 20 pts

Given an array A of length n containing positive integers $a_1, \ldots, a_n$ and positive integers m and k such that $mk \leq n$, Your goal is to choose k mutually disjoint subarrays of length m from array A to maximize the total sum of elements within these subarrays. Formally, let the start and end indices of the k intervals of length m be $[L_1, R_1]$, $[L_2, R_2]$, …, $[L_k, R_k]$ such that $1 \leq L_1 < R_1 < L_2 < R_2 < \ldots < L_k < R_k \leq n$. The objective is to maximize the value of $\sum_{1 \leq i \leq k} \sum_{L\_i \leq j \leq R\_i} a_j$.

a) Define in plain English the subproblems that your dynamic programming algorithm solves. (6 pts)

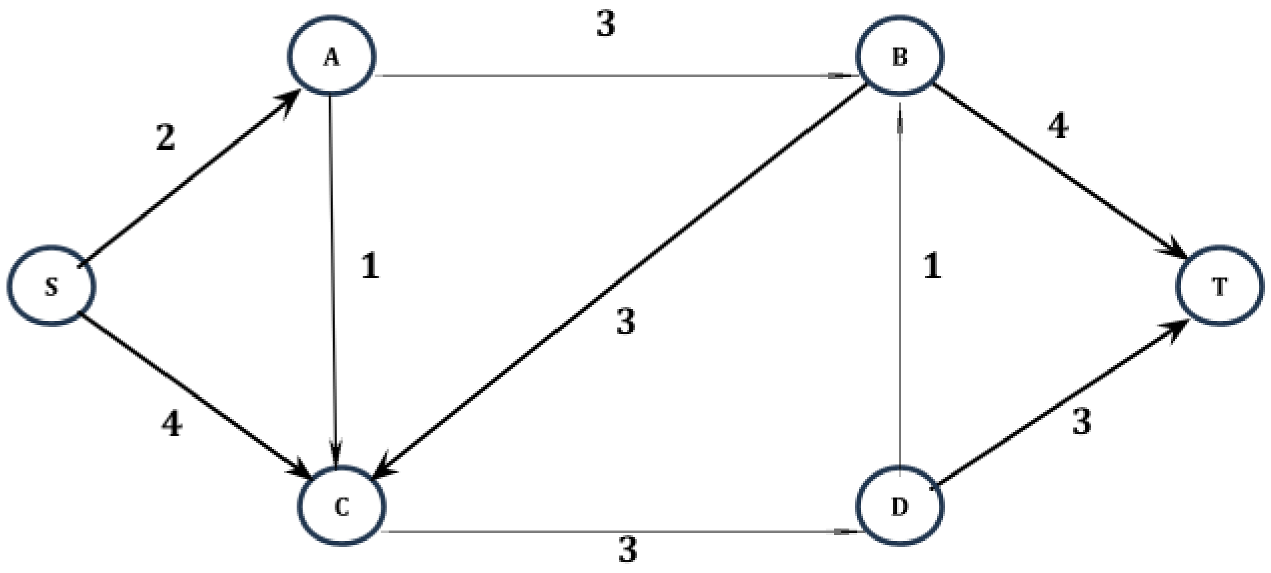
b) Define a recurrence relation that expresses the optimal solution (or its value) of each subproblem in terms of the optimal solutions to smaller subproblems. (6 pts)

c) Using the recurrence formula in part b, write pseudocode using iteration to find the maximum value. Make sure you specify base cases and their values. (6 pts)

d) What is the time complexity of your solution? (2 pts)

5) 20 pts

In the flow network illustrated below, each directed edge is labeled with its capacity. We are using the Ford-Fulkerson algorithm to find the maximum flow. The first augmenting path is S-A-C-D-T, and the second augmenting path is S-A-B-C-D-T.



a: Draw the residual networks after each of these two augmentation steps (using the above two augmenting paths in the order given). (6 pst) b: List all of the augmenting paths that could be chosen for the third augmentation step. (6 pts)

c: What is the value of the maximum flow? (4 pts)

d: Draw a dotted line through the original graph to represent the minimum cut. (4 pts)

6) 17 pts

Recall the matrix chain multiplication problem: Given the dimensions of a sequence of *n* matrices $A_1 \ldots A_n$ in an array *arr*[0..*n*], where the number of rows in the matrix A_i is *arr*[*i*-1] and the number of columns *arr*[*i*], the task is to find the most efficient way to multiply these matrices together such that the total number of element multiplications is minimized.

a) Given the notation OPT(i,j) denoting the minimum number of element multiplications to complete the product of matrices $A_i \ldots A_j$, write the recurrence formula to find the minimum number of element multiplications required. (you need to use the *arr* array to refer to the number of rows and columns) (5 pts)

Remember: if A is an m by n matrix and B is n by k, the number of element multiplications in A * B is mnk.

b) Given the array *arr*[0..4] = [4, 2, 3, 1, 3] representing the sizes of matrices $A_1 \ldots A_4$, perform a bottom up pass to find the minimum number of element multiplications. In other words, draw the two dimensional OPT array and fill in the terms in the array by numerically solving the bottom up pass. No code or pseudocode is necessary. (6 pts)

Here i and j grow along the conventional positive x and positive y direction respectively.

c) Recall that the objective is to find the most efficient way to multiply these matrices together. So using the values of the optimal solutions from part b, perform the top down pass to find the order in which the matrices should be multiplied. Again, no code or pseudocode is necessary.
Note: You need to numerically solve the top down pass and at each step show how the optimal multiplication order is revealed at that step.
(6 pts)

Additional Space

Additional Space

Additional Space