CIS 522: Assignment 8

Anubhav Shankar

Q1.)

Ans.) **Problem** -> We need to assign 'k' network routes from a source to a sink/destination node. Every file has a fixed length, and the transfer speed is uniform for all the files. So, we need to minimize the time to transfer the file through a particular route. In this problem, input is the 'n' number of files and 'k' paths, and output is the smallest time required to transfer the files.

Class -> This problem belongs to NP-Complete.

Approach -> This problem can be solved by using the Improved Load-Balancing Algorithm. Let us first sort the files by data size in descending order to minimize the file transfer time. After sorting, we will assign a path to the file with the smallest load.

Pseudo-code ->

Start with no jobs assigned

Set Ti = 0 and A(i)= 0 for all machines Mi

Sort jobs in decreasing order of processing times

Assume that $t_1 >= t_2 >= ... >= t_n$

For (j = 1; j < n; j++)

Let M~ be the machine that achieves the minimum $\min_k T_k$

Assign job j to machine M_j //transfer file to route

Set A(O <--A(i) U {j} //Assign file to route

Set $T_i = T_i + t_j$ // Update the processing time

EndFor

Return T_i //Return the final processing time

Walkthrough ->

10 files:- 5,6,7,3,2,10,9,8,6,4

3 Paths

Transfer Time speed limit :- 20

Now sort all files in descending order as per their processing time post –

10,9,8,7,6,6,5,4,3,2

Set Ti = 0 and A(i)= 0 for all machines Mi

For (j = 1; j < n; j++){

route 1 :– 8,7,4

route 2:- 10,6,3,2

route 3:- 9,6,5

}

Return T_i = 21 as the make-span which belongs to path 2

Time Complexity -> As the sorting happens in O(n) time and the assignment of routes takes O(log n) time within the for loop, the time complexity of the algorithm will be O(n log n).

Algorithm Approximation Guarantee -> the given problem's optimal solution is a 20-speed limit, and our algorithm provides a time of 21-speed limit, so 21/20 = 1.05. So here L <= 3/2 L*. Hence, by Lemma 11.5, the job schedule is optimal.

Q2.)

Ans.) **Problem** -> We need to run jobs on a processor which is only available at specific time frames. Every job requires some interval time. The primary constraint is that no two jobs can overlap. The algorithm should produce a schedule that does not have overlapping jobs.

Class -> This problem belongs to NP-Complete.

Proof -> Let there be four jobs, J1: 10-12, J2: 8-11, J3: 9-10, J4: 8-12

To ascertain whether two jobs are overlapping, we'll use a certifier. If the certifier gives yes, then we can proceed. Else we'd need to revisit the schedule.

Let's consider J1 and J3, we see that the jobs are non-overlapping, and the certifier returns yes.

We will now model this problem as an Independent Set problem and see whether we get non-overlapping job sets.

Let there be a graph, G(V, E), where the four jobs are the vertices and e ε E be the edges connecting them. The certifier will again return a yes or no output. Using the same, we can say that this problem belongs to NP-Complete class.

Q3.)

Ans.) **Problem** -> We have ‘n’ airports and ‘m’ flights which need to be scheduled. A flight can be scheduled from the source airport to the sink/destination airport. Building an airport facility has an associated cost. So, we need to select the subset of airports with minimum cost to ensure that all flights are served.

Class -> This problem belongs to NP-Complete.

Approach -> This problem can be modeled as a weighted vertex cover problem. Here, airports are the nodes, and flights are the edges between the start and destination airports. The inputs for this problem are ‘n’ number of airports and ‘m’ number of flights. The output should be the minimum price cost between the sites.

Walkthrough & Pseudocode ->

Notations: n -> number of airports

m -> number of flights

S -> tightest set

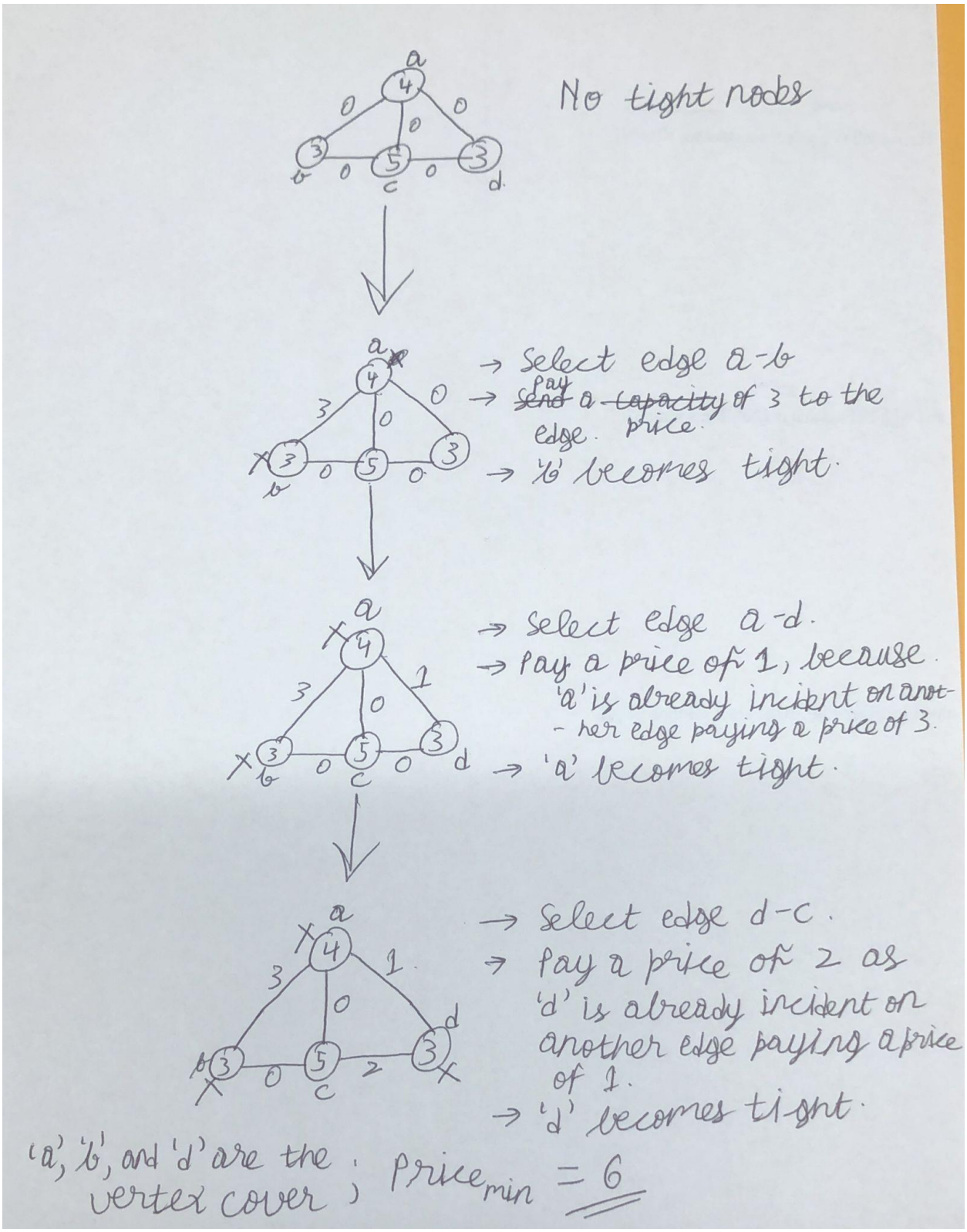
e -> edges

p_e -> set of edges

```
Vertex-Cover-Approx(G, w) :

Set pe = O for all e ε E

While there is an edge e=(i,j) such that neither i nor j is tight:

Select such an edge e
Increase Pe without violating fairness
EndWhile

Let S be the set of all tight nodes.

Return S
```

The walkthrough of the pseudo-code is given below:

The tight nodes have a cross-mark near them.



Time complexity: The typical time complexity of the algorithm would be O(m * n). The TC can get between $O(n^2)$ and $O(n^3)$ in the worst-case scenario.

Q4.)

Ans.) **Problem** -> As we all know, the longer the transmission line, the poorer is the connectivity. In this problem, the telephone company requires its base stations to be near its customers. So, we need to propose an algorithm that minimizes the distances between a set of houses and the base station.

Class -> This problem belongs to the NP-Hard class.

Approach -> The problem can be solved using the Center Selection algorithm to obtain an approximation.

Pseudocode -> A greedy approach to the Center Selection algorithm is given below:

Notations: k -> the number of points, in this case base stations

S -> Set of sites

C -> The center obtained

n -> number of houses

Assume k <= ISl (else define C=S)

Select ANY site s and let C = [s]

While [C]< k

Select a site s ε S that maximizes *dist(s, C)*

Add site s to C

EndWhile

Return C as the selected set of sites

In the above algorithm, we can select the site 's' that is farthest from all previously selected centers: If there is any site that is a set threshold away from all previously chosen centers, then this farthest site s must be one of them.

Time Complexity -> The typical time complexity of the algorithm would be O(k * n). In the worst-case scenario, the TC can get between $O(n^2)$ and $O(n^3)$.

Q5.)

Ans.) **Problem** -> We need to schedule a set of jobs to avoid conflict. There are many job requests, and each requested job gives a specific payment. However, there is a set of conflicts. So, the algorithm needs to maximize profits while simultaneously preventing conflicts in the schedule.

Class -> This problem belongs to the P space and can be solved in polynomial time.

Approach -> This problem can be solved by Dynamic Programming.

Pseudocode ->

Notations: c -> Completion time of a job

P -> Profit obtained

J -> job

Sort all the jobs basis their completion times.

Sort jobs ($c_1 <= c_2 <= .... <= c_n$) // sort jobs as per the completion time

Let P(1),P(2) … P(n) is the profit of each job

If (j equals to 0) // checks condition

Return 0

Else

For(int i = 1; i <= n; i++) // loop condition

opt[j] = max(Ai + optimal[P(i)],opt[i - 1]) // optimal solution

Time Complexity -> The time complexity of the above approach will be O(n log n)

Walkthrough -> J1: 0 -4, Profit = 4

J2: 1-5, Profit =5

J3: 4-5, Profit = 3

The above algorithm will schedule J1 & J3 with a max profit of 7.

Q6.)

Ans.) **Problem** -> We need to set up security cameras at select crossroads and then monitor multiple roads. However, our budget is limited. We need to cover ‘m’ roads by placing cameras at ‘n’ cross-roads.

Class -> This problem belongs to NP-Complete.

Approach -> This problem can be modeled as a regular vertex cover problem where the nodes will be the cross-points and the roads will be the edges of the resultant graph.

Notations: m -> number of roads

n -> number of crossroad points

C -> set of cross-points

Pseudocode ->

```
G(V,E)   // initialize graph
If (r is present in graph G) // check condition that road is present or not
{
Choose such cross point cp which is connected to number of roads in a city
// if road is present then select cross point of such road
Insert cross point to set C
// add such cross point to set c
Delete all roads which are attached to selected cross point
// remove other roads which are connected with the cross point
Return set C
// in set c get cross point to install cameras
}
```

Time Complexity -> The typical time complexity of the algorithm would be O(m * n). The TC can get between $O(n^2)$ and $O(n^3)$ in the worst-case scenario.