CS430 Homework 3

Chapter 1

Problem-1:

Answer:

N boxes arrive in order b1, b2, b3…….bn

Each box bi has positive weight wi where maximum weight carry by truck is W and

assigned into M trucks 1, 2,…….M

Such that total weight of all boxes <=W and for i < j , if bi is sent before bj then bi must arrived at company before bj.

We Prove that, for a given set of boxes with specified weights, the greedy algorithm currently in use actually minimizes the number of trucks that are required.

Greedy is not optimal and other solution is optimal

If Greedy- algo fits b1, b2 ….bi boxes in k trucks and Optimal algo fits b1, b2,…….bj boxes in k trucks, then j <= i

For k=1

Greedy algorithm fits as many as boxes into first truck.

For k=k-1

We will assume as our induction hypothesis that the statement is true for k − 1, Greedy algorithm fits x boxes into first (k-1) truck and other Algorithm fits y boxes into (k-1) truck
y < = x

For truck k > 1

Optimal solution can fit boxes till by+j.

Greedy algorithm is able to fit boxes bx+i and it can fit more boxes as y<=x

Prove greedy Algorithm is the best optimal solution for this problem.

Problem-2:

Give an algorithm that takes two sequences of events S’ of length m and S of length n

Subsequence S' is a subsequence of S.

Sequence of events in set S= {s1, s2, s3,……., Sl}

Sequence of events in set S'= {s1, s2, s3,……., Sm }

1. Initialize i=1, j=1
2. While i <= l and j <=m
3. If si == s'j
   Increase value of i and j by 1
   Else
   Increment i by 1
   End if
4. End While
5. If j > m

   print S' is subsequence of S

   else

   print S' is not subsequence of S

   End If

Running Time is: O(n)

Problem-3

In the following algorithm, the input is an array H of positions of houses. Output is list B of locations of base station.

1. sort H
2. B=[ ]
3. p = -∞
4. for i = 1 to sizeof (H) do
5. if H[i] > p + 8 then
6. B. append(H[i] + 4)
7. P = H[i]
8. return B

Running time is O (n log n).

Suppose that our algorithm assigns locations b1,……bk to base station, whereas an optimal algorithm assigns z1,……. zm .

Claim: for all 1<=i<=k, we have zi <= bi.

Proof by induction:

For the base case, we have that b1>= z1, since z1 has to cover X[1] and we select the farthest position that reaches this.

Assume the claim is true for some I , 1<= i < k.

If bi+1 < zi+1 and bi > zi then the house immediately to the east of position bi + 4 would not be covered by z1,……. zm.

Hence we must have bi+1 > zi+1, proving the claim.

Since our algorithm clearly covers all the houses, we must have m = k.

Problem:4

Suppose you are given a connected graph G, with edge costs that are all distinct. Prove that G has a unique minimum spanning tree.

Proof by Contradiction:

Assume G has not a unique minimum spanning Tree.

Means there are 2 different MST of graph G say MST A and MST B

Both tree MST A and MST B have same number of edges

Let Edge **e1** with lowest weight that is present in MST A but not MST B

There must be some edge **e2** in MST B but not in MST A.

Assume weight of e1 is less than weight of e2

As Tree B is MST so {e1} + B must contain a cycle

Replace e2 with e1 in B produces MST that has smaller weight compare to Tree B.

Which contradict with our assumption that B is MST but actually it is not.

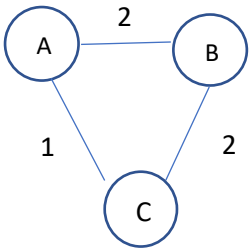
Problem 5:

Is every minimum-bottleneck tree of G a minimum spanning tree of G? Prove or give a counterexample.

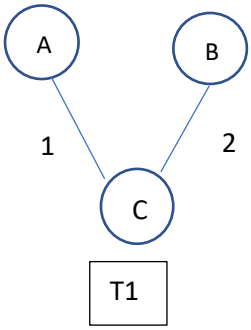
Answer:

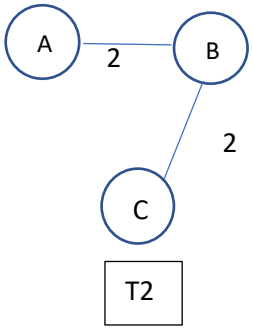
No. Every Minimum bottleneck tree of G are not Minimum spanning Tree.

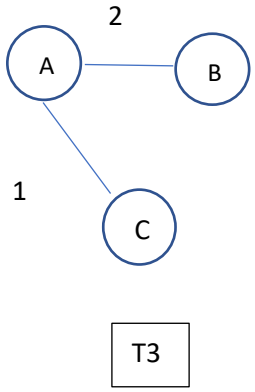
For Graph G =(V,E)



Spanning Tree of graph G







Bottleneck Edge Weight: 2 Bottleneck Edge Weight: 2 Bottleneck Edge Weight: 2

Total Weight = 3 Total Weight = 4 Total Weight = 3

Minimum spanning Tree is: T1 and T3

Minimum bottleneck spanning tree are: T1, T2 and T3 with bottleneck edge weight is 2

So, here every minimum-bottleneck tree of G are not a minimum spanning tree of G

(B)

Is every minimum spanning tree of G a minimum-bottleneck tree of G? Prove or give a counterexample

Answer:

Yes.

With the same example we can say that every MST of graph G is a Minimum bottleneck tree of G.

T2 is Minimum spanning tree as well as minimum bottleneck spanning tree of graph G.

Proof by contradiction:

T1 is a minimum spanning tree of graph G

T2 is a spanning tree with lighter bottleneck Edge

Thus, T1 contain an edge e1 that is heavier than every edge in T2

So, if we add edge e1 to tree T2 it forms a cycle on which it is heaviest edge

But by the cut property, edge e1 does not belong to any MST which contradict

with fact that edge e1 is in tree T1 and T1 is MST.

Problem-6

a)

Give an efficient algorithm to test if T remains the minimum-cost spanning tree with the new edge added to G (but not to the tree T).

Make your algorithm run in time O(|E|). Can you do it in O(|V|) time?

Please note any assumptions you make about what data structure is used to represent the tree T and the graph G.

Answer:

Minimum spanning tree T contains n-1 edges.

The addition of the new edge e = (u, v) with weight W in graph G creates exactly one

cycle C in the graph G.

1. Let e = (v, w) new edge
2. Let Ce= the cost of the new edge
3. Find shortest path P from v to w in tree T
4. For each edge, if the cost of the edge is less than Ce, then T is still MST

   //Edge e is not in minimum spanning tree, it is most expensive edge on path
5. Else
6. if the cost is greater than than Ce, then T is no longer a minimum spanning tree

   //
7. End If

8. End For

We can implement this algorithm in O|V| if Tree T represented in form of Adjacency list.

b)

Suppose T is no longer the minimum-cost spanning tree.

Give a linear-time algorithm (time O(|E|)) to update the tree T to the new minimum cost spanning tree.

Answer:

If T is no longer Minimum spanning Tree

we can generate a lower weight MST by replacing that higher-weight edge in tree T

with new edge e and obtain Tree T'.

Assume: all edges costs are distinct

Algorithm:

1. Find the unique path P from u to v in T
2. Find the edge e' in P that has maximum weight w'
3. If w < w'

   // T is not minimum spanning tree

   T' = T – e' + e is the MST for G with new edge,

   weight(T') = weight(T) – w' + w
4. Else
5. T' = T is the MST of G + e
6. End If

Claim: T' is minimum spanning Tree

Proof:

If Edge costs are not distinct

We modify all edge cost by small amount to make them distinct.

We first add small quantity to new edge e and update cost of all other edges x by much smaller quantity x.

So, for tree T C(T) is original cost and c(T') is updated cost.

Case-1:

For tree T1 and T2 if cost of C’(T2) is less than C’(T1) then cost of Tree T2 is also less than Tree T1.

Case-2:

If actual cost of both tree T1 and T2 are same and T1 contain edge e but T1 doesn’t then cost of T1 must be greater then cost of T1.

So, in both case if we compute MST with updated cost is also MST with respect to actual cost.

After updating weight If cost of T’ less than T and edge e contain by T’ but not by T

implies actual cost of T’ is less and equal to cost of (T)