Assignment - 4

CS 5084

Name : Adhiraj N Budukh

Email : abudukh@wpi.edu.

1. Given,

G is an arbitrary connected, undirected graph with distinct cost $c(e)$ on every edge 'e'

e^* is cheapest edge & $c(e^*) < c(e)$ for every edge $e \neq e^*$

If we consider kruskal's algorithm, e^* is the 1st edge
∴ It will be included in the minimum spanning tree.
∴ Statement is True.

2. a) If we replace c_e by c_e^2 costs & consider Kruskal's algorithm.
it will sort in the same order and further by putting same subset of edges in the Minimum Spanning True
∴ the statement is True.

2 (b) Given,

Shortest path s-t on Graph 'G'
(all costs are positive & distinct)

P, minimum cost s-t path for the given instance.

We are replacing each cost c_e by c_e^2 & creating a new instance of the problem with the same graph but different cost.

Lets consider graph 'G' have edges
(s, a) with cost k
(a, t) with cost k
& (s, t) with cost > k

Then the shortest path is the single edge (s,t) but after squaring cost of shortest path would go through point 'a'. Hence, 'P' cannot be the minimum cost for given path s-t.

∴ The statement is False.

3. lets say company has to ship '7' packages from NY to Boston

let Package weights be : 2, 3, 1, 6, 4, 2, 5
let max. wt. limit of each truck : 8

The company wants to use fewest no. of trucks possible to transport all packages from NY to Boston. They use Greedy Algorithm where they pack the packages in order they arrive, whenever the next package does not fit in current truck, they send truck on its way.
⇒ if package does not fit to current truck it is assigned to next truck.

To answer the given question we need to prove the Greedy Algorithm company currently using is actually minimizing the no. of trucks need.

To prove that we need to show Greedy Algorithm is optimal than any other solution.

If Greedy algorithm fits package = 1, 2, ... j in first x trucks & another solution fits packages = 1, 2, ... i in first x trucks $(i \leq j)$

To support our claim apply induction on 'x'
for base case $x = 1$, greedy algorithm fits as many packages as possible into first truck
In our example it Package 1, 2, 3 (with wt. 2, 3, 1)

Now, lets assume our claim for $x \Rightarrow x-1$
we want to show that it holds also for x.

Suppose the greedy algorithm fit j' packages into first $x-1$ trucks &
another solution fits i' packages into $x-1$ trucks ~~and~~ where $i' \leq j'$
So, here we want to show that the Greedy Algorithm can fit at least as many packages as the other solution in the x^{th} truck & it can potentially fit more.

In our example by Greedy Algorithm boxes would go as per their order as:

Truck 1 : 2, 3, 1 = 2 + 3 + 1 = 6 wt. < 8
Truck 2 : 6 wt. < 8
Truck 3 : 4, 2 = 4 + 2 = 6 wt. < 8.
Truck 4 : 5 wt. < 8.
it would use 4 trucks.

However in alternate solution,
if boxes send in the order of:
2, 6, 1, 3, 4, 2, 5 then,
Truck 1 : 2 + 6 = 8
Truck 2 : 1 + 3 + 4 = 8
Truck 3 : 2 + 5 = 7
In this case / solution less no. of trucks are used but order is not being followed hence. Greedy Algorithm is optimal solution here.

5. lets start at the western end & start moving east until we reach to some house 'h' at 4 miles. miles. We cannot go further as given condition every house within 4 miles is one of base station.

Now, all houses covered by base station are deleted an we have to repeat process on the remaining houses.

We deleted the houses to avoid redundant base stations, this will reduce the no. of remaining houses that need to be covered & ensure that the final set of base station position is as small as possible while still covering all houses.

We can also consider as follow:

we can put first base station at the eastern most position that covers all houses from western end upto that position.

Then we can iteratively place base station at largest position that covers all houses between the previous base station and new one.

8. Proof by contradiction,

Suppose, T_1 & T_2 are 2 distinct minimum spanning trees of G.

$\because$ T_1 & T_2 have the same no. of edges, but are not equal there some edge 'e' in T_2 exist but not in T_1.

If we add e to T_1 we form cycle. → ①

let e_2 be the most expensive edge on this cycle of eqn ①.

$\therefore$ By cycle property e_2 does not belong to minimum spanning tree.

Which contradicts our assumption.

9. Given,

$G = (V, E)$ connected graph

n: vertices

m - edges

$T = (V, E') \Rightarrow$ spanning tree of G

Defined bottleneck edge of T be the edges of T with greatest cost

a) Every minimum-bottleneck tree of G is not a minimum spanning tree of G.

let veritices of G be $v_1, v_2, v_3 v_4$ with the edge between each pair of vertices let weight of their edge = sum of endpoints.

The tree consisting of path through vertices v_3, v_2, v_1, v_4 has bottleneck edge of some weight which is smallest possible, but its not minimum tree

∴ 'a' is false

b) 'b' is true.
let T_1 be minimum spanning tree of G
& T_2 be spanning tree with lighter bottleneck edge.
∴ T_1 has an edge 'e' that is heavier than every edge in T_2 → ①
if we add e to T_2 then it forms cycle.
∵ all other edges in cycle belongs to T_2
it becomes heaviest edge.
∴ By cut propert of Graph
e should not be belonging to any minimum spanning tree.
⇒ contradicts eqⁿ ①.

21. Given tree is connected graph with at most $(n+8)$ edges. To find minimum spanning tree we have to follow following steps:

- We need to apply cycle property 9 times by performing BFS until we find cycle in the graph G.

since, we know the minimum spanning tree of graph with 'n' vertices has exactly 'n-1' edges, if we can reduce the no. of edges in the given near-tree G to n-1 by deleting edges without changing MST, we'll have found the MST of G

- Starting with n+8 edges, deleting one edge at a time we can reduce the no. of edges to n-1 in exactly '9' steps.

- Every time we need to make sure we delete heaviest edge on cycle.
- After 9 steps we'll have connected graph with 'n-1' edges & the same minimum spanning tree as 'G'
  ∴ It is minimum spanning tree.

Running time:-
- For each iteration it is $O(m+n)$ for BFS & for corresponding cycle.
  ∀ n - no. of vertice, m - edges.
  ∵ m is at most n+8 & we applied cycle 9 times
  ∴ Running time is $O(9n)$ ~~which~~ i.e. $O(n)$