Soumish- Agarwal 2019mT60763

Ques 1) given $G = \{V, E\}$, $T$ = MST of $G$. Let $e = \{x, y\}$ be edge in $T$, then show that $e$ there exist some cut $\{S, V \setminus S\}$ st $e$ is the Edge with smallest weight crossing the cut.

So if we remove Edge $e$ from $T$ then $T$ will be disconnected. After removing $e$ $T$ will break into 2 components $T_1, T_2$ with vertex set $V_1$ and $V_2$ respectively st $V_1$ contains $x$ and $V_2$ contains $y$. If we assume $S = V_1$ and $(V \setminus S) = V_2$

Then we have to show $(S, V \setminus S)$ has $e$ as smallest weight Edge crossing the cut.

We will proof by contradiction.
Suppose $e$ is not the smallest weight crossing the cut. Then there exists Edge $e'$ st $w(e') < w(e)$

Given $T_1, T_2$ = trees then $T_1, T_2$ will be connected.
Using property of tree [no of edges = no of vertex - 1]
we have,

$T_1$ : $m_1 = h_1 - 1$ $\quad \therefore h_1 + h_2 = h$
$T_2$ : $m_2 = h_2 - 1$ $\quad |V| = h$, $|E| = m$

Let $T_3 = T_1 \cup T_2 \cup \{e'\}$ — ① $T_3$ also connected

$T_3$ : $m_3 = m_1 + m_2 + 1$
$= h_1 - 1 + h_2 - 1 + 1$
$= h_1 + h_2 - 1$

This implies $T_3$ is also spanning tree of $G$.

Saransh Agarwal 2019MT60763

Now by Eq 1)

$$cost(T_3) = cost(T_1) + cost(T_2) + w(e')$$

$$< cost(T_1) + cost(T_2) + w(e)$$

$$< cost(T)$$

$\Rightarrow$ $T_3$ is spanning tree with Total cost less than $T$. which is wrong as $T$ is MST.

Hence contradiction, so $e$ must be smallest Edge weight crossing the cut

Hence proved

**Ques2)** We will prove by contradiction, Suppose that for every cut of graph there is unique light edge crossing the cut, but we have 2 distinct MST T and T'.

Let $\{u,v\}$ be edge in T not in T'. Then if we remove edge $\{u,v\}$ then T will disconnected and tree will break into two components. Let $T_u$, $T_v$ be vertices in component containing u and v. Then let cut $\{T_u, T_v\}$ and edge $\{a,b\}$ be unique light edge crossing the cut. If $a \neq u$ and $b \neq v$ then clearly $w(a,b) < w(u,v)$ and MST $T = \{(u,v)\} \cup \{(a,b)\}$ has less cost than T, which contradicts.

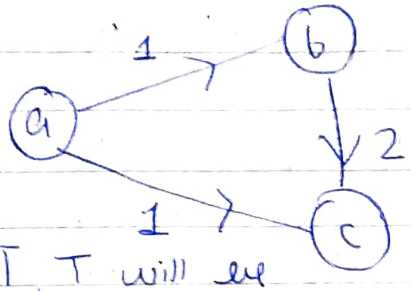
Now assume that $(a,b) = (u,v)$. Now if consider a path from u to v in T' which starts in $T_u$ and Ends in $T_v$, hence there must be some edge c on it crossing the cut $\{T_u, T_v\}$. But $\{u,v\}$ is unique light edge crossing the cut, $w(u,v) < w(c)$. Now if we add an extra Edge $\{u,v\}$ in T' then cycle will be formed in T'. Now if we remove any other Edge we again get MST. Thus

$T' \cup \{(u,v)\} - \{c\}$ is spanning tree and it has Lower cost than T', which is a contradiction.

Hence proved

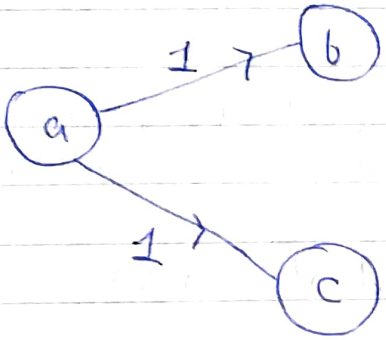
Counterexample of the converse.

Let $G = \{V, E\}$ with three vertexes st




Then MST, T will be

T =




T contains the edge $\{a, b\}, \{a, c\}$ but cut $(\{a\}, \{b, c\})$ does not have a unique light edge crossing the cut.

Saransh Agarwal 2019MT60763

Ques3) Algo)

A, B → two arrays
h → Size of arrays

```
MEDIAN (A, B)
    If h == 1
        return min (A[0], B[0])
    Endif
    if (A[h/2] < B[h/2])
        return MEDIAN (A[h/2 + 1 ..... h], B[0, ..... h/2])
    Else if
        return MEDIAN (A[0, ..... h/2], B[h/2 + 1, ..... h])
    End
```

Running Time Analysis)
At Each step it takes O(1) time i.e one access per array.
So we get recursion $T(h) = T(\frac{h}{2}) + 2$

$T(h) = O(\log h)$ times

Every time we ask for $(\frac{h}{2})^{th}$ Smallest element.

proof of correctness)
proof by induction.

Base case) If h=1 Then there will be only 2 elements in merged array and median = min (A[0], B[0])

Saransh Agarwal 2019mT60763

IH) Lets the algo. be true for $n \leq k$ for some $k \geq 2$.

IS) We have divided array into halfs

let $x$ = median of $A[\frac{n}{2}]$

$y$ = " of $B[\frac{n}{2}]$ $\quad n = 2k$

Given elements are distinct so $x \neq y$.

WLOG $x < y$ [Similar arguments can be used for $x > y$]

→ If $x < y$ Then there cannot be element in right half of B which is median since it has atleast $\frac{n}{2}$ elements less than element of B and also atleast $\frac{n}{2}$ elements less than of A. as $x < y$.

→ Similarly we can say there cannot be element in left half of A which is median since it has atleast $\frac{n}{2}$ elements greater than element of A and also atleast $\frac{n}{2}$ elements greater than of B as $x < y$.

→ By above 2 arguments there will be more than $n$ elements greater than , $4n$ total $= 2n$, but median will be at $n^{th}$ pos. in merge array.

→ The problem now reduces of finding median in remaining half of arrays of half the length which is correct by IH.

Hence proved

Ques4) $G = \{V, E\}$ undirected graph with cost $c_e > 0$
$T$ = MST of $G$. Now assume that a new edge $(e)$ is added to $G$, connecting two nodes $v, w \in V$ with cost $c$.

Algo 1

Step 1) Run DFS on $T$ with vertex $v$ and find a path from $v$ to $w$.

Step 2) If there exists an edge $(e')$ in this path from $v$ to $w$ which has cost larger than $c$, then $T$ will not be MST.

Step 3) Else it will be MST.

Algo 2) It is not MST

Step 1) from above algo 1 we will know whether $T$ is still MST or not.

Step 2) If ans of above algo is no then Remove $e'$ from $T$ and add $e$ to $T$. The resulting tree will be MST of $G' = (V, E \cup \{e\})$

Running Time Analysis

Algo 1

DFS takes $O(m+n)$ time as $T$ is tree

and finding a path takes $O(n)$ time so $m = n-1$

$n = |V|$

$m = |E|$

Total Time $= O(m+n) = O(2n+1)$

$= O(n)$

$= \boxed{O(|E|)}$

Saransh - Agarwal 2019mT60763

Algo 2)

DFS takes $O(m+h)$ time
and finding a path takes $O(h)$ time

$$\text{Total Time} = O(m+h) = O(2m+1) = O(m) = \boxed{O(|E|)}$$

$\therefore$ as T is tree
So $m = h-1$ $\quad h = |V|$ $\quad m = |E|$

proof of correctness for Algo 1 :

→ We know T is MST of G, if we add edge $e = \{v, w\}$ to it, cycle will formed in T. We are running DFS on T by property of tree we can say there will be unique path from v to w.

→ The path from v to w with the edge e together they form a cycle.

→ If there exist an Edge (e') in this path from v to w which has cost more than c so if we remove this edge (e') from the cycle we will get another spanning tree with cost less than T. This implies T is MST.

→ If there exist no such Edge that means e is costly Edge in cycle so e will not be part of MST. T is still MST

HENCE proved

Saransh Agarwal 2019mT60763

## Proof of correctness for Algo 2 →

In part 1 we have proved if we add edge $e = \{v, w\}$ to T, cycle will formed in T. And after running DFS on T we get unique path from v to w with which alongwith the Edge e form a cycle. By Algo 1, we get the most Expensive edge of the cycle.

Algo 1 outputs yes / no. If output is no then we know that e is not the most costly edge of the cycle given T is not MST of G'. In part 1 we proved if e is the costly edge in cycle then T will still be MST of G'.

As e' is most costly Edge so $T \cup \{e\} - \{e'\}$ has less cost than T so it will be MST of G'.

Hence proved

Saransh- Agarwal 2019MT60763

Ques5) Input : Array of n-objects

Output : return true if there is an object which is present more than $\frac{n}{2}$ times.

Algo)

Step1) Divide array into two subarrays [L, R].

Step2) Now there can be 2 cases : first in both half majority element is absent then it will also be absent in A, return false.

Step3) Else solve recursively for majority element Suppose it returns a [at index $i_1$] and b [at index $i_2$].

Step4) Now we will count occurence of a and b in array A using given function f we will transverse the array A using $f(i_1, t)$ and $f(i_2, t)$ where t runs from 1 to size of array A . If $f(i, t)$ returns true we will increase counter by one. If either of occurence of a or b is more than $(\frac{n}{2})$ return true .

Step5) If neither of them is more than $(\frac{n}{2})$ times return false.

Running Time analysis :

At each step we are dividing array into 2 halfs and checking through the complete array to count its occurence which takes $O(n)$ time. $\Rightarrow T(n) = 2T(\frac{n}{2}) + O(n)$

$T(n) = O(n \log_2 n)$

Saransh - Agarwal 2019mT60763

Lemma) If in array there is an element [p] which appears more than $\frac{h}{2}$ times then it will appears more than [$\frac{1}{2}$ size of subarray] times for atleast one of subarrays in which array is Divided.

Proof) By contradiction let it be false.
Assume $h_1, h_2 \ldots\ldots h_n$ sizes of partition subarray
$\sum_{i=1}^{h} h_i = h$ [size of array].

Now, count of p in $i^{th}$ subarray $\leq (\frac{h_i}{2})$
count of p $\leq \sum_{i=1}^{h} (\frac{h_i}{2}) = (\frac{h}{2})$

But count of p $> \frac{h}{2}$ Hence above lemma is true.

Now using this we proof our algo:
Base case) If h=1 then we have only one element so it occurs more $\frac{h}{2} = \frac{1}{2}$ times [trivial]

IH) let algo works correctly for $h \leq k$ for some $k \geq 2$. h = size of array.

IS) Let array size by 2h By step 4 of algo we see that if a and b are majority element in subarrays and occurence of either of them is more $\frac{h}{2}$ in array then majority exists.

→ else majority element doesn't exists [this is true by lemma proved above].

→ for every majority element of subarray checks it occurrence with complete array. Hence it returns majority element in array if it exists

Hence proved

Ques6) Algo

```
                                T = root of tree
Local_min(T)                    L, R = left, right child.
    If T has a children
        Set probe values as X_L, X_R, X_T
        If X_L < X_T
            return Local_min(L)  // in left subtree
        Else if X_R < X_T
            return Local_min(R)  // in right subtree
        Else
            return T
    Else
        return T
```

Running time analysis

Given $h = 2^d - 1$

$\log_2(h+1) = d$

When each recursive call is made, it is called on a node at level 1 less than current node.

Algo will recursively transverses the tree till end, so it will atmost d steps.

We can see for each point there are 3 probes so Total no probes $= O(3 * d)$

$= O(3 * \log_2(h+1))$

Total running Time $= \boxed{O(\log_2 h)}$

Saransh - Agarwal 2019mT60763

## Proof of correctness

→ First we consider T be the root of entire tree. In this case T has no parent and then it is trivial that T is local-min.

→ At any point of algo value of node v at which we are runing algo will be less than of its parent. Due to the fact that algo will executes forward only if atleast one children has value less than node.

→ ∴ parent of node v will call v recursively only if value of v is lesser than it.

→ At each step either we are returning node or going 1 level lower so algo will definety terminate.

Now let T be the node returned by algo there are 2 cases.

Case 1) When both of its children has a greater value than itself. T will be local-min as parent was shown to have greater value as well.

Case 2) T has no children i.e T is leaf node In this case the algo only node T would be connected to its parents and we know parent value is greater than T, hence T will be Local minimum.

Hence proved