CS 170 Midterm 2 Solutions

Write in the following boxes clearly and then double check.

Name : PNPenguin

SID :

Exam Room : ◯ Pimentel 1 ◯ Dwinelle 155 ◯ VLSB 2050 ◯ Other (Specify):

Name of student to your left :

Name of student to your right :

- **The exam will last 110 minutes.**
- The exam has 10 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 writing inside the answer boxes will be graded. **Anything outside the boxes 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.
- 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 black box, 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.
- Assume the subparts of each question are **independent** unless otherwise stated.
- **Please write your SID on the top of each page; you will get 1 point for doing so.**
- **For multiple choice questions, please fill in the bubbles fully.**
- Good luck!

1 Potpourri (10 points)

1. In a zero sum game, the optimal strategy is **never** deterministic. ◯ True ◯ False

2. If a standard form LP has n variables and m constraints (not including the non-negativity constraints), how many variables does its dual have (not including the non-negativity constraints)? How many constraints does its dual have?

   Variables: Constraints:

3. In a max flow problem, the value of the minimum s-t cut can be greater than the value of the maximum flow. ◯ True ◯ False

4. Consider the linear program

$$\max x$$

   subject to

$$-x \leq -1$$
$$x \geq 0$$

   (a) Is this LP feasible? If so, compute the optimal value.

   ◯ Feasible ◯ Infeasible

   (b) Write down the dual of the above LP.

   (c) Is the dual you just wrote down feasible? If so, compute the optimal value.

   ◯ Feasible ◯ Infeasible

Exam continues on next page

Solution:

1. False; sometimes it could be deterministic (say if one row dominates the other row).
2. m variables, n constraints.
3. False.
4. (a) Feasible. The first constraint becomse $x \geq 1$. So the optimal value is $+\infty$.

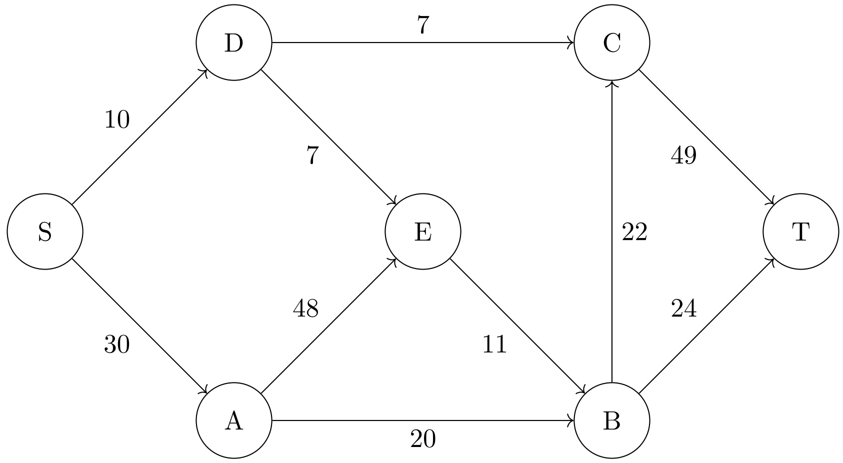
   (b) The dual is
   $$\min -y$$
   subject to
   $$\begin{aligned} -y &\geq 1 \\ y &\geq 0 \end{aligned}$$

   (c) Not feasible. The first constraint becomes $y \leq -1$. This contradicts the second constraint.

Exam continues on next page

2 Mechanical Max Flow (10 points)

Consider the following Max Flow problem where S is the source and T is the sink.



If the first path found by the Ford-Fulkerson algorithm is $S \to D \to E \to B \to T$, compute the maximum flow that can be sent along this path.

Compute the residual capacities after sending the maximum flow possible along that path.

Edge	Capacity	Edge	Capacity
$S \to D$		$D \to S$	
$D \to E$		$E \to D$	
$E \to B$		$B \to E$	
$B \to T$		$T \to B$	

Compute the maximum flow of this graph.

Which vertices are in S's side of the minimum cut?

A	◯ In ◯ Not in	B	◯ In ◯ Not in	C	◯ In ◯ Not in
D	◯ In ◯ Not in	E	◯ In ◯ Not in	T	◯ In ◯ Not in

Exam continues on next page

Solution:

1. Maximum flow along this path is equal to the minimum capacity edge, which is $D \to E$ with capacity 7.

2.

Edge	Capacity	Edge	Capacity
$S \to D$	3	$D \to S$	7
$D \to E$	0	$E \to D$	7
$E \to B$	4	$B \to E$	7
$B \to T$	17	$T \to B$	7

3. 38

4. A, E, D are in S's side of the cut.

3 Mechanical LP (10 points)

You are given the following linear program.

$$\max a + b$$

subject to

$$\begin{aligned} b &\leq 100 \\ a + 2b &\leq 300 \\ 2a + b &\leq 450 \\ a, b &\geq 0 \end{aligned}$$

(a) Draw the feasible region on the provided graph.

(b) List all of the vertices.

(c) Starting with the vertex $a = 0, b = 100$, list the coordinates of the vertices visited by the Simplex algorithm.

Exam continues on next page

As a reminder, here is the Linear Program.

$$\max a + b$$

subject to

$$\begin{aligned} b &\leq 100 \\ a + 2b &\leq 300 \\ 2a + b &\leq 450 \\ a, b &\geq 0 \end{aligned}$$

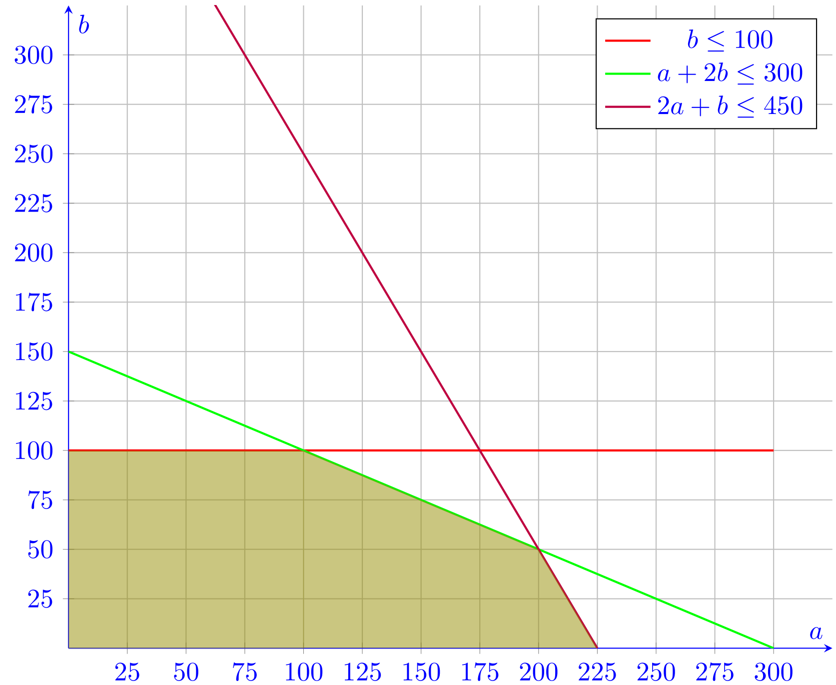
(d) Compute the optimal value of the LP.

(e) Compute the dual.

Exam continues on next page ⟹

Solution:

(a) The feasible region is indicated in yellow below:



(b) Vertices are (in clockwise order): $(0, 0); (0, 100); (100, 100); (200, 50); (225, 0)$.

(c) $(0, 100); (100, 100); (200, 50)$.

(d) The values of the objective function are: $0, 100, 200, 250, 225$. Therefore, the optimal value is 250, achieved at $(a, b) = (200, 50)$.

(e) We have 3 dual variables, one for each constraint. Thus, our objective is

$$\min 100y_1 + 300y_2 + 450y_3$$

subject to constraints

$$\begin{aligned} y_2 + 2y_3 &\ge 1 \\ y_1 + 2y_2 + y_3 &\ge 1 \\ y_1, y_2, y_3 &\ge 0 \end{aligned}$$

4 Mechanical Zero Sum Game (6 points)

Consider a two-player zero sum game with the following payoff structure:

2	10
8	4
4	5

Compute the row player's optimal strategy. (Hint: does the row player need to use all of the rows?)

$p_1 =$ $p_2 =$ $p_3 =$

Suppose the row player plays optimally. Then the row player's expected payoff is:

○ 4 ○ 5 ○ 6 ○ Depends on column player's strategy

For this question, assume that the column player's strategy is $q = (q_1, q_2) = (0.5, 0.5)$. If the column player plays this, the row player's expected payoff is:

○ 4 ○ 5 ○ 6 ○ Depends on row player's strategy

Suppose that we replace the 4 in the $(2, 2)$ entry of the payoff matrix with a 9. What is the value of this new game?

Solution:

1. Here, $p_3 = 0$ since any p_3 could be split equally among p_1 and p_2, with better payoff. With this in mind, we want to set the two values equal, so $2p_1 + 8p_2 = 10p_1 + 4p_2 \implies p_1 = 1/3, p_2 = 2/3$.

2. If the column player picks the first column the row player gets 6. If the column player picks the second column the row player still gets 6. So the answer here is 6.

3. The expected payouts for the rows are: 6, 6, 4.5. Because the row player doesn't have to play optimally, here the payoff depends on the row player's strategy.

4. If we replace the $(2, 2)$ entry with a 9, we still have $p_3 = 0$ using similar reasoning as before. Maximizing the minimum of $2p_1 + 8p_2$ and $8p_1 + 9p_2$ yields the optimal point $p_2 = 1$ and $p_1 = 0$. So, if the row player always chooses the second row, the column player will always choose the first column so the game has value 8.

Exam continues on next page

5 Planting Trees Revisited (12 points)

You have a garden and want to plant some apple trees in your garden, so that they produce as many apples as possible. There are n adjacent spots numbered 1 to n in your garden where you can place a tree. Based on the quality of the soil in each spot, you know that if you plant a tree in the i-th spot, it will produce exactly x_i apples. However, trees need space to grow, so any 4 consecutive spots can't contain more than 2 trees. Devise a dynamic programming algorithm to help you compute the maximum number of apples you can produce.

1. What are the subproblems? (precise and succinct definition needed)

2. What is the recurrence relation?

3. What is the runtime?

Exam continues on next page

Solution: Below we provide 2 ways to approach this problem:

Solution 1. For each $i \in [n]$ and boolean array A of length 3, define $f(i, A)$ to be the maximum number of apples you can produce on spots 1 to i given that there is a tree at position $i + j$ for all $j \in \{1, 2, 3\}$ such that $A[j] = 1$. The final answer is $f(n, [0, 0, 0])$.

The recurrence relation for these subproblems is

$$f(i, A) = \begin{cases} f(i-1, [0, A[1], A[2]]) & sum(A) = 2 \\ \max(f(i-1, [0, A[1], A[2]]), x_i + f(i-1), [1, A[1], A[2]]) & sum(A) < 2 \end{cases}$$

Runtime is $O(n)$.

Solution 2. For each $i \in [n]$ and boolean array A of length 3, define $g(i, A)$ to be the maximum number of apples you can produce on spots 1 to i given that for all $j \in \{1, 2, 3\}$, there is a tree at position $i - j + 1$ if and only if $A[j] = 1$. The final answer is $\max_{A \text{ s.t. } sum(A) \leq 2} f(n, A)$.

The recurrence relation for these subproblems is

$$g(i, A) = x_i \cdot A[1] + \begin{cases} g(i-1, [A[2], A[3], 0]) & sum(A) = 2 \\ \max(g(i-1, [A[2], A[3], 0]), g(i-1, [A[2], A[3], 1])) & sum(A) < 2 \end{cases}$$

Runtime is $O(n)$.

Exam continues on next page

6 Apple Tree (11 points)

There is a binary apple tree T with N nodes. Let r be the root of T. Suppose every non-leaf node has exactly two children. Every branch in this tree has a certain number of apples on it. Every branch is an edge from a parent node to a child node. We use w_e to denote the number of apples on edge e. However, this tree has too many branches, so we have to prune it (remove edges) until it has at most K remaining branches. Describe a dynamic programming algorithm to output the maximum number of apples that remain on the tree after pruning.

1. What are the subproblems? (precise and succinct definition needed)

2. What is the recurrence relation?

3. What is the runtime?

Solution:

(a) **Subproblems:** For node v and $0 \le k \le K$, let $f(v,k)$ be the maximum number of apples that can be kept in the subtree rooted at v (denoted with T_v), if k is the maximum number of branches that are allowed to remain in this subtree after pruning. Our final answer is $f(r,K)$.

(b) **Recurrence and Base Cases:**

$$f(v,k) = \max \begin{cases} \max_{0\le j\le k-2} \left(f(\text{left}(v), j) + f(\text{right}(v), k-2-j) + w_{(v,\text{left}(v))} + w_{(v,\text{right}(v))} \right); \\ f(\text{left}(v), k-1) + w_{(v,\text{left}(v))}; \\ f(\text{right}(v), k-1) + w_{(v,\text{right}(v))}. \end{cases}$$

$f(v,k) = 0$ for all leaf nodes v and $k \ge 0$. $f(u,0) = 0$ for all (non-leaf) nodes u.

(c) **Runtime:** $O(NK^2)$.

Alternative solution:

(a) **Subproblems:** For node v and $0 \le k \le K$, let $f(v,k)$ be the maximum number of apples that can be kept in the subtree rooted at v together with the branch from v to v's parent, if k is the maximum number of branches that are allowed to remain in this subtree (again, including the edge from v to v's parent) after pruning. Suppose there is an edge from root v to an auxiliary node v_0 with weight 0. Our final answer is $f(r, K+1)$.

(b) **Recurrence and Base Cases:**

$$f(v,k) = \max_{0\le i\le k-1} \left(f(\text{left}(v), i) + f(\text{right}(v), k-1-i) \right) + w_{(v,\text{parent}(v))}$$

For nodes v, $f(v,0) = 0$ and $f(v,1) = w_{(v,\text{parent}(v))}$.

(c) **Runtime:** $O(NK^2)$.

Exam continues on next page ⟹

7 Zero Sum Game? (11 points)

Consider the following two player game played on an array of N integers $A[0] \dots A[N-1]$, with an overall score S that is 0 at the start and a fixed integer $M \leq N$.

The players alternate turns with Alice going first. During Alice's move, she will remove at least 1 and at most M elements from the end of the array and add them to S. During Bob's turn, he will also remove at least 1 and at most M elements from the end of the array and **discard them**. The game ends when the array becomes empty. Alice aims to maximize S and Bob aims to minimize S. If both the players play optimally, what will S be at the end of the game? Describe an $O(N^3)$ or faster DP algorithm that outputs this S given the array A and integer M.

1. What are the subproblems? (precise and succinct definition needed)

2. What is the recurrence relation?

3. What is the runtime?

Solution: Observe that the game is symmetric, i.e we can think of it as two players each having a score and they add the elements from the turn to their own scores. So each player's goal is to maximize their score while minimizing the other player's score.

(a) **Subproblems:** Let $dp[i]$ denote the maximum score the first player can obtain by playing this game on the subarray $A[0] \dots A[i-1]$. The final answer is $dp[N]$.

(b) **Recurrence and Base Cases:** Let $p[i]$ denote the sum of the first i elements of A, i.e,

$$p[i] = \sum_{k=0}^{i} A[k] \tag{1}$$

$$dp[i] = \begin{cases} 0 \quad \text{if } i = 0 \\ p[i] - \min\limits_{0 \le j < i} dp[j] \quad \text{if } i < M \\ p[i] - \min\limits_{i-M \le j \le i} dp[j] \quad \text{otherwise} \end{cases} \tag{2}$$

(c) **Runtime:** $p[0] \dots p[N]$ can be computed in $O(N)$ (or naively $O(N^2)$). Taking the min naively gives a $O(N^2)$ algorithm.

Alternate Solution:

(a) **Subproblems:** for $0 \le i \le N$, and $t \in \{0, 1\}$ let $dp[i][t]$ be the optimal value of S that player t (0 for Alice and 1 for Bob) can achieve using the first i elements of the array $A[0], \dots, A[i-1]$ while it is currently player t's turn. The final answer is $dp[N][0] = dp[N][1]$.

(b) **Recurrence and Base Cases:** We set $dp[0][0] = dp[0][1] = 0$ and for any $i < 0$ we set $A[i] = dp[i][t] = 0$. For $i > 0$ the recurrence relation is given by:

$$dp[i][t] = \begin{cases} \max_{1 \le j \le M} dp[i-j][1] + \sum_{k=1}^{j} A[i-k] & t = 0 \\ \min_{1 \le j \le M} dp[i-j][0] & t = 1 \end{cases}$$

(c) **Runtime:** There are $2N$ states here and each state takes $O(M) = O(N)$ time to compute so this recurrence also has a $O(N^2)$ runtime.

Exam continues on next page

8 Round Table (12 points)

There are representatives from m different organizations attending an international conference. The i-th organization has sent r_i representatives. The conference center has n round tables, and the j-th round table can accommodate c_j representatives. To ensure representatives engage effectively, it's preferable that members from the same organization do not sit at the same round table. Please design an efficient algorithm to output a feasible seating arrangement if one exists.

Solution: We provide two valid approaches to this problem:

Solution 1 (Max Flow). We create a bipartite graph with $m + n + 2$ nodes: a source node s, a sink node t, m nodes as organizations, and n nodes as round tables. Create the following edges:

- An edge from s to organization i with capacity r_i, for $i \in [m]$;
- An edge from round table j to sink t with capacity c_j, for $j \in [n]$;
- An edge from organization i to round table j with capacity 1, for all $(i, j) \in [m] \times [n]$.

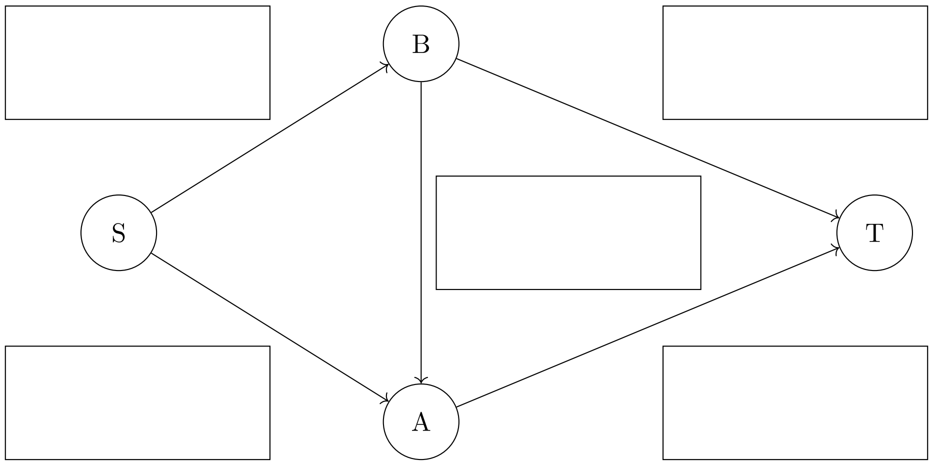
Find the max flow on this graph from s to t. If the max flow equals $\sum_i r_i$, then we have a feasible seating assignment. A feasible seating assignment can be obtained by looking at the fully saturated edges between organizations and round tables; e.g. if the edge from organization i to round table j is fully saturated, then we assign one representative from org i to table j.

Solution 2 (Greedy). Iterate through the m organizations in decreasing order of representative count. For each organization i, assign each of its r_i representatives to the round tables with the most remaining seats, such that each representative from i sits at a distinct table. If this is not possible, output INFEASIBLE. Otherwise, we are able to iterate through all organizations to produce a feasible seating arrangement.

Exam continues on next page $\Longrightarrow$

9 Ford-Fulkerson (9 points)

We saw in class that in a max flow problem with integer capacities, the Ford-Fulkerson algorithm will run for at most U iterations, where U is the value of the maximum flow. In this problem, we will show that there exist graphs where U iterations are required. Consider the following graph $G = (V, E)$:



Label the capacities of the edges of the above graph with integers between 1 and 1,000,000 inclusive so that

1. The max flow U of the graph G is equal to 2,000,000, AND
2. Ford-Fulkerson can take U iterations to compute the maximum flow.

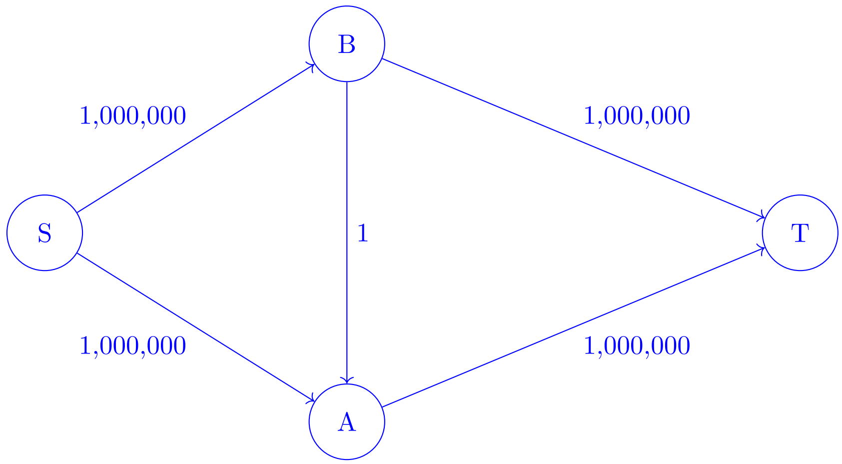
You should assume that the Ford-Fulkerson algorithm sends the maximum amount of flow possible along the augmenting path it finds in each iteration.

Describe a sequence of augmenting paths that would make Ford-Fulkerson take U iterations.

Now suppose that Ford-Fulkerson always selected its augmenting path to be a shortest s-t path in the residual graph. How many iterations would it take? Describe a sequence of augmenting paths it could use.

Solution:

1.



2. Consider alternating between the augmenting paths $S \to B \to A \to T$ and then $S \to A \to B \to T$ a total of two million times. Ford-Fulkerson would send one unit of flow each iteration.

3. If we always select a shortest path, we would saturate $S \to A \to T$ followed by $S \to B \to T$ (or in the other order; it doesn't matter). Thus, Ford-Fulkerson would run for two iterations.

10 Football (8 points)

Brock Purdy has the football and is deciding who to throw it to:

- The running back, gaining 5 yards
- The tight end, gaining 10 yards
- The wide receiver, gaining 15 yards

However, the defense has two defenders, who will each guard a different one of Purdy's options. If Purdy throws to someone who is guarded, the throw will be incomplete and Purdy would gain 0 yards. However, if Purdy throws to someone who is not guarded, they would gain the corresponding amount of yards.

Modelling the situation as a zero-sum game between Brock Purdy (as the row player) and the defense (as the column player), what is the payoff matrix?

Write the linear program for Brock Purdy's optimal strategy.

Solution:

1. Since the defense guards 2 of the 3 options, basically the defense has 3 actions: choose someone to not guard. So, the payoff matrix can be like this:

5	0	0
0	10	0
0	0	15

2. Our objective is

$$\max p$$

subject to

$$\begin{aligned} p &\leq 5x_1 \\ p &\leq 10x_2 \\ p &\leq 15x_3 \\ x_1 + x_2 + x_3 &= 1 \\ x_1, x_2, x_3 &\geq 0 \end{aligned}$$

Solving this LP (not required): we want to set the constraints to equal, so $5x_1 = 10x_2 = 15x_3$. This means we should have the ratio $x_1 : x_2 : x_3 = 1 : 1/2 : 1/3$. Since they sum to 1, we can normalize and divide by the sum to get

$$x_1 = 6/11 \qquad x_2 = 3/11 \qquad x_3 = 2/11$$

which means the value of the game is $30/11 \approx 2.727$.

This page left intentionally blank
for scratch purposes.
This page **will not be graded**.

DO NOT DETACH THIS PAGE.