CS570 Spring2022: Analysis of Algorithms Exam II

	Points		Points
Problem 1	20	Problem 4	16
Problem 2	6	Problem 5	16
Problem 3	20	Problem 6	22
	Total	**100**	

Instructions:
1. This is a 2-hr exam. Open book and notes. No electronic devices or internet access.
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 0-1 knapsack problem, a solution that uses up all of the capacity of the knapsack will always be optimal.

[TRUE/FALSE]

Suppose that a new max-flow algorithm operating on a flow network *G* with integer capacities finds a non-integer max-flow *f,* i.e., the flow assigned to each edge is not necessarily an integer. Then there may exist some *s-t* cut *(A,B)* in *G* where $f^{out}(A) - f^{in}(A)$ is non-integer.

[TRUE/FALSE]

For every min-cut *(A,B)* and max-flow *f* in a flow network *G*, if *e=(u,v)* is a directed edge with $u \in A$ and $v \in B$, then $f(e) = c_e$.

[TRUE/FALSE]

When the Ford-Fulkerson algorithm terminates, there must not be any directed edge going out of the source node *S* in the residual graph.

[TRUE/FALSE]

If removing an internal node *u* from a flow network *G* disconnects its source from its sink, then the number of iterations required for Ford-Fulkerson to find max-flow in G will be bounded by the sum of the capacities of the edges going into *u*.

[TRUE/FALSE]

In a flow network, if all s-t cuts have integer capacities that are multiples of 3, then all edge capacities must be multiples of 3.

[TRUE/FALSE]

If all edge capacities in a flow network are the same constant integer, then a maximum flow can be found in linear time using the Ford-Fulkerson algorithm.

Continued on the next page

[TRUE/FALSE]
The Sequence Alignment problem discussed at length in lecture (between two strings of size *m* and *n*) can be solved given only *O(k)* memory space where *k=min(m,n)*

[TRUE/FALSE]
The time complexity of the space-efficient version of the sequence alignment algorithm is *O(m + n)* (between two strings of size *m* and *n*)

[TRUE/FALSE]
A flow network with unique edge capacities may have several min cuts.

2) 6 pts

a) Consider the following 0-1 knapsack problem. Given a knapsack capacity W=40, what is the maximum value that can be obtained for the following set of items? You do not need to show your work. (3 pts)

$w_0 = 10$	$w_1 = 20$	$w_2 = 30$
$v_0 = 40$	$v_1 = 60$	$v_2 = 50$

(a) 60
(b) 90
(c) 100
(d) 110
(e) 150

b) If each edge capacity of a flow network G is multiplied by the same positive integer >1, then which of the following will change? (Assume that G has a non-zero max flow) (3 pts)

A. The location of min cut, or locations of min cuts if there is more than one
B. The capacity of any min-cut
C. The maximum value of flow from source to sink
D. A and B
E. A and C
F. B and C
G. A and B and C

3) 20 pts

A company has *n* software applications. Each application *i* has *F(i)* features (or functionalities). Features are not shared across different applications. Each feature requires one DB connection from a given subset of databases *D(j)*. For example: An application *i* with *F(i)=3* features could have these database requirements: {{1, 3}, {2, 4, 5}, {6}}. This means that the first feature requires access to either database 1 or 3, the second feature requires access to databases 2, or 4, or 5, and the third feature requires access to database 6. Note that a database may be required for more than one feature.

Assuming that each database *k* can only accommodate *C(k)* connections at a given time, design a network flow solution to determine if there is an assignment of features to databases in which each application will at most have one feature without a DB connection. In other words, each application *i* should have at least *F(i)-1* of its features up and running.

a) Describe the complete construction of your network. (12 pts)

Continued on the next page

b) Which problem will you solve in this network and what algorithm will you use to solve it? (4 pts)

c) Describe how the solution to your network flow problem can be used to determine whether each application can have at least all but one of its features up and running. No proof is necessary. (4 pts)

4) 16 pts

You are given an unsorted array $A[1..n]$ of positive integers. You want to maximize the number of points you get by performing the following operations.
Pick any $A[i]$ and delete it to earn $A[i]$ points. Afterwards, you must delete elements $A[i-1]$ and $A[i+1]$ (if they exist) for which we won't get any points. This process will be called one move. Develop an efficient dynamic programming solution to return the maximum number of points you can earn with exactly k moves, where $k<n$.

Example: A=[1,8,2], k=1 Solution: 8
Example: A=[1,8,2], k=2 Solution: 3 (if we pick 8 we won't be able to make 2 moves because all elements will be deleted with that first move)

Note: if the array is of size 1 or 2 we cannot make more than one move. If the array is of size 3 or 4 we cannot make more than 2 moves, etc.

I. Define (in plain English) the subproblems to be solved. (3 pts)

II. Write a recurrence relation for the subproblems (5 pts)

Continued on the next page

III. Using the recurrence formula in part II, write pseudocode using iteration to compute the maximum number of points that can be earned. (6 pts total) Make sure you specify:

   i. the base cases and their values (2 pts)

   ii. rest of the pseudocode (3 pt)

   iii. where the final answer can be found (e.g. *OPT(n)*, or *OPT(0,n)*, etc.) (1 pt)

IV. What is the run time complexity of your solution? (2 pts)

5) 16 pts

A string is called *palindromic* if it is equal to its reverse, e.g., the string *racecar* is palindromic. Design a dynamic programming algorithm for finding the length of the longest palindromic subsequence of a given input string *S* of length *n*.

Example:

S='bbbab', output=4 (the longest palindromic subsequence is bbbb)

S ='abbcab', output=4 (the longest palindromic subsequence is abba)

I. Define (in plain English) subproblems to be solved. (3 pts)

II. Write a recurrence relation for the subproblems (5 pts)

Continued on the next page

III. Using the recurrence formula in part II, write pseudocode using iteration to compute the length of the longest palindromic subsequence. (6 pts total) Make sure you specify:

   i. the base cases and their values (2 pts)

   ii. rest of the pseudocode (3 pt)

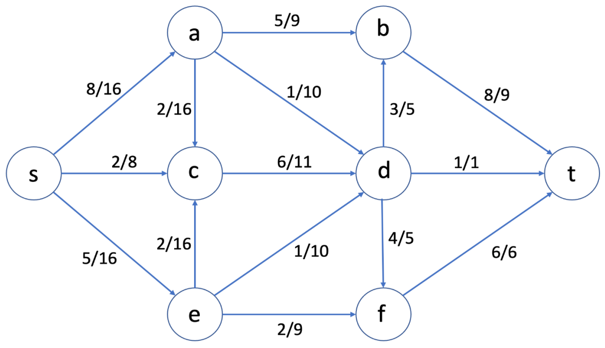
   iii. where the final answer can be found (e.g. *OPT(n)*, or *OPT(0,n)*, etc.) (1 pt)

IV. What is the run time complexity of your solution? (2 pts)

6) 22 pts

Consider the flow-network below, for which an *s*-*t* flow has been computed. The numbers *x/y* on each edge shows that the capacity of the edge is equal to *y*, and the flow sent on the edge is equal to *x*.

[redacted]



[redacted]

a. What is the current value of the flow? (2 pts)

b. Draw the corresponding residual graph. (6 pts)

Continued on the next page

c. Starting from the given flow, perform as many iterations of the Ford-Fulkerson algorithm as required to find a max flow. Draw the augmenting path at each step and draw the final max flow (6 pts)

Continued on the next page

d. Use a drawing to show a min cut in the above flow network (2 pts)

e. Assuming that there is no flow currently going through the network (i.e., zero flow going through each edge), use the scaled version of Ford Fulkerson to complete one augmentation step and draw the residual graph $G_f(\Delta)$ right after the first augmentation step. (6 pts)

Additional Space

Additional Space

Additional Space