

# CS430 Homework 1

## Chapter 1

### Problem-1:

#### Answer:

N boxes arrive in order  $b_1, b_2, b_3, \dots, b_n$

Each box  $b_i$  has positive weight  $w_i$  where maximum weight carry by truck is  $W$  and assigned into  $M$  trucks  $1, 2, \dots, M$

Such that total weight of all boxes  $\leq W$  and for  $i < j$ , if  $b_i$  is sent before  $b_j$  then  $b_i$  must arrived at company before  $b_j$ .

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  $b_1, b_2, \dots, b_i$  boxes in  $k$  trucks and Optimal algo fits  $b_1, b_2, \dots, b_j$  boxes in  $k$  trucks, then  $j \leq i$

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  
 $y \leq x$

For truck  $k > 1$

Optimal solution can fit boxes till  $b_{y+j}$ .

Greedy algorithm is able to fit boxes  $b_{x+i}$  and it can fit more boxes as  $y \leq x$

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  $S = \{s_1, s_2, s_3, \dots, s_l\}$

Sequence of events in set  $S' = \{s_1, s_2, s_3, \dots, s_m\}$

1. Initialize  $i=1, j=1$
2. While  $i \leq l$  and  $j \leq m$
3.   If  $s_i == s'_j$   
      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.  $p = -\infty$
4. for  $i = 1$  to  $\text{sizeof}(H)$  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  $b_1, \dots, b_k$  to base station, whereas an optimal algorithm assigns  $z_1, \dots, z_m$ .

**Claim:** for all  $1 \leq i \leq k$ , we have  $z_i \leq b_i$ .

**Proof by induction:**

For the base case, we have that  $b_1 \geq z_1$ , since  $z_1$  has to cover  $X[1]$  and we select the farthest position that reaches this.

Assume the claim is true for some  $i$ ,  $1 \leq i < k$ .

If  $b_{i+1} < z_{i+1}$  and  $b_i > z_i$  then the house immediately to the east of position  $b_i + 4$  would not be covered by  $z_1, \dots, z_m$ .

Hence we must have  $b_{i+1} > z_{i+1}$ , 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  $e_1$  is less than weight of  $e_2$

As Tree B is MST so  $\{e_1\} + B$  must contain a cycle

Replace  $e_2$  with  $e_1$  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.

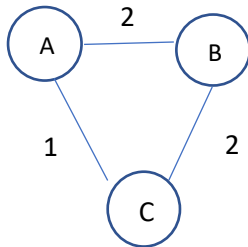
#### **Problem 5:**

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

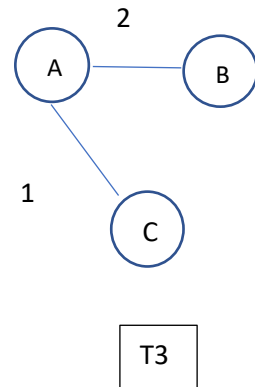
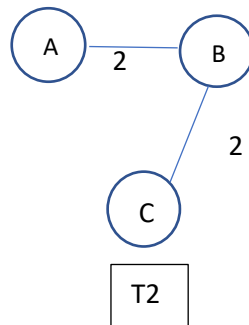
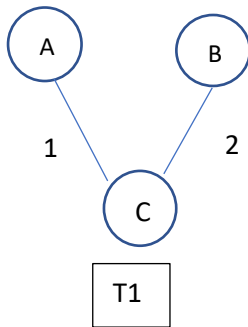
**Answer:**

**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

Total Weight = 3

Bottleneck Edge Weight: 2

Total Weight = 4

Bottleneck Edge Weight: 2

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:**

1. Let  $e = (v, w)$  new edge
2. Let  $C_e$  = the cost of the new edge
3. Find shortest path P from v to w in tree T //Tree **T represented as an adjacency list**
4. For each edge if the cost of the edge is less than  $C_e$ , then T is still a 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  $C_e$ , then T is no longer a minimum spanning tree  
//
7. End If
8. End For

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:**

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  $G$ .

If the new edge weight ( $W$ ) is less than the weight of the highest-weight edge in this cycle  $C$ , then we can generate a lower weight MST by replacing that higher-weight edge with  $e$ .

**Assume: all edges cost is 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,**  
     **$\text{weight}(T') = \text{weight}(T) - w' + w$**
4. Else
5.      $T' = T$  is the MST of  $G + e$
6. End If

By replacing heaviest edge on path  $P$  in tree  $T$  with new edge  $e=(v, w)$  obtaining new tree  $T'$ .

**Claim:**  $T'$  is minimum spanning Tree

**Proof:**

Consider any edge  $e'$  not in Tree  $T'$ .

By cycle property  $e'$  is not part of any spanning tree. so adding  $e'$  to tree  $T'$  gives a cycle  $C'$ .

If we add new edge  $e'$  to tree  $T$  it creates a cycle  $K$ .

So, by cycle property  $e'$  is most expensive edge on cycle  $K$ .

Suppose heaviest weight edge  $f$  is most expensive edge on cycle  $C$

$e'$  is most expensive edge on Cycle  $K$

New edge does not belong to  $C'$  then  $C'=K$  and  $e'$  is most expensive edge on  $C'$ .

Otherwise, Cycle  $K$  include  $f$  and  $C'$  use portion of  $C$  and  $K$ .

In this case  $e'$  is most expensive edge than  $f$  so most expensive than other edge on  $C$ .

So, most expensive edge on  $K$  and  $C'$  as well.

### **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.

The following two property hold:

- 1) For  $T_1$  and  $T_2$   
If  $C'(T_2) < C'(T_1)$  then  $C(T_2) < C(T_1)$
  - 2) If  $C(T_1) = C(T_2)$  and  $T_2$  contain  $e$  but  $T_1$  doesn't then  $C(T_2) > C(T_1)$
- a)  $C'(T_1) < C'(T)$  and edge  $e$  contain by  $T'$  but not by  $T$  property (1) implies  $C(T') < C(T)$  and property (2) implies  $c(T') < C(T)$
- b) If we compute minimum spanning tree with updated cost by property (1) is also one of the MST with respected to actual cost.