CS 170 Midterm 2 Solutions

Write in the following boxes clearly and then double check.

Name :	Oski Bear
SID :	
Exam Room :	◯ Dwinelle 145 ◯ Hearst Field Annex A1 ◯ VLSB 2050 ◯ VLSB 2040 ◯ Evans 10 ◯ Other (Specify):
Name of student to your left :	
Name of student to your right :	

- **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 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.
- 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 True or False (12 points)

True or False (3 points each). Include a short justification (1-2 sentences).

1. The optimal value of a maximizing LP problem can be greater than a feasible value of its dual LP problem.

   ○ True ○ False

2. Consider a maximum S-T flow in a directed graph with vertices V and edges E. If we add 1 to the capacity of each edge, then the value of the maximum flow increases by $|E|$.

   ○ True ○ False

3. Consider a maximum S-T flow in a directed graph with vertices V and edges E. If we multiply the capacity of each edge by 2, then the value of the maximum flow is also multiplied by 2.

   ○ True ○ False

4. The optimal multiplicative weights algorithm always guarantees that the total loss is less than or equal to the expert with the highest loss.

   ○ True ○ False

Exam continues on next page

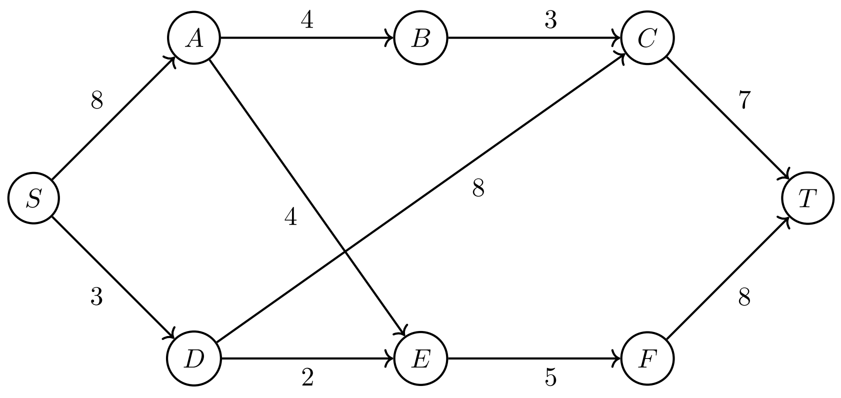
Solution:

1. False. Any feasible value of the dual must be at least the optimal value of the max LP.
2. False. Consider a graph where there are edges that are unused by the maximum flow.
3. True. The max flow multiplied by 2 is a valid flow on the doubled graph, and the residual graph still has disconnected S and T.
4. False. Consider losses $\ell^{(1)} = \begin{bmatrix}0 & 1\end{bmatrix}$ and $\ell^{(2)} = \begin{bmatrix}1 & 0\end{bmatrix}$. Since MWU is probabilistic, we may end up getting a loss of 2 while the worst expert still has a loss of 1.

Exam continues on next page $\Longrightarrow$

2 Maximize the Flow (8 points)

We wish to compute the maximum flow of the below graph from S to T using the Ford Fulkerson algorithm.



1. (4 points) Let's say that the first path from S to T that we find is $S \to A \to B \to C \to T$.

   (a) What is the maximum flow we can send along this path?

   (b) List the following capacities of edges in the residual graph after sending the maximum possible flow along this path.

$S \to A$ $A \to S$

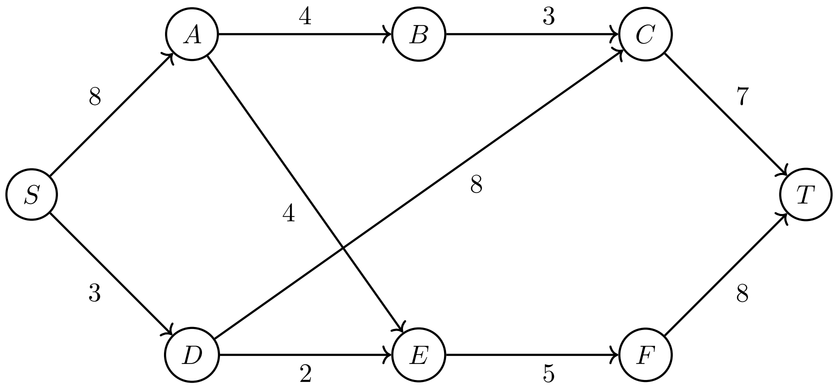
$A \to B$ $B \to A$

$B \to C$ $C \to B$

$C \to T$ $T \to C$

Exam continues on next page

The graph has been redrawn below for your convenience.



2. (2 points) Compute the value of the maximum flow of the graph.

3. (2 points) In the minimum S-T cut of the graph, which vertices are in S's side of the cut? (List them in alphabetical order).

Solution:

1. (a) 3

   (b) .

$S \to A$	5
$A \to B$	1
$B \to C$	0
$C \to T$	4
$T \to C$	3
$C \to B$	3
$B \to A$	3
$A \to S$	3

2. 10

3. A, B, S

Exam continues on next page $\Longrightarrow$

3 Liam's Linear Program (8 points)

Consider the following LP.

$$\begin{aligned} &\max 2x + y \\ \text{subject to } & x - y \geq -4 \\ & 6 - x \geq y \\ & x - 4y \leq 1 \\ & x, y \geq 0 \end{aligned}$$

We have provided a blank graph on the next page if you need it, but **we will not grade anything written on the graph**.

1. (4 points) What is the optimal x, y and objective? Answer in the boxes below.

   x^*: , y^*:

   Optimal objective value:

2. (4 points) Now complete the dual of this LP in its canonical form.

   ○ max ○ min $4a + 6b + c$

   subject to

   $a +$ $b +$ $c \geq 2$

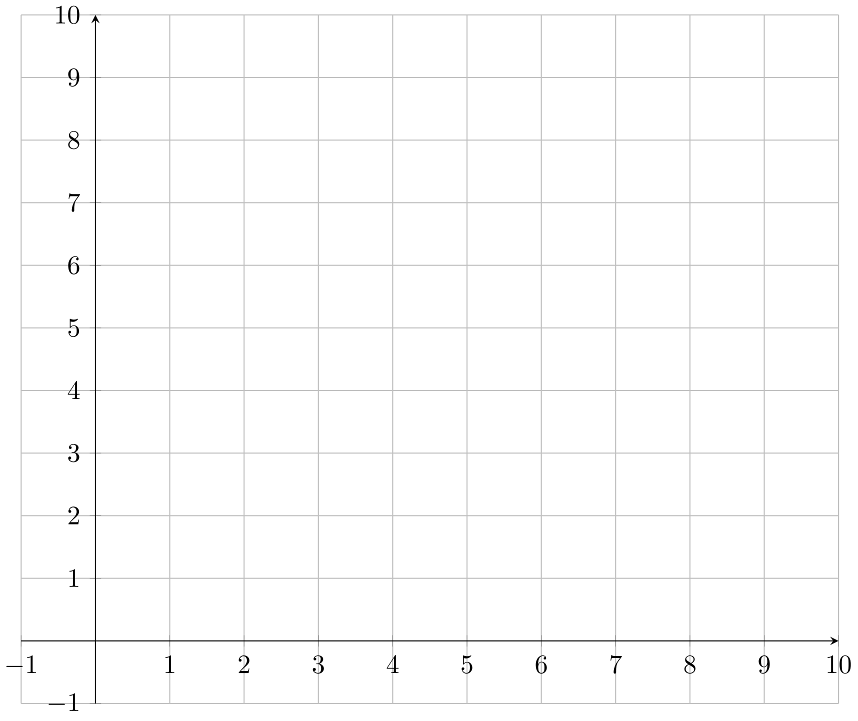
   $a +$ $b +$ $c \geq 1$

   $a, b, c \geq 0$

   Optimal objective value of the dual:

Exam continues on next page ⟹

As a reminder, you may use this graph for scratch work, but it will not be graded.



Solution: The vertices are $(x, y) = (0, 0), (1, 0), (5, 1), (1, 5), (0, 4)$. The values of the objective function at those points are $0, 2, 11, 7, 4$, which means that the optimal x, y and objective are $x^* = 5, y^* = 1$, and objective $= 11$.

For the dual, we want to minimize $4a + 6b + c$. Rewriting the original constraints, they are

$$\begin{aligned} -x + y &\leq 4 \\ x + y &\leq 6 \\ x - 4y &\leq 1 \end{aligned}$$

Therefore, the constraints of the dual are

$$\begin{aligned} -a + b + c &\geq 2 \\ a + b - 4c &\geq 1 \end{aligned}$$

The optimal value is the same, which means it is 11.

The solution to the above LP is

$$(a, b, c) = \left(0, \frac{9}{5}, \frac{1}{5}\right)$$

(although students need not be able to solve this).

Exam continues on next page $\Longrightarrow$

4 Pokemon Battle 0 (10 points)

Jonathan has 3 pokemon and Ajit has 2 pokemon. They will each choose just 1 of their pokemon. If Jonathan chooses his i-th pokemon and Ajit chooses his j-th pokemon, then Jonathan gets a score of s_{ij} (could be negative). Jonathan would like to maximize his score and Ajit would like to minimize Jonathan's score. Given all of the values of s_{ij}, please write out the LP formulation for both Ajit and Jonathan. Use a and b for the probabilities for Ajit's strategy and x, y, and z for the probabilities for Jonathan's strategy. **You do not have to solve the LPs**. In the table below, Jonathan is the row player while Ajit is the column player.

$$S_{ij} = \begin{bmatrix} 3 & -3 \\ 0 & 3 \\ -2 & 7 \end{bmatrix}$$

Ajit:

Jonathan:

Exam continues on next page

Solution: For Ajit's strategy, he chooses the first column with probability a and the second column with probability b. Then, Jonathan will choose the row that maximizes his EV, which is

$$\max(3a - 3b, 3b, -2a + 7b)$$

So, Ajit wants to minimize this value. We can add a dummy variable to represent the max, call it c. So, the LP is

$$\min c$$

subject to

$$\begin{aligned} c &\geq 3a - 3b \\ c &\geq 3b \\ c &\geq -2a + 7b \\ a + b &= 1 \\ a &\geq 0 \\ b &\geq 0 \end{aligned}$$

If we wanted just inequalities we could use $a + b \geq 1$ and $a + b \leq 1$ or we could have just substitutde $b = 1 - a$.

For Jonathan's strategy, he chooses the first row with probability x, the second with probability y, and the third with probability z. Then, Ajit will choose the column that minimize the expected score, which is

$$\min(3x - 2z, -3x + 3y + 7z)$$

So, Jonathan wants to maximize this value. We can add a dummy variable to represent the min, call it t. So, the LP is

$$\max t$$

subject to

$$\begin{aligned} t &\leq 3x - 2z \\ t &\leq -3x + 3y + 7z \\ a + b + c &= 1 \\ a, b, c &\geq 0 \end{aligned}$$

Exam continues on next page $\Longrightarrow$

5 Pokemon Battle 1 (16 points)

Jonathan and Ajit each have n pokemon. They are playing a game that takes n rounds. Each round, they each choose 1 of their remaining pokemon (one that hasn't been chosen before). If Jonathan chooses his i-th pokemon and Ajit chooses his j-th pokemon, then Jonathan gets a score of s_{ij} (could be negative). However, Jonathan knows that Ajit will just play his pokemon in order (i.e. play pokemon 1 in round 1, pokemon 2 in round 2, etc.). Given this information, please describe a dynamic programming algorithm to help Jonathan determine the order to play his pokemon to maximize his total score. Your algorithm should run in time $O(n2^n)$.

For this problem, write your answer in the following 3 part format (you do not need to prove that the recurrence is correct):

(a) Define a function $f(\cdot)$ in words, including how many parameters are and what they mean, and tell us what inputs you feed into f to get the answer to your problem.

(b) Write the "base cases" along with a recurrence relation for f.

Exam continues on next page

(c) Analyze the runtime and space complexity of your final DP algorithm.

Solution: We can use dynamic programming to make our algorithm faster than $n!$. Our DP keeps track of the maximum score Jonathan can get while using a particular subset S of his pokemon (in the first $|S|$ battles). Our recurrence is

$$dp[S] = \max_{v \in S} dp[S \setminus \{v\}] + s_{v|S|}$$

There are 2^n states, and each state takes $O(n)$ computation, so the overall runtime is $O(n2^n)$. The final answer is $dp[\{1, 2, \ldots, n\}]$ and the initial state is $dp[\emptyset] = 0$.

Note: A $O(n^3)$ solution exists using the Hungarian algorithm

Exam continues on next page $\Longrightarrow$

6 Migration (16 points)

A group of k PNPenguins are currently in Guinland but have decided to move to Hawaii, using m self-driving sailboats that travel between n islands (Guinland and Hawaii are both islands).

Each island can host unlimited number of penguins, but sailboat i can only have c_i penguins onboard. Each sailboat has a list of destinations d_i which it will travel through each of them repeatedly: for example, $d_3 = [1, 5, 6]$ means sailboat 3 will stop at island 1 on day 0, island 5 on day 1, island 6 on day 2, island 1 on day 3, island 5 on day 4, island 6 on day 5, and so forth. Each sailboat takes 1 day to travel from any island to any other island. Penguins can only embark and disemembark when the sailboat is at an island.

Given m, n, c_i and d_i, your job determine whether or not it is possible to transport all of the penguins from Guinland to Hawaii within 170 days.

Model this problem as a flow network. Specify the vertices, edges, and capacities; show that a maximum flow in your network can be transformed into an optimal solution for the original problem. You do not need to explain how to solve the max-flow instance itself.

Exam continues on next page

Exam continues on next page ⟹

Solution: To test if we can transport penguins with d days, we build a graph with $(d+1)n+2$ vertices where 2 vertices are s and t, and each vertex $v_{i,t}$ means the island (including Guinland and Hawaii) at day t, $0 \leq t \leq d$. We make edges $(v_{i,t}, v_{i,t+1})$ with capacity infinity (as infinite number of penguins can stay on the island for a day) and for each consecutive islands p, q in d_j, given that sailboat j will be at island p at day t, we make edges $(v_{p,t}, v_{q,t+1})$ with capacity c_j (as sailboat j has capacity c_j to transport penguins from island p on day t to q on day $t+1$).

Finally, make edges $(s, v_{0,0})$ and $(v_{n+1,d}, t)$ with infinite (or $\geq k$) capacity. Run Edmonds-Karp and check if the max-flow is greater than or equal to k.

We run Edmonds-Karp on a graph that has $O(170n)$ vertices and $O(170m)$ edges. Since 170 is a constant, the runtime is just nm^2.

Exam continues on next page $\Longrightarrow$

7 Alarming Knapsack (17 points)

During a robbery, a burglar finds much more loot than he had expected and has to decide what to take. His bag (or knapsack) will hold a total weight of at most W pounds ($W > 0$). There are n items to pick from, of positive weight $w_1, \ldots, w_n$, and positive dollar value $v_1, \ldots, v_n$. However, if the burglar takes 4 items in a row (i.e. the burglar takes items $k, k+1, k+2$, and $k+3$ for some k), then an alarm will trigger and the burglar will get caught. The burglar can first look at all of the items and then decide which ones to take. What is the dollar value of the most valuable combination of items he can fit into his bag without taking 4 items in a row?

For this problem, write your answer in the following 4 part format:

(a) Define a function $f(\cdot)$ in words, including how many parameters are and what they mean, and tell us what inputs you feed into f to get the answer to your problem.

(b) Write the "base cases" along with a recurrence relation for f.

Exam continues on next page

(c) Prove that the recurrence correctly solves the problem.

(d) Analyze the runtime and space complexity of your final DP algorithm.

Exam continues on next page ⟹

Solution: This problem is similar to knapsack without repetition. To ensure that we don't take more than 4 items in a row, we have to modify the DP algorithm to also store the number of items in a row that has been taken. Therefore, the subproblem definition will be $dp[w, i, j]$ = the maximum value achievable with weight w, the first i items, and taking each of the last j items in a row starting at index i.

The algorithm should return $\max_{k \in \{0,1,2,3\}} dp[W, n, k]$

The base cases are
$dp[0, 0, 0] = 0$
$dp[0, 0, j] = -\infty \ \forall 1 \leq j \leq 3$

The recurrence relation is

$$dp[w, i, j] = \begin{cases} \max_{k \in \{0,1,2,3\}} dp[w, i-1, k] & j = 0 \\ v_i + dp[w - w_i, i-1, j-1] & 1 \leq j \leq 3 \end{cases}$$

Proof:
Base cases: The base case $dp[0, 0, 0] = 0$ is trivially true since we can achieve a value of 0 with weight 0, 0 items, and 0 items in a row by just not taking anything. The base case $dp[0, 0, j] = -\infty \ \forall 1 \leq j \leq 3$ is true since it's not possible to take the previous j items in a row, if you're not considering any of the items yet.

Inductive hypothesis: Assume that all subproblems $dp[w - a, i - b, j - c]$ are correctly calculated where $a + b + c \geq 1$.

Inductive step: If $j = 0$, we are not using the current item i. Therefore, we only need to consider the maximum subproblem involving the same weight, first $i - 1$ items, and any valid number of items chosen in a row. This gives us the correct value for $dp[w, i, 0]$. Otherwise if $j \geq 1$, we pick the current item, so we add the value of the current item v_i to the knapsack. The remaining capacity of the knapsack is $w - w_i$, and we must have taken the previous $j - 1$ items in a row (starting at index $i - 1$), so that at index i, we will have picked j items in a row. Therefore, this correctly calculates $dp[w, i, j]$.

Runtime:
We have $W \cdot n \cdot 4 = 4nW$ entries in our table, and each one takes constant time to run. So therefore, the overall runtime is $O(nW)$ which is the same as regular knapsack.

Space Complexity:
We have $O(nW)$ subproblems so the memory complexity is $O(nW)$. Using a bottom up implementation, the memory complexity can be reduced to $O(W)$. Both answers are accepted.

Exam continues on next page $\Longrightarrow$

8 Param's Paper (13 points)

1. (3 points) Suppose we run Multiplicative Weight Updates with 3 experts and $\epsilon = \frac{1}{2}$. The losses are

$$\ell^{(1)} = \begin{bmatrix} \frac{1}{2} & \frac{2}{3} & 0 \end{bmatrix}$$
$$\ell^{(2)} = \begin{bmatrix} \frac{1}{2} & \frac{1}{3} & 1 \end{bmatrix}$$

Here, the i-th element of $\ell^{(1)}$ denotes the loss of the i-th expert in the first iteration, and the i-th element of $\ell^{(2)}$ denotes the loss of the i-th expert in the second iteration. As a reminder, we initialize the weights of each expert to 1 and our update rule is

$$w_i^{(t+1)} = w_i^{(t)} \cdot (1-\epsilon)^{\ell_i^{(t)}}$$

(a) Compute $w_i^{(1)}$ for $i = 1, 2, 3$.

$w_1^{(1)}$ $w_2^{(1)}$ $w_3^{(1)}$

(b) Compute $w_i^{(2)}$ for $i = 1, 2, 3$.

$w_1^{(2)}$ $w_2^{(2)}$ $w_3^{(2)}$

(c) Compute the probability that the 1st expert is chosen on the 3rd iteration.

Exam continues on next page

2. (10 points) Param is playing k games of Rock Paper Scissors. Each game, Param must decide whether to play a rock, paper, or scissors. Thankfully, Param can listen to the advice of N experts, each of whom will tell Param which action to take for each game (it is up to Param to decide which one(s) to listen to).

   Param decides that that he will use a modified version of the MWU algorithm. He initializes all of the weights of each expert to 1. Every turn he chooses the action that maximizes the sum of the weights of the experts voting for that action. After each game, Param will multiply the weights of the experts that got the action wrong by half. Param would like to do as well as the best expert. If the best expert was wrong M times, show that the number of times Param is wrong at most $c \cdot (M + \log_2 N)$ for some constant c, and compute c.

   (a) First, show that if Param is wrong, then the sum of the weights of the experts is reduced from W to at most $\frac{3}{4}W$.

   (b) Second, show that if the best expert makes M mistakes, then the sum of the weights of the experts at the end is at least $\left(\frac{1}{2}\right)^M$.

Exam continues on next page

(c) Finally, show that the number of times Param is wrong is at most $c \cdot (M + \log_2 N)$ for some constant c, and compute c. You may use the fact that

$$\log_a b = \frac{\log_x b}{\log_x a}$$

c:

Exam continues on next page $\Longrightarrow$

Solution:

1. After the first iteration the weights are

$$w^{(1)} = \begin{bmatrix} \left(\frac{1}{2}\right)^{\frac{1}{2}} & \left(\frac{1}{2}\right)^{\frac{2}{3}} & 1 \end{bmatrix}$$

   After the second iteration the weights are

$$w^{(2)} = \begin{bmatrix} \left(\frac{1}{2}\right)^{1} & \left(\frac{1}{2}\right)^{1} & \left(\frac{1}{2}\right)^{1} \end{bmatrix}$$

   Therefore, the probability that the 1st expert is chosen is simply $\frac{1}{3}$.

2. Each iteration that we are wrong, we end up reducing the sum of the weights of all the experts from W to at most $\frac{3}{4} \cdot W$. This is because at most $\frac{1}{2}$ of the experts are right, which means at least $\frac{1}{2}$ of the experts are wrong, and we reduce their weights by $\frac{1}{2}$. Therefore, if we make L mistakes then the sum of the weights at the end is at most $\left(\frac{3}{4}\right)^L \cdot N$. Additionally, if the best expert was wrong M times, that means their weight at the end was $\left(\frac{1}{2}\right)^M$, which means the sum of the weights of the experts was at least $\left(\frac{1}{2}\right)^M$ since all other weights are nonnegative. Therefore, we have

$$\left(\frac{1}{2}\right)^M \leq N \left(\frac{3}{4}\right)^L$$

   Solving for L, we find that

$$M \log_{4/3} \frac{1}{2} \leq \log_{4/3} N - L$$

$$L \leq \log_{4/3} N + M \log_{4/3} 2 = \log_{4/3} 2(M + \log_2 N)$$

   Here we have used the fact that

$$\frac{\log_{4/3} N}{\log_{4/3} 2} = \log_2 N$$

Blank scratch page.
This page **will not be graded**.