CS 170 Midterm 2

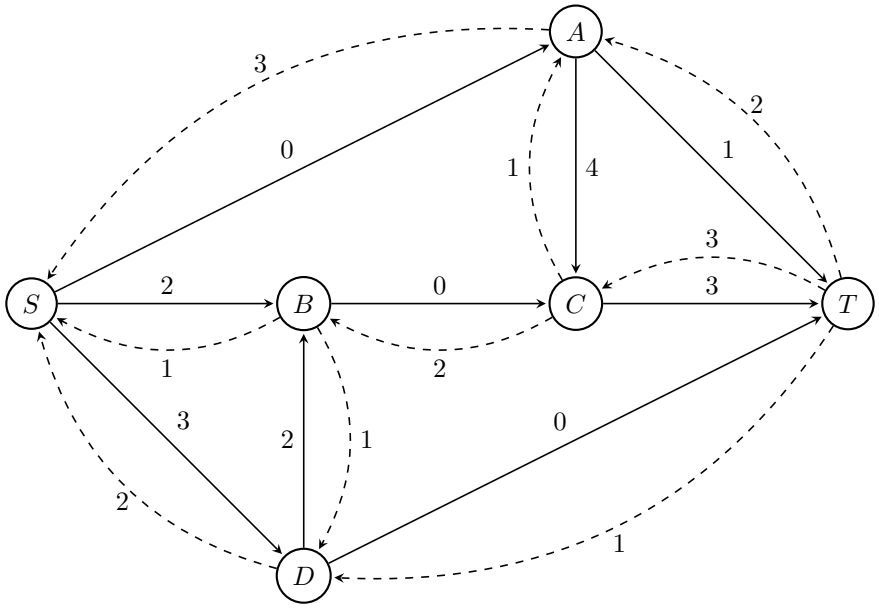
Write in the following boxes clearly and then double check.

Name:	
SID:	

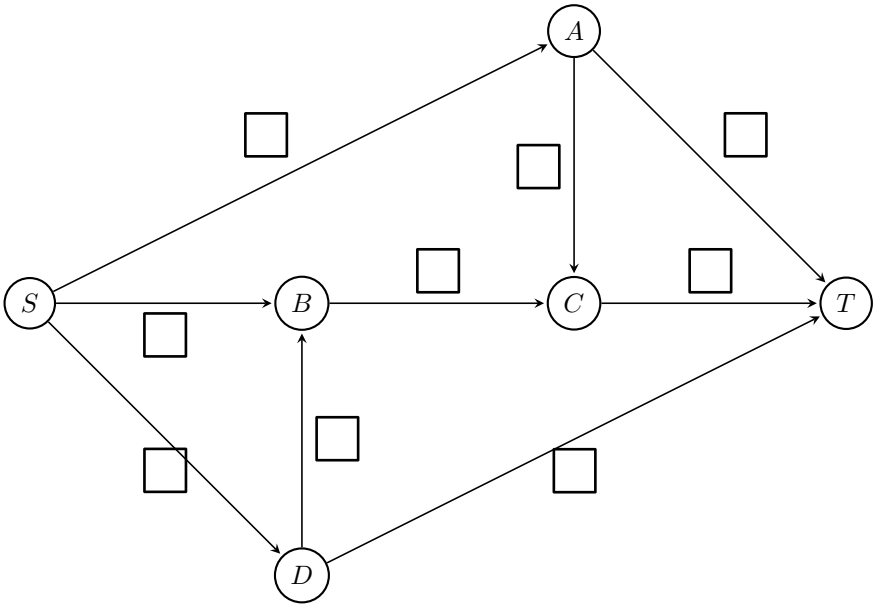
- **The exam will last 110 minutes.**
- The exam has 8 questions with a total of 100 points. You may be eligible to receive partial credit for your proof even if your algorithm is only partially correct or inefficient.
- Only your writings inside the answer boxes **on the answer sheet** will be graded. **Anything outside the boxes and on this exam booklet will not be graded.** The last page is provided to you as a blank scratch page.
- Answer all questions. Read them carefully first. Not all parts of a problem are weighted equally.
- Be precise and concise.
- The problems may **not** necessarily follow the order of increasing difficulty.
- The points assigned to each problem are by no means an indication of the problem's difficulty.
- The boxes assigned to each problem are by no means an indication of the problem's difficulty.
- Unless the problem states otherwise, you should assume constant time arithmetic on real numbers. Unless the problem states otherwise, you should assume that graphs are simple.
- If you use any algorithm from lecture and textbook as a blackbox, you can rely on the correctness and time/space complexity of the quoted algorithm. If you modify an algorithm from textbook or lecture, you must explain the modifications precisely and clearly, and if asked for a proof of correctness, give one from scratch or give a modified version of the textbook proof of correctness.
- Unless the problem states otherwise, assume the subparts of each question are **independent**.
- Please write your SID on the top of each page.
- Good luck!

1 Ford Fulkerson, Interrupted (8 points)

We ran the Ford Fulkerson algorithm on a directed graph, and after three iterations, the residual graph looks as follows. In this picture, the solid edges correspond to the original edges in the graph, the dotted edges are the backwards edges in the residual graph, and the numbers indicate the capacities of each edge in the residual graph.



(i) (2 points) Write down the capacities of the edges as they appear in the original graph.



(ii) (3 points) In each iteration, the Ford Fulkerson algorithm picks a path in the residual graph and sends some units of flow along that path. Write down the three paths that the Ford Fulkerson algorithm picks, in the order that it picks them, which result in this residual graph. In addition, write the amount of flow that it sends along each path.

Exam continues on next page ⟹

	Vertices along path (in order)	Flow on path
Path 1		
Path 2		
Path 3		

(iii) (1 point) What is the maximum flow of this graph?

(iv) (2 point) List the minimum $S-T$ cut in the graph. **List vertices on each side in alphabetical order**.

Vertices on S side of the cut	Vertices on T side of the cut

2 In a Mirror, Dually (8 points)

Consider the following primal linear program Prime.

$$\begin{aligned} \text{Maximize } & -8x - 7y - 2z \\ \text{Subject to } & \quad + 5y + 2z \leq 8, \\ & -5x \quad - 2z \leq 7, \\ & -2x + 2y \quad \leq 2, \\ & \quad x, y, z \geq 0. \end{aligned}$$

(i) (**3 points**) Derive the dual linear program Dual (use variables a, b, c).

(ii) (**2 points**) Suppose (x, y, z) is a feasible solution to the primal linear program Prime from above with objective value $v \in \mathbb{R}$. Explain why $a = x, b = y, c = z$ is also a feasible solution to the dual linear program Dual, and give its dual objective value (in terms of v).

(iii) (**3 points**) Suppose P is a primal maximization LP and D is its dual. (Note: this subproblem does *not* refer to the primal LP Prime from above.) Suppose that for all $v \in \mathbb{R}$, P has a feasible solution with value v if and only if D has a feasible solution with value $-v$. Prove that if P has a bounded optimum, then its optimum value is 0. (Hint: use strong duality.)

3 Simplex on the hypercube (3 points)

Consider the following linear program.

$$\begin{aligned} \text{Maximize } & 10 \cdot x_1 - 20 \cdot x_2 + 15 \cdot x_3 - 45 \cdot x_4 + 30 \cdot x_5 \\ \text{Subject to } & 0 \le x_1 \le 1, \\ & 0 \le x_2 \le 1, \\ & 0 \le x_3 \le 1, \\ & 0 \le x_4 \le 1, \\ & 0 \le x_5 \le 1. \end{aligned}$$

This has 2^5 vertices, corresponding to each point $x \in \{0,1\}^5$. Suppose we run the simplex algorithm on this linear program, starting at the vertex $x = (0,1,0,1,0)$. Write down the sequence of vertices it visits until it reaches the optimum, along with their objective values. (Note: there may be more lines than vertices visited.)

	Vertex	Objective value
Vertex 1	$(0,1,0,1,0)$	-65
Vertex 2		
Vertex 3		
Vertex 4		
Vertex 5		
Vertex 6		
Vertex 7		
Vertex 8		
Vertex 9		

4 Experts (4 points)

(i) (**2 points**) Suppose we run the multiplicative weights algorithm on n experts with parameter $\epsilon > 0$ in the setting where all losses are 0 or 1. Let $w_i^{(t)}$ be the weight the algorithm gives to expert i at the beginning of day t, and let

$$W^{(t)} = \sum_{i=1}^{n} w_i^{(t)}$$

be the total weight of all experts on day t. (Recall that the algorithm starts on day 1 with $w_i^{(1)} = 1$ for all $1 \leq i \leq n$.)

(a) Suppose $W^{(T+1)} = n$. What is the loss of the algorithm after T days? What is its regret?

Loss:		Regret:	

(b) Suppose $W^{(T+1)} = n \cdot (1-\epsilon)^T$. What is the loss of the algorithm after T days? What is its regret?

Loss:		Regret:	

(ii) (**2 points**)

(a) Suppose we have $n = 2$ experts A and B. Consider the following algorithm: "Always pick the expert with the lowest total loss so far. If A and B are tied, pick A." Specify a sequence of 0/1 losses for the experts over 8 days such that the algorithm has total loss 8 and as big a regret as possible.

	Loss of A	Loss of B
Day 1		
Day 2		
Day 3		
Day 4		

	Loss of A	Loss of B
Day 5		
Day 6		
Day 7		
Day 8		

(b) What is the loss of the multiplicative weights algorithm after 8 days? What is its regret? Use the parameter $\epsilon = 1/2$.

Loss:		Regret:	

5 Golden State Warriors (6 points)

Steph Curry has the basketball, and he is trying to decide which range to shoot it from:

- (Option 1) close range,
- (Option 2) medium range, or
- (Option 3) long range.

However, he is being defended by Jaren Jackson Jr., who will decide to guard one of the three ranges (either close, medium, or long). If he guards the same range that Steph shoots from, he will block Steph's shot, and Steph will receive 0 points. However, if Jaren guards a different range, then Steph will make his shot. In this case, Steph will receive 2 points if he shot from close or medium range and 3 points if he shot from long range.

(i) Modelling the situation as a zero-sum game between Steph Curry (as the row player) and Jaren Jackson Jr. (as the column player), what is the payoff matrix?

(ii) Write the linear program for Steph Curry's optimal strategy.

6 Dynamic Programming Potpourri (12 points)

(i) Recall the knapsack without repetition problem. You have a knapsack with capacity W, and there are n items of weights $w_1, ..., w_n$ and values $v_1, ..., v_n$. Your goal is to determine the most valuable combination of items that can fit in the knapsack given that there is one of each item. In class, we saw a dynamic programming algorithm for this problem using a 2-dimensional array K which has an entry $K[c, i]$ for all integers $0 \leq c \leq W$ and $0 \leq i \leq n$.

Suppose we are given an instance of this problem with $W = 10$, $n = 5$, and we run the dynamic programming algorithm on it, producing the following array K.

	0	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0	***a***
1	0	0	0	1	1	1	1	1	1	1	1
2	0	0	0	2	2	2	3	3	3	3	3
3	0	0	0	2	2	2	4	4	4	***b***	5
4	0	0	4	4	4	***c***	6	6	8	8	8
5	0	0	4	4	6	6	10	10	10	***d***	12

Note that four entries have been erased and replaced with the variables ***a***, ***b***, ***c***, and ***d***.

(a) (**3 points**) Below we have filled in the weights w_3, w_4, w_5 and the values v_3, v_4, v_5. What are the remaining weights and values?

w_1:		w_2:		w_3:	3	w_4:	2	w_5:	4
v_1:		v_2:		v_3:	2	v_4:	4	v_5:	6

(b) (**3 points**) What are the values of the missing cells?

a:		***b***:		***c***:		***d***:	

(ii) Recall the edit distance problem. You have two strings $x[1 \ldots n]$ and $y[1 \ldots m]$, and you want to compute the minimal number of edits needed to transform x into y. Allowable edits are inserting a character, deleting a character, and substituting one character for another. In class, we saw a dynamic programming algorithm for this problem using a 2-dimensional array E which has an entry $E[i, j]$ for all integers $0 \leq i \leq n$ and $0 \leq j \leq m$.

(a) (**3 points**) Suppose we run the dynamic programming algorithm on a string x and the string $y =$ "apple", and the resulting array E is given as follows.

	0	1	2	3	4	5
0						
1		1	1			
2		1				
3					2	
4						2

Note that we have erased all but five of the entries in E. What string is x?

x_1:		x_2:		x_3:		x_4:	

(b) (**3 points**) For the last three questions, we will suppose we have run the dynamic programming algorithm on strings x and y, resulting in the following array E.

Exam continues on next page ⟹

	0	1	2	3	4	5
0						
1						
2						
3						
4						
5						

We will focus on the bottom-right 2×2 subarray which we have filled in with gray. The following three questions are true or false and worth 1 point apiece.

i. It is possible for this 2×2 subarray to looks as follows: $\begin{array}{|c|c|}\hline 0 & 1 \\ \hline 1 & 0 \\ \hline \end{array}$.

◯ True ◯ False

ii. Suppose this subarray looks as follows: $\begin{array}{|c|c|}\hline 2 & 1 \\ \hline 3 & 2 \\ \hline \end{array}$. Then it must be that $x_5 = y_5$.

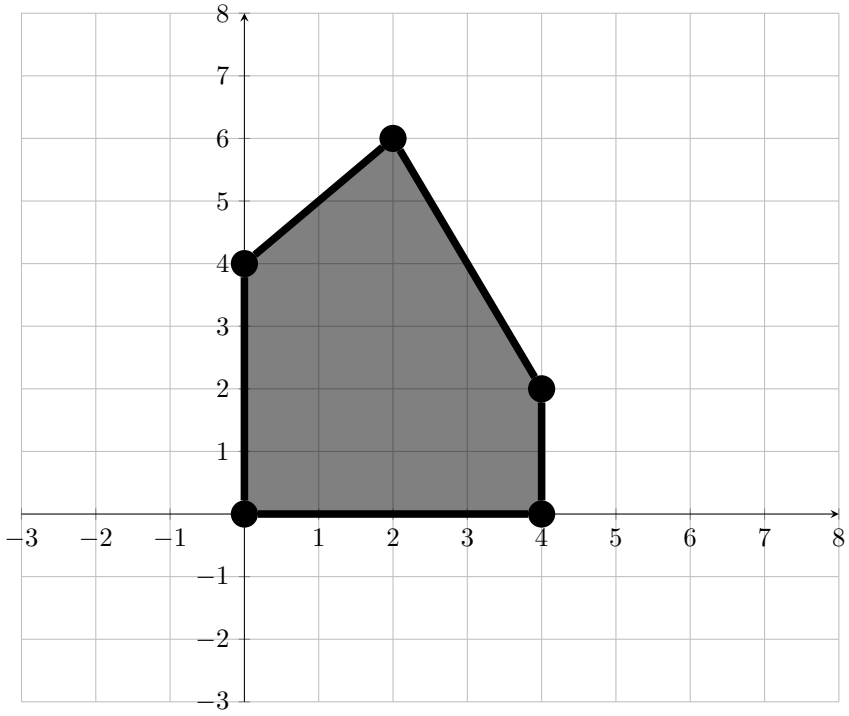
◯ True ◯ False

iii. The following is a legal configuration of this subarray: $\begin{array}{|c|c|}\hline 4 & 5 \\ \hline 4 & 5 \\ \hline \end{array}$.

◯ True ◯ False

7 LP Short Answers (8 points)

Consider a linear program with two variables x and y that has the following feasible region. The horizontal axis is the x-axis and the vertical axis is the y-axis. The vertices are $(0,0)$, $(4,0)$, $(4,2)$, $(2,6)$, and $(0,4)$.



(i) (**2 points**) Suppose that the objective function is "maximize $3x + y$". Select all of the following objective values which are possibly achievable by the **dual** linear program.

◯ 11	◯ 12	◯ 13	◯ 14	◯ 15

(ii) (**2 points**) Suppose that the objective function is "minimize $x - y$". Select all of the following points (x, y) which are feasible and achieve the optimal value.

◯ $(0,4)$	◯ $(2,8)$	◯ $(1,5)$	◯ $(1,4)$	◯ $(4,2)$

(iii) (**2 points**) Suppose that the LP's objective function is not constant. In addition, suppose that $(4,2)$ is an optimal point. Select all of the following points (x, y) that must have a strictly worse objective value than $(3,2)$.

◯ $(1,5)$	◯ $(2,1)$	◯ $(2,4)$	◯ $(2,6)$	◯ $(3,1)$

(iv) (**2 points**) Suppose that both $(4,2)$ and $(2,6)$ are both optimal points of the LP. Select all of the following points which must also be optimal points of the feasible region.

◯ $(4,0)$	◯ $(5,0)$	◯ $(3,4)$	◯ $(1,5)$	◯ $(2,5)$

Exam continues on next page ⟹

8 Faculty Advising (15 points)

Every semester, UC Berkeley schedules *faculty advising sessions* between its students $s_1, \ldots, s_n$ and professors $p_1, \ldots, p_m$. Each professor holds a single advising session with a group of students, and each student attends one of these group advising sessions.

- Each student s_i has a list of preferred professors $L_i \subseteq \{p_1, \ldots, p_m\}$. They want to meet with one professor from this list.
- Each professor p_j has an integer bound $b_j \in \{1, 2, 3, \ldots, \}$. They want at most b_j students in their advising session.

We would like to use one Maximum Flow computation to decide whether one can set up the faculty advising sessions so that the preferences of all students and professors are satisfied. Construct a directed graph G with a source s and a sink t which we can use to solve this problem by computing the Maximum Flow from s to t.

(i) Describe the nodes of the graph G.

(ii) Describe the edges of the graph G (draw a picture if needed).

UC Berkeley has five departments $d_1, \ldots, d_5$, and each professor p_j belongs to one of these five departments. Each department is expected to do an equal amount of work in the faculty advising sessions. This means that the number of students advised by professors in each department d_i should be exactly equal to $n/5$ (assume that n is divisible by 5).

(iii) Describe how to modify your graph G from parts (i) and (ii) such that we can see if is possible to satisfy this and the preferences of all students and professors using one Maximum Flow computation.

9 Topologically sorting general graphs (15 points)

Let $G = (V, E)$ be a directed graph. A *topological sort* of G is an ordering of its vertices from left to right so that all directed edges go from left to right. We saw in class that it is possible to topologically sort G if and only if G has no cycles. However, we could still try to topologically sort general graphs, including those which do have cycles—it won't be perfect, but we could still try to do as good as possible.

Given an ordering of the vertices $v_1, \ldots, v_n$ consider the number of edges $(v_j \to v_i) \in E$ with $j > i$. We define the *back count of* G, denoted $\text{Back}(G)$, to be the minimum of this quantity over all orderings of the vertices. Devise a dynamic programming algorithm for computing $\text{Back}(G)$. (A runtime of $O(n^2 \cdot 2^n)$ or better is required for full credit. Partial credit is available for runtimes asymptotically faster than $O(n!)$.)

(i) What are the subproblems? (Precise and succinct definition needed.)

(ii) What is the recurrence relation? (Note: no need to specify base cases.)

10 Chess dominos (15 points)

You have a $3 \times n$ chess board, consisting of 3 rows and n columns, for a total of $3n$ squares. You also have a set of dominos, which are 2×1 tiles which you can orient either vertically or horizontally. A *tiling* of your chess board by dominos involves covering your chess board with dominos according to the following rules:

- each domino is placed either vertically or horizontally on 2 adjacent squares of the chess board,
- no two dominos overlap, and
- every square of the chess board is covered by a domino.

Your goal is to compute the number of ways you can tile your chess board with dominos. Design a dynamic programming algorithm for computing this number.

(i) What are the subproblems? (Precise and succinct definition needed. No need to state the recurrence relation.)

(ii) How many subproblems are there? (In other words, how many subproblems must be computed to find the answer to a $3 \times n$ board?)

Blank scratch page.
DO NOT DETACH THIS PAGE.