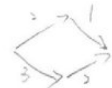
A002FA5B-7682-4D8B-AE6C-32D7C3CD44A4

csci570-fa19-exam3

#493 2 of 10



Q1 18

1) 20 pts
Mark the following statements as **TRUE** or **FALSE, or UNKNOWN** (if unknown is given as a choice). No need to provide any justification.

For the next four questions consider a flow network G, increase the capacity of an edge e by an amount x>0 to obtain Network G'. Let f, f' denote the value of max flow in G, G'. Then,

a) **[TRUE/FALSE]** f' - f is either 0 or x.

b) **[TRUE/FALSE]** If there is a min-cut in G containing e and f'>f, then there's a min-cut in G' containing e.

c) **[TRUE/FALSE]** If f' = f, there is no min-cut in G' containing e

d) **[TRUE/FALSE]** If f' = f, there is no min-cut in G containing e.

[TRUE/FALSE/UNKNOWN]
Let S denote the set of problems reducible to problem X, where X is NP-hard. Then every problem in S must be in NP. $S \le_p X$

[TRUE/FALSE/UNKNOWN]
A general 3-SAT problem cannot be solved in polynomial time.

[TRUE/FALSE/UNKNOWN]
If a general integer programming optimization problem can be reduced in polynomial time to a linear programming problem, then P = NP.

[TRUE/FALSE/UNKNOWN]
If a problem is both in NP and NP-hard, then this problem is NP-complete.

[TRUE/FALSE]
A matching in a bipartite graph G=(V, E) can be tested in O(|V|+|E|) time to determine if it is a maximum size matching. $O(mn)$

[TRUE/FALSE]
For a graph with all edges having distinct weights there is a unique MST.

BD1F0044-21EE-469F-B54E-3C19C38E658E

csci570-fa19-exam3

#493 3 of 10

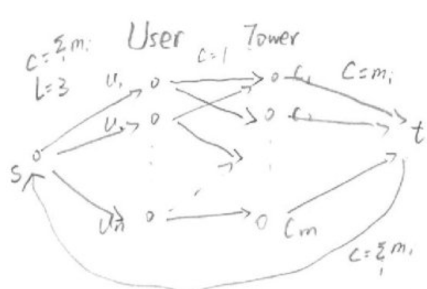
Q2 15

2) 15 pts

You are given a collection of n points $U=\{u_1,\ldots,u_n\}$ in the plane, each of which is the location of a cell-phone user. You are also given the locations of m cell-phone towers, $C=\{c_1,\ldots,c_m\}$. A cell-phone user can connect to a tower if it is within distance Δ of the tower. To guarantee reliability in the network each cell-phone user must be connected to at least three different towers. For each tower c_i you are given the maximum number of users (m_i) that can connect to this tower.

Give a polynomial time algorithm, which determines whether it is possible to assign all the cell-phone users to towers, subject to these constraints. Prove its correctness. (You may assume you have a function that returns the distance between any two points in O(1) time.)

Hint: Use network flow.



Construct a circulation network $G=(V,E)$.

Each user is represented by one node and each tower is represented by one node as well.

Add an edge from u_i to c_j if the distance between them is within Δ. This can be done in $O(mn)$.

Set capacity to 1 for these edges.

Add a source node and a sink node and connect s to each u_i with capacity $\sum_i m_i$ and lowerbound $l=3$

Connect each c_j to t with capacity m_i.

Link t to s with capacity $\sum_i m_i$ and set all the node with demand value 0.

Use Circulation with Lowerbound to find one feasible circulation if exists.

Since this can be done in polynomial, this algorithm is polynomial.

Claim: There exist an assignment with users and towers if only if the circulation has a feasible circulation

Proof: If there is an assignment, I can assign (s, u_i) with flow value of the no. of tower served assign (u_i, c_j) with value 1 if u_i is served by tower c_j. and assign (c_j, t) with flow value of no. of users c_j serves. This can give me a feasible circulation

If there is a feasible circulation, I can follow the above method to find an assignment between users and towers.

66ED8D9E-20DD-45E1-B258-AA61133C1BBE

csci570-fa19-exam3

#493 4 of 10

Q3 15

3) 15 pts

Given n (positive) integers $x_1, x_2, \ldots, x_n$. The Partition problem asks if there is a subset $S \subseteq [n]$ such that:

$$\sum_{i \in S} x_i = \sum_{i \notin S} x_i$$

It can be proven that the Partition problem is NP-complete. Now assuming that the Partition problem is NP-complete, show that the Bin Covering problem is also NP-complete:

Bin Covering: Given a set of n items with sizes $s_1, s_2, \ldots, s_n$, a requirement R, and an integer k. Can the items be placed into at least k bins such that the total size of items in each bin is at least the requirement R?

Certificate: the number of bin each item is placed on.

this is in polynomial length.

Certifier: Check the number of bins used is at least k.

Check each bin contains the total size of items at least R.

this can be done in polynomial time.

Claim: Partition Problem $\leq_p$ Bin Covering

Take an instance of Partition Problem. $[n] = \{x_1 \cdots x_n\}$

Since for every subset $S \subseteq [n]$, we have $\sum_{i \in S} x_i + \sum_{i \notin S} x_i = \sum_{i=1}^{n} x_i$

if there is a subset S, such that $\sum_{i \in S} x_i = \sum_{i \notin S} x_i$, then it follows that

$\sum_{i \in S} x_i = \sum_{i \notin S} x_i = \frac{1}{2} \sum_{i=1}^{n} x_i$.

We construct an instance of Bin Covering with n items, each item is of size x_i.

Claim: There is a subset in Partition Problem if and only if the set of n items can be placed into at least 2 bins such that the total size of items in each bin is at least $\sum_{i=1}^{n} x_i / 2$.

Proof in Page 8.

0ADF820D-880E-47F0-99A1-E8085A118360

csci570-fa19-exam3

#493 5 of 10

Q4 20

4) 20 pts

The instructor gives lots of homework assignments, many of which are very difficult. Furthermore, he will only accept a homework assignment if it has been solved in its entirety. Each homework assignment has a value $p_i > 0$ that represents its value to you (points awarded for successful completion) and an integer value $w_i > 0$, indicating how exhausting it is to complete. If you choose to do homework assignment i, you will get p_i points, but you will not be able to do the next w_i homework assignment(s) (so ,if you do homework i, you will have to skip assignments i+1, i+2,... i+ w_i), and will get a score of zero for those.

Your goal is to get the highest possible total score. Given the n homework assignments' values p_i and w_i, design an efficient dynamic programming algorithm to determine the highest possible total score.

a) Define (in plain English) subproblems to be solved. (4 pts)

Define OPT(i) is the highest possible total score I can get from ith assignment to the last one.

b) Write the recurrence relation for subproblems. (6 pts)

$$OPT(i) = \begin{cases} \max(OPT(i+W_i+1)+P_i,\ OPT(i+1)) & \text{if } i+W_i+1 \le n \\ \max(P_i,\ OPT(i+1)) & \text{if } i+W_i+1 > n \end{cases}$$

base case: $OPT(n) = P_n$

c) Using the recurrence formula in part b, write pseudocode (using iteration) to compute the highest possible score. (6 pts)

Make sure you have initial values properly assigned. (2 pts)

```
Initially OPT[n] = Pn
For i = n-1 to 1
    If i+Wi+1 ≤ n,
        OPT[i] = max(OPT[i+Wi+1] + Pi , OPT[i+1])
    Else OPT[i] = max(Pi , OPT[i+1])
    Endif
Endfor
```

The highest possible score will be OPT[1]

d) Compute the runtime of the algorithm described in part c and state whether your solution runs in polynomial time or not (2 pts)

Since there are n term in OPT[·], each term requires O(1) time

This algorithm takes O(n) time

It is polynomial.

3E863377-599B-4FA9-B682-854122CA3EFA

csci570-fa19-exam3

#493 6 of 10

Q5 15

5) 15 pts
A politician knows three ways to campaign. She estimates that each hour of public speech will generate 20 votes for her, each hour spent on the telephone will generate 15 votes for her, and each hour spent talking with people in shopping areas will generate 35 votes for her. The candidate wants to spend at most 5.5 hours in shopping areas and is required to perform at least eight one-hour speeches. Our politician thinks she can win if she generates a total of at least 1000 votes. How should the politician split her time in order to win while spending the least amount of time campaigning? Present a linear programming solution. You do not need to solve the problem numerically.

a) Describe your variables (4 pts)

Define x as the number of hours she spent on public speech
y as the no. of hours . . . on telephone
z as the no. . . . on talking in shopping areas

b) Present an objective function (4 pts)

minimize $x+y+z$

c) Present your constraints (7 pts)

$20x + 15y + 35z \geq 1000$

$z \leq 5.5$

$x \geq 8$

$x \geq 0, y \geq 0, z \geq 0$

7A19B049-8EDB-439F-A0A7-6E4AB40D156C

csci570-fa19-exam3

#493 7 of 10

Q6 6

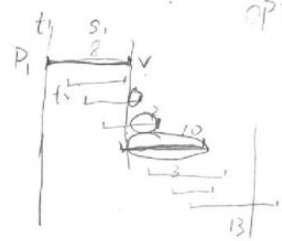
6) 15 pts

Once again, Tom is looking to solve the problem of buying smartphones. He has the information on upcoming phone releases. Each phone i releases to the public at some time t_i and is given software support for some number of years s_i and **has a cost of c_i**. Tom wants to spend as little as possible on phones over the rest of his (unfortunately finite) lifetime. Assuming that we know the date of Tom's demise and all the phone release data until that time, design an efficient algorithm to minimize the total cost of Tom's phone purchases for the rest of his life while ensuring that he never goes without a supported phone.

Sort all the phone based on their release time $P_1, P_2, \cdots P_n$

Aussme Tom starts with a phone and software support just stop at t_1

OPT !



~~Define OPT(i, T) is the minimal total cost from phone(1 ... i) and at time T.~~

Check each phone. do: initialize $t_{pre} = t_1$, buy phone P_1

if the phone $t_i + S_i > t_{pre}$

Choose the phone with the least $\frac{c_i}{t_i + s_i - t_{pre}}$

update t_{pre} as t_i

Until $t_{pre} + S_{pre} \geq$ Tom's demise

7DFE3537-501A-43B4-BA8F-140A49BCB5A8

csci570-fa19-exam3

#493 8 of 10

Additional Space

3) Proof: If there is a way to place n items into at least 2 bins with the total size of items in each bin is at least $\sum_{i=1}^{n} x_i / 2$, it's easy to show there are exact 2 bins and each bin contain $\sum_{i=1}^{n} x_i / 2$ size of items, since the total size of items is $\sum_{i=1}^{n} x_i$

Therefore, we can find a subset $S \subseteq [n]$ by checking which item is in one bin. This subset satisfies the constrain $\sum_{i \in S} x_i = \sum_{i \notin S} x_i$.

If there is a subset $S \subseteq [n]$ such that $\sum_{i \in S} x_i = \sum_{i \notin S} x_i$,

we can put the items in one bin by checking which integer is in S. and put the other in another bin. This would give us a way to Bin Covering.

6A73A260-242F-4315-8165-8A8A025549E4
csci570-fa19-exam3
#493 9 of 10

Additional Space

24AD22FD-7604-4C82-8E78-CF4EE3542E5A

csci570-fa19-exam3

#493 10 of 10