CSCI-570 Fall 2023

Practice Exam 2

INSTRUCTIONS

- The duration of the exam is 140 minutes, closed book and notes.
- No space other than the pages on the exam booklet will be scanned for grading!
- Please note that most of the questions on this exam do not require lengthy responses; we advise you to fit your answers within the space provided for each question.
- If you require an additional page for a question, you can use the extra page provided at the end of this booklet. However please indicate clearly that you are continuing the solution on the additional page.

1. True/False Questions

Mark the following statements as **T** or **F**. No need to provide any justification.

a) **(T/F)** If $\sum_v d_v \neq 0$, then there is no feasible circulation in the graph for the set of demands $\{d_v\}$.

True. For there to be a feasible circulation, $\sum_v d_v = 0$ and value of flow $|f|$ should be equal to total demand value D.

b) **(T/F)** Given a graph G and a total demand value of D if the absolute value of the max flow $|f|$ is less than D then a feasible circulation exists in G.

False. If value of flow $|f|$ is less than D, then the network doesn't have enough capacity to support a feasible circulation.

c) **(T/F)** Suppose the maximum (s,t)-flow of some graph has value *f*. Now we increase the capacity of every edge by 1. Then the maximum (s,t)-flow in this modified graph will have value at most *f+1*.

False. See the figure 0.1 given below, increasing edge capacities by 1 would increase max-flow by 2.

d) **(T/F)**. For P and D, a primal-dual pair of Linear Programming problems, if the primal problem P has a bounded optimal solution, then the dual problem D also has a bounded optimal solution.

True. Strong Duality.

e) **(T/F)**. Both the primal and the dual problems in the context of Linear Programming (LP) can be unbounded.

False. Only one can be unbounded.

f) **(T/F)**. Problem X is in NP. And if it can be solved in polynomial time, then P = NP.

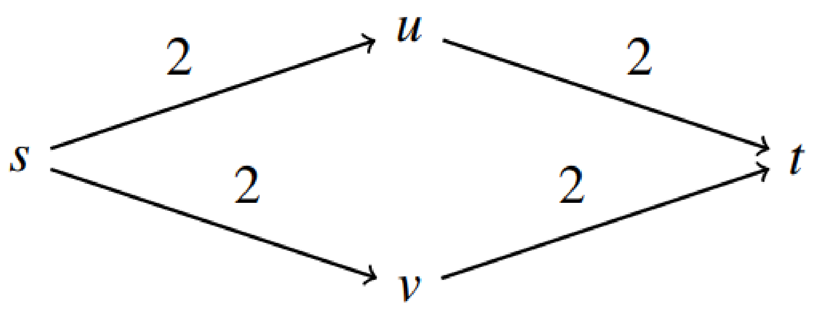


Figure 0.1: Example Graph

False. Since if X is a problem in P, then it can't get the result of P=NP.

g) **(T/F)**. If all edges in a graph have capacity 1, then Ford-Fulkerson runs in linear time.

False

h) **(T/F)**. Every optimization problem has an equivalent decision problem.

True: We can always ask if the optimum value is greater/lesser than some estimate.

i) **(T/F)**. Every problem in NP can be solved in polynomial time by a nondeterministic Turing machine.

True: this is the definition of NP

j) **(T/F)**. The Integer Linear Programming is in NP.

False: checking the feasibility of each assignment takes polynomial time, however computing the optimal value of the objective function is not a polynomial.

2. Multiple Choice Questions

Please select the most appropriate choice. Each multiple choice question has a single correct answer.

a) In the Ford-Fulkerson Algorithm for network flow, the residual capacity of an edge (u, v) in a residual graph is defined as:

   a) $\min(c(u,v), f(u,v))$
   b) $c(u,v) + f(u,v)$
   c) $c(u,v) - f(u,v)$
   d) $|c(u,v) - f(u,v)|$

   Where c(u,v) is the capacity of edge u, v and f(u, v) is the current flow.

   Answer: c) $c(u,v) - f(u,v)$

   Explanation: The residual capacity of an edge in the residual graph represents the remaining capacity for flow in the original graph. Mathematically, it is defined as the difference between the edge's capacity ($c(u,v)$) and the current flow ($f(u,v)$), denoted as $c(u,v) - f(u,v)$.

b) Problem X is in NP. Then problem X is NP-Complete if

   a) Y is some problem in NP, and Y can be reduced to X in polynomial time
   b) Y is some problem in NP, and X can be reduced to Y in polynomial time
   c) X can be reduced to the independent set problem in polynomial time
   d) The independent set problem can be reduced to X in polynomial time

   (d) is correct. For (a)(b), if Y is in P (also in NP), then X may be not NP-Complete. For (c), X can be some problem in P.

c) For the following statements, which of them is true?

   a) NP-Hard is a subset of NP-Complete
   b) X is a known NP-Complete problem, if we can reduce X to another problem Y in polynomial time, then Y must be in the set of NP-Hard problems
   c) The 3-SAT problem remains NP-Complete even when each clause is restricted to contain exactly one literal.

   (b) is correct. Because X is NP-Hard, so Y must also be NP-Hard.
   (a) is incorrect since NP-Hard is a subset of NP-Complete.
   (c) is incorrect since the circuit satisfiability problem is the first problem that was proved as NP-Complete

d) Which of the following statements is correct?

   a) If f is a max flow of a flow network G with source s and sink t, then the value of the min cut in the residual graph G_f is 0.
   b) Let (S, T) be a minimum cut of flow network G, if we increase the capacity of every edge by 1, then (S, T) will still be a minimum cut.
   c) If in a flow network all edge capacities are distinct, then there exists a unique min-cut.
   d) If you are given a maximum flow in a graph then you can find a minimum cut in time $O(\log m)$.

   a

e) What is the role of the objective function in a linear programming problem?

   a) To define the feasible region.
   b) To set the constraints.
   c) To measure the profit or cost of a decision.

d) To set the decision variables.

c. The goal of the problem is to find the optimal values of the decision variables that satisfy the constraints and optimize the objective function.

f) Which of the following statements is correct ?

a) Every LP problem has at least one optimal solution.
b) Every LP problem has a unique solution.
c) If an LP problem has two optimal solutions, then it has infinitely many solutions.
d) If a feasible region is unbounded then LP problem has no solution

c

3. **Dynamic Programming** In share trading, a buyer buys shares and sells on a future date. Given the stock price of n days, the trader is allowed to make at most k transactions, where a new transaction can only start after the previous transaction is complete, find out the maximum profit that a share trader could have made. Examples:

   a) Input: Price = [10, 22, 5, 75, 65, 80]. K = 2. Output: 87
      Trader earns 87 as sum of 12 and 75. Buy at price 10, sell at 22, buy at 5 and sell at 80.

   b) Input: Price = [12, 14, 17, 10, 14, 13, 12, 15]. K = 3. Output: 12
      Trader earns 12 as the sum of 5, 4 and 3. Buy at price 12, sell at 17, buy at 10 and sell at 14 and buy at 12 and sell at 15.

   a) Define (in plain English) subproblems to be solved. (1 points)

      Let profit[t][i] represent maximum profit using at most t transactions up to day i (including day i).

      Rubrics (1 points):

      - Completely irrelevant definition: -1 points
      - Any alternate definition is eligible for full credits.

   b) Write a recurrence relation for the subproblems and provide an explanation for it. (3 points)

      profit[t][i] = max(profit[t][i-1], max(price[i] – price[j] + profit[t-1][j])) for all j in range [0, i-1]

      Explanation: Profit[t][i] will be maximum of –

      i. profit[t][i-1] which represents not doing any transaction on the ith day.

ii. Maximum profit gained by selling on ith day. In order to sell shares on ith day, we need to purchase it on any one of [0, i – 1] days. If we buy shares on jth day and sell it on ith day, max profit will be price[i] – price[j] + profit[t-1][j] where j varies from 0 to i-1. Here profit[t-1][j] is best we could have done with one less transaction till jth day.

Rubrics (2 points):

- Completely irrelevant recurrence relation: -2 points
- Any alternate recurrence relation is eligible for full credits.

c) Using the recurrence formula in part b, write pseudocode to compute the maximum profit using at most k transactions over n days. (2 points)

- Initialize for base cases (mentioned below for clarity)
- For $i = 0$ to n
  for $j = 0$ to k: if not base case, call recursion
- Return ans (mentioned below for clarity)

Rubrics (2 points):

- Base case and final answer are not required to be reflected in pseudocode, only for loop is expected
- Completely irrelevant pseudocode: -2 points
- Any alternate pseudocode is eligible for full credits.

d) Make sure you specify base cases and their values; where the final answer can be found. (2 points)

There are two possible base cases:

- profit[i][0] = 0;
- or dp[0][j]=0

the final answer - dp[k][n]

Rubrics (2 points):

- Either of the two base cases can get full points
- Wrong/missing base values: -1 point
- Wrong/missing final answer: -1 point

e) What is the space and time complexity of your solution? Explain your answer. (2 point)

Space complexity: O(kn). Time complexity: O(kn^2).

Rubrics (1 point):

- Wrong/missing time complexity or explanation: -1 point
- Wrong/missing space complexity or explanation: -1 point

4. Linear Programming

A company produces three products, Product X, Product Y, and Product Z. The company has a limited amount of resources, including production time, labor, and materials. The goal is to maximize profits subject to resource constraints. The following information is available:

- Each unit of Product X requires 1 hour of production time, 1 hour of labor, and 2 units of raw materials. Each unit of Product Y requires 2 hours of production time, 2 hours of labor, and 3 units of raw materials.
- Each unit of Product Z requires 3 hours of production time, 4 hours of labor, and 2 units of raw materials.
- The company has 200 hours of production time, 250 hours of labor, and 150 units of raw materials available.
- The profit per unit of Product X is \$50, the profit per unit of Product Y is \$60, and the profit per unit of Product Z is \$80.
- The company must produce at least 50 units of Product X, 75 units of Product Y, and 25 units of Product Z. The company can only produce a maximum of 125 units in total.

Formulate a linear programming model to help the company maximize profits. You do not have to solve the resulting LP.

a) Describe what your LP variables represent (3 points).

b) Show your objective function (3 points).

c) Show your constraints (9 points).

solution:

a) Let x be the number of units of Product X to produce, y be the number of units of Product Y to produce, and z be the number of units of Product Z to produce.

b) The objective function to maximize profits can be expressed as:

$$Maximize \qquad 50x + 60y + 80z$$

c) The constraints are as follows:

$$\begin{gathered}
x + 2y + 3z \leq 200 \text{ (production time constraint)} \\
x + 2y + 4z \leq 250 \text{ (labor constraint)} \\
2x + 3y + 2z \leq 150 \text{ (raw materials constraint)} \\
x \geq 50 \text{ (minimum production of Product X)} \\
y \geq 75 \text{ (minimum production of Product Y)} \\
z \geq 25 \text{ (minimum production of Product Z)} \\
x + y + z \leq 125 \text{ (total production capacity constraint)} \\
x, y, z \geq 0 \text{ (non-negativity constraint)}
\end{gathered}$$

5. Network Flow

You are employed at a video game company, which is preparing to release n distinct types of video games. Before their release, these games need to be tested. There are m players available for testing, where $m \geq n$. Each game type appeals to a specific subset of these players. The goal is to distribute these n game types among the players for testing, adhering to the following conditions:

a) Each game type should only be given to players who have shown interest in that specific type.

b) A player can test a maximum of k different game types.

c) No player should receive the same game type more than once.

d) The objective is to maximize the total number of games distributed for testing.

Design an efficient network flow-based algorithm for this problem. You are required to make a claim regarding the correctness of your algorithm and provide proofs in both directions to support your claim.

Solution: Construct a flow graph as follows:

Let s and t be the source and sink vertices respectively.

For each player, introduce a vertex. Call these $p_1, p_2, ..., p_m$. Likewise, for each video game introduce a vertex. Call these $g_1, g_2, ..g_n$.

There is an edge from s to each of the games and it has infinite capacity. That is $c(s, g_i) = \infty$, for $1 \leq i \leq n$.

There is an edge from g_i to p_j if and only if the player p_j is interested in the game g_i. Let its capacity be 1. T here is an edge from every player to the sink t of capacity k. That is $c(b_j, t) = k$, for $1 \leq j \leq m$.

Claim: We can maximize the total number of video games given if and only if we maximize the flow in graph G.

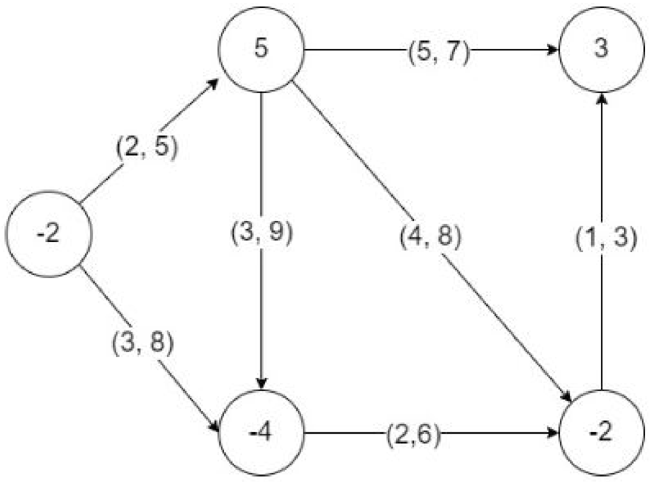
Correctness Proof:

We show this in two steps. First, we show that any flow translates to a valid solution to the video game distribution problem. Then we show that maximizing flow maximizes the number of games distributed.
Assume that we have an integral flow f assigned to the graph. Send $f(g_i, p_j)$ number of g_i game to player p_j.
The flow $f(g_i, p_j)$ can be greater than zero only if the there is an edge between g_i and p_j. Thus players are only given games they are interested in. Further $f(g_i, p_j) \leq 1$, thus no one is given any duplicate games. The flow flowing into p_j corresponds to the number of games received by the player p_j. From the flow conservation constraint at p_j, this equals the flow going out of p_j which is bounded by k. Thus no player received more than k games. From the conservation constraint at g_i, the flow assigned to the edge $e(s, g_i)$ is the total number of games of type g_i sent. We have thus met all the constraints of the game distribution problem, and the flow assignments translates to a valid solution.

Similarly, we can show that any valid solution to the distribution problem can be mapped to a valid flow for the graph. The flow out of the source s is nothing but the total number of games distributed. Thus we indeed maximize the total number of games

6. **Network Circulation**

In the network below, the demand values are shown on vertices. Lower bounds on flow and edge capacities are shown as (lower bound, capacity) for each edge. Determine if there is a feasible circulation in this graph. You need to show all the steps.



Solution: Remove lower bounds by changing demands on each vertex and update the capacity of the edges

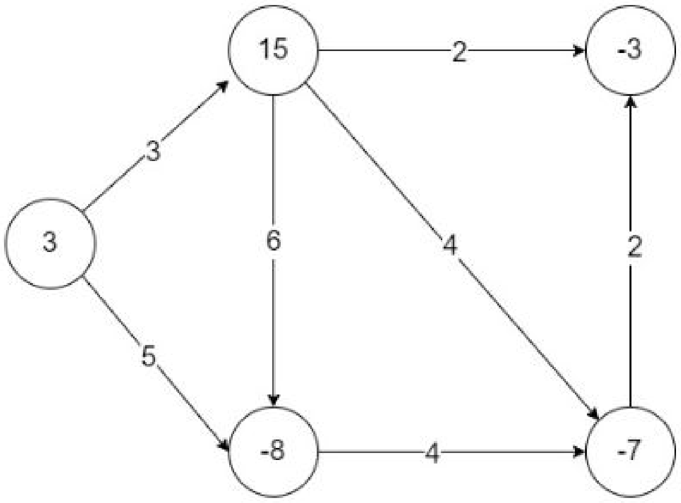
Reduce it to max-flow by adding s and t vertices

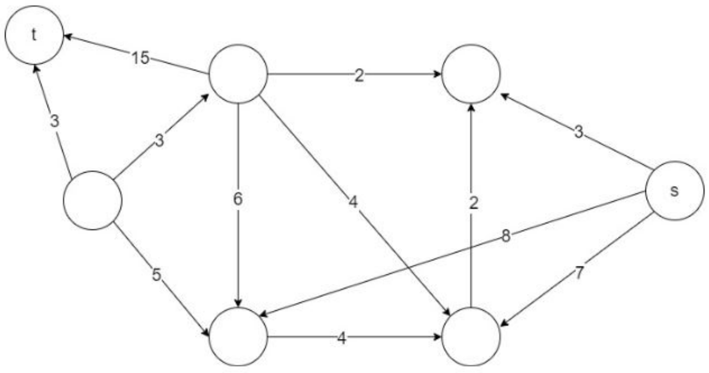
Check if there is an s-t flow $|f| = 18$

There is no such flow. Therefore there is no feasible circulation.

Rubrics:

1. Remove lower bounds and update the capacity of the edges (7 points)
2. Reduce it to max-flow by adding s and t vertices (5 points)
3. Calculate the s-t flow (5 points)





4. Deducing there is no feasible circulation (3 points)

7. NP-Completeness Prove that finding the shortest simple path in an undirected graph with negative cost edges is NP-hard.
Hint: You can use HAM-Path or HAM-Cycle for your reduction.

Refer to the problem 'Shortest simple path w/ negative cost edges' as SSPN. Proof using HAM-PATH.

a) We will show that HAM-PATH is $\leq_p$ SSPN

b) Given an instance of the HAM-PATH problem on graph G = (V, E)

c) Construct G' with the same exact set of nodes and edges as in G with all edges having a weight/cost of -1

d) Claim: G has a HAM Path if and only if SSPN instance has a path of cost $-(|V| - 1)$ between two arbitrary nodes. (Need to call the blackbox $O(n^2)$ times each time using a different pair of starting and ending nodes)

e) Proof: If there is a path of length $-(|V|-1)$ between two nodes s and t in G' this will give us a HAM Path in G. If there is a HAM Path in G, there will be two nodes s and t in G' (the two ends of the HAM Path in G) where cost of their shortest path will be $-(|V| - 1)$

Proof using HAM Cycle. We will show that HAM Cycle p Shortest simple path w/ negative cost edges

a) Given an instance of the HAM Cycle problem on graph G = (V, E)

b) Construct G' with the same exact set of nodes and edges as in G with all edges having a weight/cost of -1

c) Repeat for all edges: Remove edge (uv) in G', add two new nodes v' and u', and add edges uu' and vv' with weights -1.

d) Claim: G has a HAM-Cycle if and only if SP w/ negative cost edge has a path of length $-(|V|+1)$ between v'u'.

e) Proof: If there is a path of length $-(|V|+1)$ between v'u' in G' this will give us a HAM Path from v to u in G which we can turn into a HAM Cycle by adding edge vu in G. If there is a HAM Cycle in G, there will be two nodes v and u in G' (the two ends of one of the edges on the HM Cycle) where the cost of the shortest path between them will be $-(|V|+1)$