HOMEWORK 1

Problem 1

Soln:

let the given array is $A[1 \ldots n]$
and k = Order Statistic
If we apply black-box subroutine on A, it will return $n/2$ element. If $k=n/2$, it will return $n/2$th element of the array, ie, $A[n/2]$, else we divide A into two groups; one group will have elements less than $A[n/2]$ named as A_1 and other group will have elements greater than $A[n/2]$ named as A_2.
If $k < n/2$, we will try to find the order statistic for the k^{th} element in A_1.
If $k > n/2$, we will fetch the order Stastic for the $\left(\frac{n}{2} - k\right)^{th}$ element in A_2.
Below are the pseudocode using procedures:-

```
SELECTION (A, k)
        BLACK_BOX(A)
        IF  k=n/2
            then return A[n/2]
        DIVIDE (A)
        IF  k<n/2
            SELECTION (A1, k)
        ELSE SELECTION (A2, n/2-k)
END SELECTION
```

Using black-box subroutine, the cost in computing the median = $O(n)$

Cost in dividing the array = $O(n)$

Suppose $R(n)$ will be the cost of computing k^{th} order statistic then

Total running time $R(n) \leq cn + R(n/2)$

$$R(n) \leq cn + R(n/2)$$
$$\leq c\left(n + \frac{n}{2} + R\left(\frac{n}{4}\right)\right)$$
$$\leq c\left(n + \frac{n}{2} + \frac{n}{4} + R\left(\frac{n}{8}\right)\right)$$
$$\leq c\left(n + \frac{n}{2} + \frac{n}{4} + \frac{n}{8} + \cdots R(1)\right)$$
$$\leq 2cn$$
$$= O(n)$$

Problem 2:

(a)

Solution: Suppose x_k is the median of n distinct elements $x_1, x_2, \cdots x_n$.

According to definition given, if x_k is greater than $\left\lfloor \frac{n+1}{2} \right\rfloor - 1$ other elements x_i then, Sum of weights of elements less than x_k will be :

$$\sum_{x_i < x_k} w_i = \frac{1}{n} \cdot \left(\left\lfloor \frac{n+1}{2} \right\rfloor - 1 \right)$$

$$= \frac{1}{n} \cdot \left\lfloor \frac{n+1-2}{2} \right\rfloor = \frac{1}{n} \cdot \left\lfloor \frac{n-1}{2} \right\rfloor$$

$$\leq \frac{n-1}{2n}$$

$$< \frac{n}{2n}$$

$$< \frac{1}{2}$$

If x_k is lesser than exactly $n - \left\lfloor \frac{n+1}{2} \right\rfloor$ other elements then

$$\sum_{x_i > x_k} w_i = \frac{1}{n} \cdot \left(n - \left\lfloor \frac{n+1}{2} \right\rfloor \right)$$

$$= 1 - \frac{1}{n} \cdot \left\lfloor \frac{n+1}{2} \right\rfloor$$

$$\leq 1 - \left(\frac{1}{n} \right) \left(\frac{n}{2} \right)$$

$$\leq \frac{1}{2}$$

So, we can see that x_k is also the weighted median.

(b).

Soln: To compute the weighted median of n elements, we will first sort the elements, then sum up the weights of elements till the median found.

Below are the pseudo code for this:

```
WEIGHTED-MEDIAN(A)
    K ← 1
    S ← 0          S = Total weight of all x_i < x_K
    while  S + ω_K < 1/2
    do  S ← S + ω_K
        K ← K + 1
    return x_K
```

S is the sum of weights of all elements less than x_K then

$$S = \sum_{x_i < x_K} \omega_i$$

By Induction, we can prove that this is true.

In the first iteration, $S = 0$, therefore the best case is true, since the list is sorted $\forall\, i < k$, $x_i \leq x_k$.

Since in every iteration through the loop S increases by the weight of next element therefore by Induction, S is correct.
Since the sum of weights of all elements is 1, the loop is guaranteed to terminate.
Now, we can prove that x_k is the weighted median using the definition of weighted median, when the loop Ends.
Suppose S' = value of S at the start of the next to last iteration of the loop

$$\therefore S = S' + w_{k-1}$$

Since next to last iteration is not following the termination condition,

$$\therefore S' + w_{k-1} < \frac{1}{2}$$

$$\sum_{x_i < x_k} w_i = S < \frac{1}{2}$$

It is still true if the loop has zero iterations since $S = 0$ which is less than $1/2$.
∴ This proves the first condition for being a weighted median.
Now for the second condition, the sum of weights of elements greater than

x_k is :

$$\sum_{x_i > x_k} \omega_i = 1 - \left(\sum_{x_i < x_k} \omega_i \right) - \omega_k$$

$$= 1 - s - \omega_k$$

From the loop condition,

$$\sum_{x_i < x_k} \omega_i = s \geq \frac{1}{2} - \omega_k$$

$$-s \leq \frac{-1}{2} + \omega_k$$

$$1 - s - \omega_k \leq \frac{1}{2}$$

$$\sum_{x_i > x_k} \omega_i \leq \frac{1}{2}$$

Therefore x_k also satisfies the second condition for being the weighted median.

∴ x_k is the median and the algorithm is correct.

Now, the running time for this algorithm = time to sort the array + time to find the median

$= O(n \lg n) + O(1)$ → $O(n)$ iterations require $O(1)$ time per iteration

$= O(n \lg n)$.

(c)

Soln:

The weighted median can be computed in $\Theta(n)$ worst-case time using a linear-time median algorithm. Like binary search, it will first compute the median and recurses on half of the input that contains the weighted median.

Below is the pseudo code for this:

```
1   LINEAR-TIME-WEIGHTED-MEDIAN (A, l)
2       n ← length[A]
3       m ← MEDIAN[A]
4       B ← Φ                    where B = {A[i] < m}
5       C ← Φ                    where C = {A[i] ≥ m}
6       w_B ← 0                  where w_B = total weight of

7       if length[A] = 1
8         then return A[1]
9       for i ← 1 to n
10        do if A[i] < m
11             then w_B ← w_B + w_i
12                  Append A[i] to array B
13             else Append A[i] to array C
14      if l + w_B > 1/2          where weighted median e
15      then LINEAR-TIME-WEIGHTED-MEDIAN (B, l)
16      else LINEAR-TIME-WEIGHTED-MEDIAN (C, w_B)
```

LINEAR-TIME-WEIGHTED-MEDIAN(A, l) is the initial call. An array A contains median of initial input and l is the total weight of elements of initial input that are less than all the elements of A. B will have all the elements less than the median. C contains all elements greater than or equal to median, w_B is the total weight of elements in B.

The weighted median y of the initial A is always present in the recursive call of A and l is the total weight of all elements x_i less than all the elements of A. For initial call, this precondition is partially true. By Induction, we can prove that the precondition is true for every recursive call.

First assume the case where $l + w_B > 1/2$

Since y must be in A, y must be either in B or C (line 14).

Since total weight of all elements less than any element in C is greater than $1/2$ then the weighted median cannot be in C, it must be in B. Also, we have not discarded any element less than any element in B, so l is correct & precondition is satisfied.

Now if $l + w_B \leq 1/2$ then y must be in C. All elements of C are greater than all elements of B, so total weights of elements less than the elements of C is $l + w_B$ and the precondition is also satisfied.

∴ By Induction, the precondition is always true.

The size of A decreases for every recursive call, the algorithm will end. When the algorithm ends, the result is correct.

∵ the weighted median is always in A,

∴ in case of only one element leftover, that will become the weighted median.

Time for computing the median and dividing A into B and C = $\Theta(n)$

Each recursive call reduces the size of array from n to $\lceil n/2 \rceil$.

$$\therefore T(n) = T(n/2) + \Theta(n)$$
$$= \Theta(n)$$

(d)

Soln:

$$C(p) = \sum_{i=1}^{n} w_i d(p, p_i)$$

Rewriting C(p) as the sum of the cost contributed by points less than p and points

greater than p :

$$C(p) = \left(\sum_{p_i<p} w_i (p - p_i)\right) + \left(\sum_{p_i>p} w_i (p_i - p)\right)$$

If $p = p_k$ for some k, then the point does not contribute to cost. Because of $\lim_{p \to x} C(p) = C(x) \; \forall \; x$, this function is continuous.

If we take derivative w.r.t. p :

$$\frac{dc}{dp} = \left(\sum_{p_i<p} w_i\right) - \left(\sum_{p_i>p} w_i\right)$$

for $p = p_i$ for some i, this derivative is undefined. Also, $\frac{dc}{dp}$ is non-decreasing function, since as p increases, the no. of points $p_i < p$ cannot decrease.

for $p < \min(p_1, p_2, \ldots, p_n)$, $\frac{dc}{dp} < 0$

for $p > \max(p_1, p_2, \ldots, p_n)$, $\frac{dc}{dp} > 0$

$\therefore$ there will be some point p^* where

$\frac{dc}{dp} \leq 0$ for $p < p^*$ and

$\frac{dc}{dp} \geq 0$ for $p > p^*$ and

this point is a global minimum.

For all $p < y$ where p is not the weighted median and $p \neq p_i$ for some i,

$$\sum_{p_i<p} w_i < \sum_{p_i>p} w_i$$

$$\Rightarrow \quad \frac{dc}{dp} < 0$$

Likwise, for $p > y$, where p is not the weighted median and $p \neq p_i$ for some i,

$$\sum_{p_i<p} w_i > \sum_{p_i>p} w_i$$

$$\Rightarrow \quad \frac{dc}{dp} > 0$$

~~For $p = p_i$ where~~ for some i and $p \neq y$ where $p = p_i$, both left and right limit of $\frac{dc}{dp}$ will have the same sign.

i.e. $c(p) > c(y)$ for all p that are not the weighted median.

So the weighted median y is a global median.

(e)

Soln: Let the soln be $p = (p_x, p_y)$

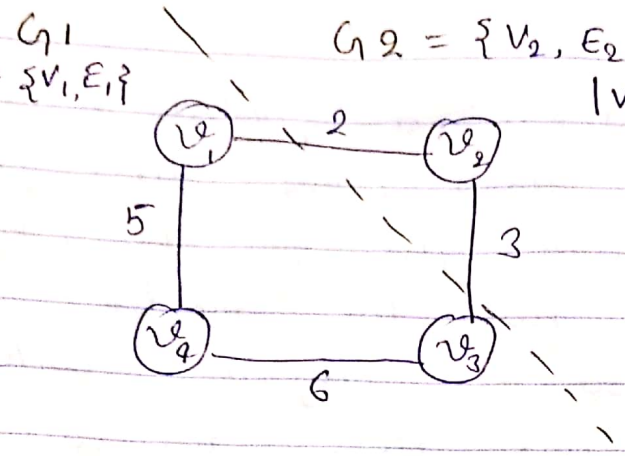
Writing the cost function as sum of two one-dimensional post office location cost functions :-

$$g(p) = \left(\sum_{i=1}^{n} w_i |x_i - p_x|\right) + \left(\sum_{i=1}^{n} w_i |y_i - p_y|\right)$$

$\frac{\delta g}{\delta p_x}$ does not depend on y-coordinates of the input points and has the same form as $\frac{dc}{dp}$ from the previous part using only the x coordinates as input. Likewise, $\frac{\delta g}{\delta p_x}$ depends only on the y-coordinates.

∴ To minimize $g(p)$, we will minimize the cost for the two dimensions independently. Let p_x be the soln to the one-dimensional post-office location problem for inputs $x_1, x_2, x_3, \ldots, x_n$ and p_y be the soln to the one-dimensional post-office location problem for inputs $y_1, y_2, y_3, \ldots, y_n$.

(3) Soln:



Consider the above graph of mst.
In above figure, graph $G = (V, E)$ has 4 vertices and we divided it into two graphs:

G1 with V_1 = [illegible] (V_1, E_1)
& G2 with $V_2 = (V_2, E_2)$
Graph G1 contains $\{v_1, v_4, v_3\}$
Graph G2 contains $\{v_2\}$ Since $|V_2| = 1$

1. The minimum-weight edge crossing the cut (V_1, V_2) has weight 2. The Spanning tree forming by the proposed algorithm is:
$v_1 - v_2 - v_3 - v_4$ has weight 11.
$v_2 - v_3 - v_4 - v_1$ has weight 14.
$v_3 - v_4 - v_1 - v_2$ has weight 13.
$v_4 - v_1 - v_2 - v_3$ has weight 10.

Now, if we look into the above spanning

~~this point is a global minimum~~ For all trees, the first path $v_1 - v_2 - v_3 - v_4$ will give you the weight 11 which is not the ~~shortest~~ smallest weight, so the algorithm will fail here in this case. Here, the algorithm traverses through each edge in a way that it form acyclic and includes all the vertices. Out of four ~~paths~~ paths, ~~mentioned~~ mentioned above, only one path will be the MST which has smallest weight. In this example, path $v_4 - v_1 - v_2 - v_3$ will be the minimum spanning tree since it has the smallest weight. The algorithm will fail to produce a minimum spanning tree, if the partition is done in worst-case manner ie, when we choose graph with highest weight ie,

$$v_2 - v_3 - v_4 - v_1 = 14$$

2.

Ans: If the ~~partition~~ partition is always done in best-case manner, the algorithm will always produce a Minimum spanning tree. In the above example, the ~~graph~~ path which will have the lowest

or smallest weight will consider as a best-case manner and in this case, the path:

$v_4 - v_1 - v_2 - v_3$ having weight 10 will produce the minimum spanning tree. However the percentage of finding MST is really low which is 25% for the graph proposed. Hence, it gives really a low scope of finding the correct graph.

(4.)

Soln: Let $G = (V, E, C)$ be a weighted undirected graph where all the costs C_e for $e \in E$

Let T be a minimum spanning tree in

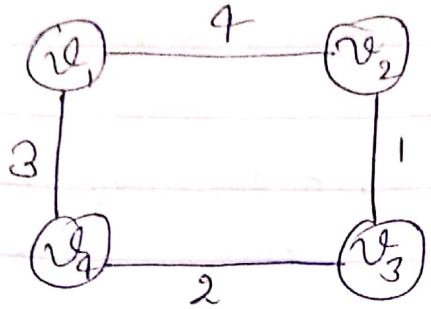
$$G = (V, E, c).$$

Now let replace the cost of each edge $e \in E$ by $C_e' = C_e^2$ for graph $G' = (V, E, C')$ of graph G

So, if T is the MST then T will also be the MST of graph G'.

If we have graph G with n vertices & (n-1) edges, then incrementing the cost ~~by~~ ~~square~~ of edges by square will increase the cost of every spanning tree in the same order of square.

example:



For the above graph, the MST will be $v_2 \to v_3 \to v_4 \to v_1$

The edges covered will be $\{(v_2, v_3), (v_3, v_4), (v_4, v_1)\}$ having cost $= 1 + 2 + 3 = 6$.

Now, when we square the cost of edges, the cost will changed to

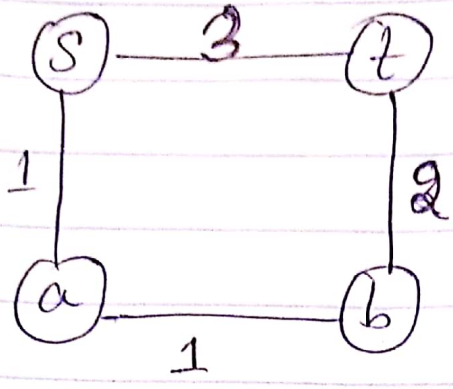
$1^2 + 2^2 + 3^2 = 1 + 4 + 9 = 14$

So, if we find the MST of a graph G, it will remain the MST of the graph G' even if we square the edges in G'. (cost of MST of G will always be smaller if we increase the cost of each edge by square).

Part II →

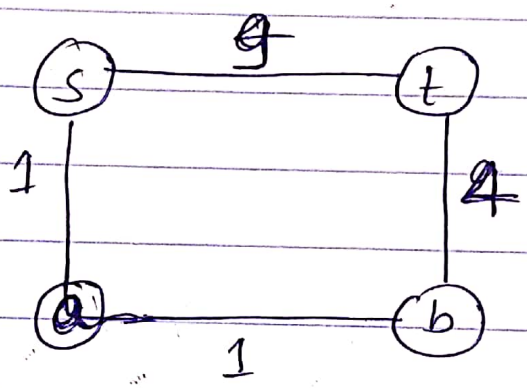
If P is the shortest s-t path in G, it is not necessary that it will be the shortest path in G'.

Lets' consider the below graph:



Here, we can easily see that the shortest path will be s-t with cost 3.

But, when we square the cost of edges, we will get the below graph



Now, the shortest path will change & it will become s-a-b-t having cost 6.

Hence, it is proved that the shortest path will change in G' after squaring the edges in G.