YUQIAN ZHANG N19945556 Recitation. Ojas

Fundamental Algorithms Assignment #10

Q1. proof: suppose the old MST T is still the MST.
then it should have the minimum weight w.(T)
$w(T) = \sum_{x \to y} w(e) + \sum_{rest} w(e)$

However, there is an edge $\{x, y\}$ originally not covered in T and now reduce its weight $w(x,y)$ to a new value w.

And w is less than an edge of the path from x to y in MST T. That means $w < \sum_{x \to y} w(e)$
By definition of MST, we will replace $\sum_{x \to y} w(e)$ by w for the path from x to y.
Now after recalculation, we have the new min weight $w(T)' = w + \sum_{rest} w(e)$ to form MST.

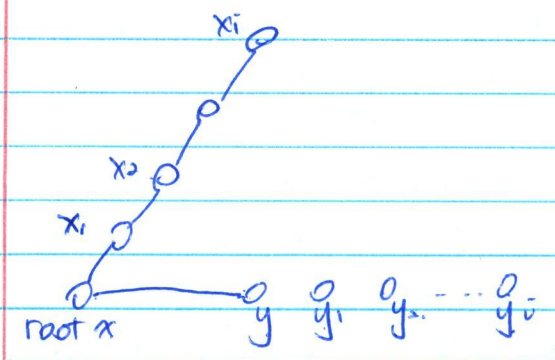
Obviously, $w(T)'$ is not the same as $w(T)$ and is less than $w(T)'$.
This contradicts the assumption made in the beginning.
Thus, the old Minimal Spanning Tree is no longer the Minimal Spanning Tree.

Q2: (a) For graph G with n vertices, to form a minimal cost tree we need $(n-1)$ edges with minimal cost.
From the problem, it is assumed that the $n-1$ edges of minimal cost form a tree T.
Then T is the minimal cost tree.

(b) Assumed that the edges are given in an array in increasing order of weight. And the Algorithm stops when it finds the MST.
Just apply the Union-Find introduced in the class.
Edges: $n-1$ Vertices: n
Time: $O(n-1)\ln n \rightsquigarrow O(n \ln n)$

(c) Dumb kruskal without the SIZE function

Time: $O(n^2)$



The inside while loop climb to roots will take $O(n)$ time

while $x \neq \pi(x)$.

$x \in \pi(x)$.

This runs $(n-1)$ time in worst case.

Thus, for $I = 1$ to E $O(n-1)$

while ... } $O(n-1)$

while ...

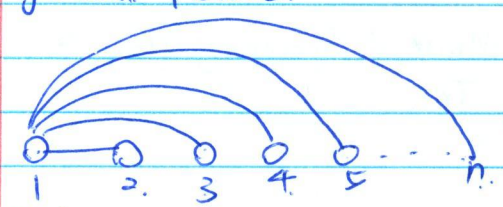
$O(n-1)^2 \sim O(n^2)$

Q3. Vertex set $\{1 \cdots n\}$

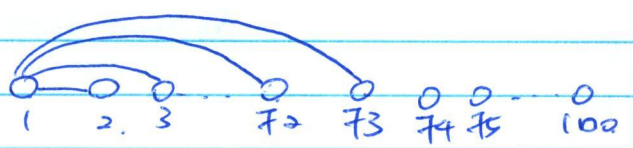
order of the weights $\{1,2\}, \{2,3\}, \{3,4\} \cdots \{n-1, n\}$

(a) $n = 100$. stop when $\{72, 73\}$ has been processed.

general picture:



stop when $\{72, 73\}$ processed:



initially $size[1] = 1$, $\pi[1] = 1$

when $\{1,2\}$ has been processed.

$size[2] = 2$, $\pi[2] = 1$

$size[3] = 3$, $\pi[3] = 1$.

$size[4] = 4$, $\pi[4] = 1$

⋮

$size[72] = 72$, $\pi[72] = 1$

stop — $size[73] = 73$, $\pi[73] = 1$.

$size[74] = 1$, $\pi[74] = 74$.

⋮

$size[100] = 1$, $\pi[100] = 100$.

(b) Time $O(n)$

Initialization all $\pi(x) = x$ size $(x) = 1$.

for $I = 1$ to I. { $O(n)$.

$x \leftarrow x[I]$ $y \leftarrow y[I]$

while $x \neq \pi(x)$
$x \leftarrow \pi(x)$.
while $y \neq \pi(y)$.
$y \leftarrow \pi(y)$.
} $O(1)$ *

if $x \neq y$
reset $\pi(y) \leftarrow x$
size$(x) \leftarrow$ size(x) + size(y)
} $O(1)$.

}

*. Because all previous vertex have already been assigned the same parent, root, $\pi(x_{i-1}) = \pi(x_{i-2})$.
Thus, x will just be assigned once in the while loop.
~~Same thing applies to y in the while too.~~
For y, it is a new vertex, whose root is itself.
$y = \pi(y)$ all the time, so it will never go into the while loop.

Thus the total time searching/going down to roots is $O(1)$.