CS570 Fall 2025: Analysis of Algorithms Exam II

	Points		Points
Problem 1	18	Problem 5	19
Problem 2	9	Problem 6	18
Problem 3	17		
Problem 4	19		
			Total 100

First name	
Last Name	
Student ID	

Instructions:

1. This is a 2-hr exam. Closed book and notes. No electronic devices or internet access.
2. A single double sided 8.5in x 11in cheat sheet is allowed.
3. 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.
4. No space other than the pages in the exam booklet will be scanned for grading.
5. 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.
6. Do not detach any sheets from the booklet.
7. 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.
8. Do not write your answers in cursive scripts.
9. This exam is printed double sided. Check and use the back of each page.

Student ID:

1) (18 points)
Mark the following statements as **TRUE** or **FALSE** by circling the correct answer. No need to provide any justification. (2 points each)

[TRUE/FALSE]
In a 0-1 knapsack problem, a solution that uses up all of the capacity of the knapsack will be optimal.

[TRUE/FALSE]
The Ford-Fulkerson algorithm can be used to find a maximum flow in a flow network that contains cycles.

[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]
Suppose an edge e is saturated due to max flow f in the flow network G. Then increasing e's capacity will increase the max flow value of the network.

[TRUE/FALSE]
In a dynamic programming solution, the value of the optimal solution should always be saved for all unique sub-problems.

[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.

[TRUE/FALSE]
The Dynamic Programming approach only works on problems with non-overlapping sub-problems.

[TRUE/FALSE]
The Ford-Fulkerson Algorithm finds a maximum flow in a unit-capacity flow network with n vertices and m edges in time O(mn).

Student ID:

2) (9 points)
For each question below, select **all** correct answers! 3 points each. **No partial credit**.

I. Which of the following hold true for a feasible circulation with lower bounds? (The variables are as defined in lectures.)

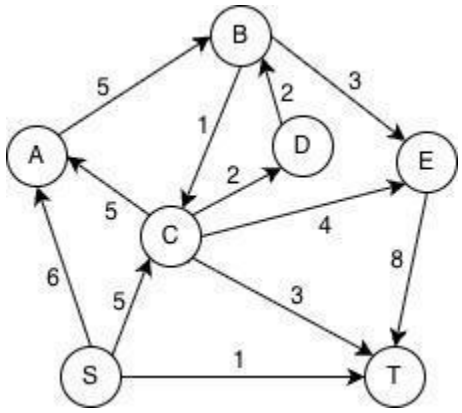
(A) For each $v \in V$: $f^{out}(v) = f^{in}(v)$
(B) For each $e \in E$: $l_e \leq f(e) \leq c_e$
(C) $\sum_{(v \in V)} f^{out}(v) = \sum_{(v \in V)} f^{in}(v)$
(D) If d_v is an integer for each $v \in V$, and l_e, c_e are integers for each $e \in E$, then $f(e)$ is an integer for each $e \in E$

II. Which of these give a correct bound on the runtime of the Bellman–Ford algorithm?
(A) **O** (|V|^3)
(B) **Θ** (|E|^2) for graphs where each vertex has exactly 10 incident edges
(C) **Θ** (|V|*(|E|+|V|))
(D) **Θ** (|V||E|)
(E) Θ (|E|log|V|)

III. Which of the following run time complexities are considered to be efficient?
(A) Run time is polynomial time with respect to the number of integers in the input
(B) Run time is polynomial time with respect to the number of bits in the input
(C) Run time is polynomial time with respect to the numerical value of input terms
(D) Run time is logarithmic time with respect to the numerical value of input terms

Student ID:

3) (17 points)
Consider the following flow network *G* with source S and sink T, and edge capacities as marked on the edges in the diagram below.



a) Find a maximum flow in *G* using the Edmonds-Karp algorithm. For each iteration, you need to show the augmentation path, and the bottleneck value. (10 pts)

Edmond-Karp computes (unweighted) shortest paths (a path from *S* to *T* that uses as few edges as possible) at each augmentation. The resultant augmenting paths, in order:

1. ST (bottleneck value 1)
2. SCT (bottleneck value 3)
3. SCET(bottleneck value 2)
4. SABET (bottleneck value 3)
5. SABCET (bottleneck value 1)

Rubrics: 2 points per iteration (1 +1 for path and bottleneck)

See next page for parts b and c

b) What is the value of the maximum flow in *G*? (2 pts)

The value of the max-flow is 10.
Rubric: 2 points for a correct answer.

c) Identify two min-cuts in *G*. (5 pts)

A min-cut in *G* is ({S, A, B}, {C, D,E, T}).
Another is ({S, A, B, D}, {C, E, T}).

Rubric: 2.5 points for each cut

Student ID:

4) (19 points)

An adventure sport tournament is being organized where participants cross a river while jumping on tiny platforms. The participants start on one side of the riverbank, and have *N* platforms positioned in a straight line (numbered 1, …, n) at uniform distances to cross the river and get to the other side. The participants are not allowed to turn around at any point and must make their jumps forward.

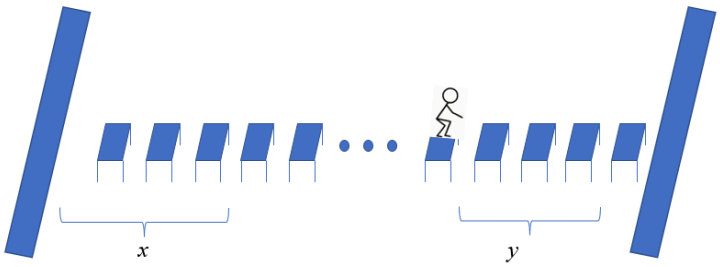
When taking off the riverbank, a participant is able to jump anywhere up to platform no. *x*, but taking a jump from a platform is a bit more difficult, so, in each subsequent jump, a participant can only land on a platform (or opposite riverbank) that is at most *y* positions further ahead.

To make things challenging, the platforms are constructed to not be very durable and tend to get weaker with every jump onto it. Consequently, each platform can only withstand *r* jumps onto it and another participant (r+1st) jumping onto it will result in a fatal accident.

Finally, a participant is disqualified if they repeat another participant's jump, that is, if one participant jumps from platform *i* to *j*, any other participant landing on platform *i* cannot jump to *j* anymore. This rule does not apply to jumping from and to a riverbank.

Our goal is to determine the maximum number of participants that can successfully finish the tournament.

a) Given the input *N, x, y, r* mentioned above, design an efficient flow network based algorithm to find the maximum number of participants that can successfully finish the tournament under the constraints above.
Clearly describe all the components of the network that is constructed and the final answer obtained by your algorithm. (13 points)



Solution:

Construct a network as follows:

1) Add an edge (p^{in}, p^{out}) for each platform p, and set cap *r* for each.

2) For each platform p and platform q within *y* positions ahead of p, create edge (p^{out}, q^{in}), with cap 1.

3) Create source S and add edges (S, p^{in}) for each of the first x platforms (unlimited capacity)

4) Create sink T and add edges (p^{out}, T) for each of the last y platforms (unlimited capacity)

See next page for part b

Find max-flow with a poly-time algorithm such as Edmonds-Karp, this gives the largest set of subsequences under the given constraints.

Rubric:
3 points for representing platforms as edges (p_{in}, p_{out})
2 points for edges (p_{out}, q_{in}) correctly described to be at most y positions apart
1.5 + 1.5 points for capacity r on edges (p_{in}, p_{out}) and 1 on edges (p_{out}, q_{in}) resp.
1 point for creating source S and edges (S, p_{in}) and 1 point for sink T and edges (p_{out}, T)
2 points for unlimited (or sufficiently large, e.g., r*n) capacity on edges connecting S or T
1 point for calling a max flow algorithm

b) Prove the correctness of your algorithm. (6 points)

Claim: The network has a max flow of value Z if a maximum of Z people can successfully finish the tournament.

Forward proof. Suppose Z people can cross. For each participant making a jump from riverbank or platform to a riverbank or platform - add 1 unit of flow to the corresponding edge in the graph, and 1 unit to edge (p^{in}, p^{out}) every time anyone lands on platform p. This gives a max flow of value Z.

Backward proof. Suppose max flow value is Z. For each S-T path of unit flow, assign the corresponding platform jumps to a participant, thus Z participants in total. Capacity constraints of (p^{in}, p^{out}) ensure no platform breaks. Capacity constraint of any (p^{out}, q^{in}) ensures jump (p,q) is not repeated.

Rubric:
1 point for correct claim
1 point for breaking down into both directions
2 points for correct forward proof
    1 for describing how to get max flow given tournament solution
    1 for justifying flow constraints and value
2 points for correct backward proof
    1 for describing how to get tournament solution given max flow
    1 for justifying tournament constraints satisfied due to flow constraints

Student ID:

5) (19 points)
Imagine you are driving on a k-lane highway. The route can be viewed as consisting of points $p_0, \ldots, p_n$, where p_0 is the starting point, p_n the ending point, and you can switch lanes at any p_i but must stick to a lane between any p_i to p_{i+1}. Thanks to the advanced navigation technology, we have accurate estimates for how long it takes to travel a section in a particular lane: going from p_i to p_{i+1} in lane j requires T_{ij} minutes. In general, you would want to pick the fastest lane available for each section, but there is a catch - switching lanes requires slowing down and merging, which adds a delay. In particular, switching to the immediate next lane adds a delay of 2 minutes; thus, switching from lane h to lane j adds a delay of 2*|h-j| minutes. Your goal is to plan the route to compute the minimum possible travel time.

Example: Consider the route to be p_0–p_1–p_2 and k=3 lanes. Traversing p_0–p_1 via the 3 lanes takes 5,10,10 minutes respectively, and traversing p_1–p_2 takes 10,10,5 minutes respectively. Then, using lane 1 until p_1 takes 5 minutes, switching from lane 1 to lane 3 at p_1 adds **4** minutes, and going to p_2 in lane 3 takes another 5 minutes, giving a total of 14 minutes, which is the least time possible.

Given the number of lanes k, and all the times T_{ij} ($0 \le i \le n$, $1 \le j \le k$), **design an algorithm using dynamic programming** to achieve this. You should provide your solution using the 4 part template (a through d) below.

a) Define in plain English the **subproblems** that your DP algorithm solves. (4 points)

OPT(i,j) = Min. time to reach p_i while arriving in lane j ($0 \le i \le n$, $1 \le j \le k$)

b) Define a recurrence relation that expresses the value of the optimal solution for each subproblem in terms of that for smaller subproblems. (6 points)

$OPT(i,j) = \min_{1 \le h \le k} \{OPT(i-1,h) + 2*|h-j| + T_{(i-1)j}\}$

See next page for parts c and d

Student ID:

c) Using the recurrence formula in part b, write pseudocode to find the solution. Make sure you specify all the base cases and their values, as well as the final answer to the problem. (6 points)

Pseudocode should correctly show the following:
Base case: $OPT(0,j) = 0$ for all $1 \leq j \leq k$

Final answer: $\min_{1 \leq j \leq k} OPT(n, j)$

Computation order: i from 1 to n, j in any order

Rubric:
2 points for Base cases
2 for final answer
1 for correct loop order for i (j order does not matter)
1 for correctly calling the recurrence and computing min over all $1 \leq h \leq k$

d) What is the time complexity of the solution? Further, determine whether it is an efficient solution or not. (3 points)

$O(nk^2)$. Polynomial time (efficient!)

Rubrics: 2 points for correct runtime, 1 for efficient or not.

Student ID:

6) (18 points)
Recall the DNA string matching problem where given two strings $X = (x_1 x_2 \ldots x_m)$ and $Y = (y_1 y_2 \ldots y_n)$ we are asked to find the similarity and the optimal alignment between the two strings. Also recall that similarity was defined as the minimum cost of an alignment between the two strings where the two cost factors are:

- For each gap we incur a penalty of δ.
- For each mismatch between letters p and q we incur a mismatch penalty of α_{pq}.

A possible dynamic programming formulation for this problem defines the following subproblems:

OPT(i,j) = The minimum cost of aligning $\mathbf{(x_i \ldots x_m)}$ and $\mathbf{(y_j \ldots y_n)}$
(i > m and j > n denote empty substrings)

Note that in this formulation OPT(1,1) will hold the similarity of X and Y.

a) Write the recurrence formula to compute OPT(i,j) as defined above for $1 \le i \le m$, $1 \le j \le n$. (2 pts)

OPT(i,j) = Min {OPT(i+1, j) + δ, OPT(i, j+1) + δ, OPT(i+1,j+1) + α_{pq}}

Where p = x_i and q = y_j

b) Suppose the gap penalty is $\delta = 8$ and the mismatch penalties are given by the following matrix, where each (p,q) entry shows the mismatch penalty α_{pq}.

	A	C	G	T
A	0	5	10	20
C	5	0	5	10
G	10	5	0	5
T	20	10	5	0

We want to compute the minimum cost alignment for X = ATG and Y = CA.
Fill in the OPT array shown below by numerically solving the bottom-up pass.
You must extend the number of rows and columns in the OPT array below such that it contains your entire tabulation.
No code or pseudocode is necessary. (10 points)

	j=1	j=2	…
i=1			
i=2			
…			

See next page for part c

Student ID:

Solution:

	j=1	j=2	j=3
i=1	23	16	24
i=2	20	18	16
i=3	13	10	8
i=4	16	8	0

Rubric:
1 point for the base case (i=4, j=3)
1 points for rest of row 4
1 points for rest of column 3
1 point each for the remaining 6 values (i=1,2,3 and j=1,2)
1 bonus if all correct.

c) Recall that the objective is to find the optimal alignment between the two strings. So, using the values of the optimal solutions from part b, perform the top-down pass to find the optimal alignment showing which elements of X and Y are paired with each other and where the gaps should be placed in the optimal alignment.
(6 pts)

Solution)
We start at (1,1). We compare the three terms from the recurrence, to find that the value (23) comes from 5 + OPT(2,2). Thus, this corresponds to matching X_1 and Y_1.

Next, we do the same for OPT(2,2) and see that the value 18 comes from 8 + OPT(3,2), thus X_2 was matched to a gap inserted in Y.

Finally OPT(3,2) has value 10 which comes from 10 + OPT(3,3). Thus, this means that matching X_3 and Y_2 were matched. OPT(3,3) is a subproblem corresponding to matching empty strings, so the top-down pass ends.

Thus, our answer is ATG mapped with C_A.

Rubric:
2 pts for each of the 3 steps

Student ID:

Additional Space

Student ID:

Additional Space

Student ID:

Additional Space