Ques 1.1) D1) $f(n) = O(g(n))$ if there exists a constant c st for all $n \in \mathbb{N}$, the inequality $f(n) \leq c\,g(n)$ holds.

D2) $f(n) = O(g(n))$ if there exists constant n_0 and c st for all $n \geq n_0$, the inequality $f(n) \leq c\,g(n)$ holds.

To prove) D1 and D2 are equivalent.

Fist we will show D1 $\Rightarrow$ D2

Then we have $f(n) \leq c\,g(n)$ for all $n \in \mathbb{N}$ and $f(n)$, $g(n)$ are +ve function that means it holds for all natural numbers then it will definitely holds for some $n \geq n_0$ [let n_0 be 0].

Which implies if D1 holds then D2 will also holds.

Second we have to show D2 $\Rightarrow$ D1

D_2 says $f(n) = O(g(n))$ if there exists constant n_0 and c st for all $n \geq n_0$, the inequality $f(n) \leq c\,g(n)$ holds.

n_0 = natural number

Case I) If $n_0 = 0$ Then $n \geq 0$ so n holds for all natural numbers, so we can

$h \in \mathbb{N}$ which implies D1,

So we are done.

Case II) If $h_0 \neq 0$ [h_0 - natural number]

Then let have a constant d such that d is $\max\{f(h), g(h)\}$ for all h less than h_0, and obviously ($h_0 \geq 0$) h_0 = natural number.

Then $f(h) = \frac{f(h)}{g(h)} g(h) \leq d\, g(h)$

Thus $\boxed{f(h) \leq d\, g(h) \text{ for } h < h_0}$ ①
$f(h) = O[g(h)]$ for $h < h_0$

And we know D2 holds for

$\boxed{f(h) \leq c\, g(h),\ h > h_0}$. ②

Now By 1 and 2 we can conclude

$f(h) \leq c'\, g(h)$ $\quad c' = \max(c, d)$

$\forall\ h \in \mathbb{N}$

That implies $f(h) = O(g(h))$ for all $h \in \mathbb{N}$

Thus $D_2 \Rightarrow D_1$ Hence proved

Ques 1)

1.1

ii) a) $n^2 = O(n^2 \ln n)$ True

let $f(n) = n^2$ and $g(n) = n^2 \ln n$

We know by ~~defination~~ definition for the big - Oh notation that let $T(n)$ be any function, given $T(n)$ and $f(n)$, we say that $T(n) = O[f(n)]$ if $\exists\ n_0 \in Z^+$ and $c > 0$ such that $T(n) \leq c f(n)$ for all $n \geq n_0$.

Now $f(n) = n^2$

$g(n) = n^2 \ln(n)$

$$n^2 \leq c n^2 \ln(n)$$

$$\frac{1}{\ln(n)} \leq c$$

as $\ln(n)$ is increasing from $[1, \infty)$ $\frac{1}{\ln(n)}$ will be decreasing in $[1, \infty]$

$\ln(n)$ at $n = 1$ is 0 so $\frac{1}{\ln(n)} \to \infty$

and $\frac{1}{\ln(\infty)} = 0$, so we can say $\ln(n)$ is bounded

$\frac{1}{\ln(n)} \leq \frac{1}{\ln(2)}$ for all $n \geq 2$

Thus we have prove $f(n) \leq c\,g(n)$ for $n \geq n_0$

Above $n_0 = 2$ $\quad c = \frac{1}{\ln(2)}$

b) If $f(n) = O(g(n))$ then $2^{f(n)} = O(2^{g(n)})$ False

Let $f(n) = 2n$ and $g(n) = n$

Then $f(n) \leq c\,g(n)$

$2n \leq cn$

$2 \leq c$

So $f(n) \leq c\,g(n)$ for all n and let c be any one fixed greater than 2.

Let $c = 3$

so $f(n) = O(g(n))$

Now $2^{f(n)} = 4^n = 2^n 2^n$

$2^{g(n)} = 2^n$

if $2^{f(n)} = O(2^{g(n)})$ there would exist n_0 and c such that $n \geq n_0$ implies

$2^n 2^n = 2^{2n} \leq c\,2^n$,

so $2^n \leq c$ for $n \geq n_0$ which is not possible since c is constant

Hence we have prove $f(n) = O(g(n))$ does not implies $2^{f(n)} = O(2^{g(n)})$.

c) To prove $\sum_{i=1}^{n} i^8 = \Theta(n^9)$ True

let $f(n) = \sum_{i=1}^{n} i^8$

$g(n) = n^9$

Then $\sum_{i=1}^{n} i^8 = 1^8 + 2^8 + 3^8 + \_\_\_ + n^8$ then we can say each term will be less than n,

so $\sum_{i=1}^{n} i^8 \leq \sum_{i=1}^{n} n^8$

$f(n) \leq n^9$

Therefore There exists ~~$n_0 = 1$ and~~ constant $c = 1$ s.t $f(n) \leq c\,g(n)$ $\forall$ $n \geq 1$ $(n_0 = 1)$

$\therefore f(n) = O[g(n)]$

$$\boxed{\sum_{i=1}^{n} i^8 = O(n^9)} \quad \text{---} (1)$$

for $\Theta(g(n))$ $f(n)$ must also be $\Omega(g(n))$

$\sum_{i=1}^{n} i^8$ will be greater than $\sum_{\lfloor \frac{n}{2} \rfloor + 1}^{n} i^8$

↓

by replacing $i=1$ by $\lfloor \frac{n}{2} \rfloor + 1$ we are taking only half terms so it will be less than $\sum_{i=1}^{n} i^8$

$$\sum_{i=1}^{n} i^8 \geq \sum_{i=\lfloor \frac{n}{2} \rfloor + 1}^{i=n} i^8 \geq \sum_{i=\lfloor \frac{n}{2} \rfloor + 1}^{i=n} \left(\frac{n}{2}\right)^8$$

↓

replacing each term by $\left(\frac{n}{2}\right)$ will also be smaller.

$$\geq \sum_{i=(\lfloor \frac{n}{2} \rfloor + 1)}^{n} \left(\frac{n}{2}\right)^8$$

$$= \left(\frac{n}{2}\right)^8 \left[n - \lfloor \frac{n}{2} \rfloor \right]$$

as $\lfloor n \rfloor \leq n$ and $\lfloor n \rfloor > n-1$

so $\frac{n}{2} - 1 < \lfloor \frac{n}{2} \rfloor \leq \frac{n}{2}$

$$1 - \frac{n}{2} > -\lfloor \frac{n}{2} \rfloor \geq -\frac{n}{2}$$

$$1 + \frac{n}{2} > n - \lfloor \frac{n}{2} \rfloor \geq \frac{n}{2}$$

$$\geq \left(\frac{n}{2}\right)^8 \frac{n}{2}$$

$$= \frac{n^9}{2^9}$$

So $\sum_{i=1}^{n} i^8 \geq \frac{1}{2^9} n^9$ (n = Natural no)

Therefore there exists $c = \frac{1}{2^9}$ and such that $f(n) \geq c\, g(n)$ $\forall$ $n \geq n_0$ $(n_0 = 1)$

$$\therefore \boxed{\sum_{i=1}^{n} i^8 = \Omega(n^9)} \quad \text{---} \quad (2)$$

By 1 and 2 we conclude

$$\boxed{\sum_{i=1}^{n} i^8 = \Theta(n^9)}$$

d) To prove $\sum_{i=1}^{n} \sqrt{i} = \Theta(n^{\frac{3}{2}})$ True

Let $f(n) = \sum_{i=1}^{n} \sqrt{i}$

and $g(n) = n^{\frac{3}{2}}$

Then $\sum_{i=1}^{n} \sqrt{i} = \sqrt{1} + \sqrt{2} + \text{---} \sqrt{n}$ then

Each term will be less than $\sqrt{n}$

so $\sum_{i=1}^{n} \sqrt{i} \leq \sum_{i=1}^{n} \sqrt{n}$

$\leq n\sqrt{n}$

$\leq n^{\frac{3}{2}}$

Therefore there exists ~~$n=1, n_0=1$~~ constant $c=1$ st

$$f(n) = O(g(n)) \quad \forall\ n \geq 1,\ n_0 = 1$$

$$\therefore \boxed{\sum_{i=1}^{n} \sqrt{i} = O(n^{\frac{3}{2}})} \quad \text{---} (1)$$

for $\Theta(g(n))$ $f(n)$ must also be $\Omega(g(n))$

$\sum_{i=1}^{i=n} \sqrt{i}$ will be greater than $\sum_{i=\lfloor \frac{n}{2} \rfloor + 1}^{i=n} \sqrt{i}$

by replacing $i=1$ by $\lfloor \frac{n}{2} \rfloor + 1$ we are talking only half terms so it will be less than $\sum_{i=1}^{i=n} \sqrt{i}$

$$\sum_{i=1}^{n} \sqrt{i} \geq \sum_{i=\lfloor \frac{n}{2} \rfloor + 1}^{i=n} \sqrt{i} \geq \sum_{i=\lfloor \frac{n}{2} \rfloor + 1}^{i=n} \sqrt{\frac{n}{2}}$$

as $n-1 < \lfloor n \rfloor \leq n$

$1 + \frac{n}{2} \geq n - \lfloor \frac{n}{2} \rfloor \geq \frac{n}{2}$

replacing each term by $\frac{n}{2}$ will also be smaller

$$\geq \sum_{i=\lfloor \frac{n}{2} \rfloor + 1}^{i=n} \sqrt{\frac{n}{2}}$$

$$= \sqrt{\frac{n}{2}} \left\{ n - \lfloor \frac{n}{2} \rfloor \right\}$$

$$= \sqrt{\frac{h}{2}}\,\frac{h}{2}$$

$$= \frac{h^{\frac{3}{2}}}{2^{\frac{3}{2}}}$$

So $\boxed{\sum_{i=1}^{h} \sqrt{i} \geq \frac{1}{2^{\frac{3}{2}}} h^{\frac{3}{2}}}$ h = natural no.

Therefore there exists $c = \frac{1}{2^{\frac{3}{2}}}$ St

$f(h) \geq c\, g(h) \quad \forall \quad h \geq 1 \quad (h_0 = 1)$

$$\therefore \boxed{\sum_{i=1}^{h} \sqrt{i} = \Omega\left(h^{\frac{3}{2}}\right)} \quad \text{---} \quad (2)$$

By 1 and 2 we conclude

$$\boxed{\sum_{i=1}^{h} \sqrt{i} = \Theta\left(h^{\frac{3}{2}}\right)}$$

Ques 1.2) Input : Graph $G = (V, E)$

Output : The diameter of graph

The diameter of graph is the maximum distance between any pair of vertices.

Algo)

1) Run BFS with starting vertex s.
2) Let the farthest vertex from s obtained by BFS be u.
3) Run BFS with starting vertex u.
4) Let the farthest vertex from u obtained by BFS be v.
5) Return the distance b/w u and v. This will be diameter of graph.

Running Time Analysis →

Step 1) BFS it takes $O(m+n)$

Step 2) also BFS it takes $O(m+n)$

Total Time $= O(m+n)$

But we are given graph is a tree.

as we know for tree $m = h-1$.

$$= O(m+h) = O(h-1+h) = O(2h-1)$$

$$= O(h)$$

Thus our algo runs in linear time.

proof of correctness →

Suppose u and v are end points of the diameter.
WLOG ; we can assume u and v are unique.

Let s be any vertex

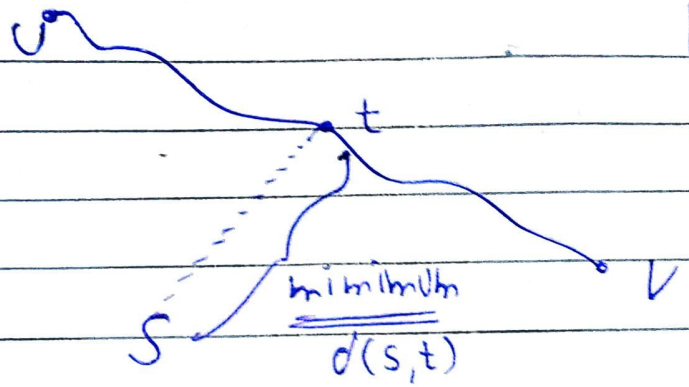
Claim 1) If we run BFS with starting vertex as s then the farthest vertex obtained from s will be either u or v [or both].

proof) By contradiction.

Suppose the farthest vertex from s is not u or v. Let it be c.

Then $d(s,u) < d(s,c)$ ——①

$d(s,v) < d(s,c)$ ——②



[t is vertex on path u,v such that t is nearest to S]

By properties of tree, we can say

$$d(s,u) = d(s,t) + d(t,u)$$
$$d(s,v) = d(s,t) + d(t,v)$$

[If it exists more than one path, we could form a cycle But given graph is tree so cycle can't exist in it]

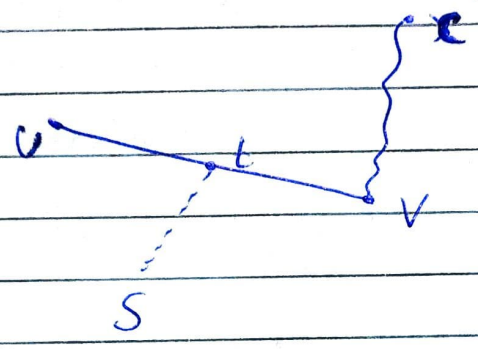
$$\rightarrow d(u,v) = d(t,u) + d(t,v)$$
$$\rightarrow d(u,v) + 2d(s,t) = d(t,u) + d(s,t) + d(t,v) + d(s,t)$$

$$\rightarrow \text{Now } \cancel{d(u,v) + 2d(s,t)} \leq \{ d(s,x) + d(s,t) + d(t,v)$$

~~Now we have to prove claim that~~

Now $d(s,t) \leq d(s,v) + d(s,t)$

Now we claim that $d(c,v) = d(s,t) + d(s,v)$. If this is not the case then it must be less than.
Then there exist a path from c to v not passing through t.



which contradicts, because if it will happen then cycle is formed, Then

$$d(u,v) < d(u,v) + 2d(s,t) < d(t,v)$$

So it assumes $d(u,v)$ is maximal path which exist in the tree, which we have a contradiction.

Hence proved

Ques 3) Input : n-vertex undirected unweighted graph $G = (V, E)$.

Algo

Step 1) Run BFS with starting vertex s.

Step 2) Stores all the vertex at each Level in List $L[i]$.

Step 3) find level j, $1 < j < \frac{n}{2}$ which has only one node. Output that node.

Running time Analysis)

Step 1 is BFS it takes $O(m+n)$

Step 3 Finding Level will take atworst $O(n)$

Thus, Total time $= O(m+n)$

Proof of correctness

Suppose that n-vertex undirected weighted graph $G = \{V, E\}$ contains two nodes s and t st distance between s and t is strictly greater than $\frac{n}{2}$.

To prove) Show that there must exists some node v, not equal to either s or t, such that deleting v from G destroys all paths between s and t.

$|V| = h$

Let T be the BFS tree obtained with s as starting vertex. Then we know BFS will divide the graph into Layers / Levels.

[as we know level[i] is the length of shortest path from s to some node v which is in level[i]].

Given Every path between s and t is strictly greater than $\frac{h}{2}$.

Now let l = length of path between s and t.

$\Rightarrow l \geq \frac{h}{2} + 1$

node t will be at level $\geq l$.

Now if we consider level from 1 through $\frac{h}{2}$ between s and t.

The no of nodes in level 1 through $\frac{h}{2}$ will be atmost $h-2$ [excludes nodes s and t].

Now if each of $\frac{h}{2}$ levels has 2 or more nodes then clearly total no. of nodes will be greater than h.

Which implies there exists atleast one level which has only one node. So if we delete that node that all paths between s and t will be destroyed.

Hence proved

Ques 1.4)

Input → integer L

Output → minimum cardinality collection of coins required to make L paisa change [that is, you want to use as few coins as possible]

Assumption → unlimited supply of coins of each type.

a) The coins are worth 1, 5, 10, 20, 25, 50.

The greedy will not work in this case.

We will prove by counterexample

Suppose L = 40 paisa

Our algo will output

40 = 25 + 10 + 5

3 coins

while Optimum is

40 = 20 + 20

2 coins

Thus our algo is not correct greedy.

b) The coins are worth $2^0, 2^1, 2^2, 2^3, \ldots, 2^{1000}$.
[power of two]

We will prove it now.

Claim) let a_i be no of coins of denomination 2^i [$i = 0, 1, 2 \ldots, k$] $k <= 1000$ used to exchange amount L.
That there is unique optimal sol. that agress it.

Proof let $\{a_0', a_1', a_2', \ldots, a_k'\}$ be the optimum sol.
Let it be O

$$O = \{a_0', a_1', a_2', \ldots, a_k'\}$$

Then $a_i' < 2$ for all $i < k$
[because if $a_i' \geq 2$ then we can replace 2 coins of denomination 2^i by a single coin of 2^{i+1}]

$$A = \{a_0, a_1, a_2, \ldots, a_k\}$$
↙
Our sol.

Now we have to prove that,

$a_i' = a_i$ for all $i \leq k$

We will prove by contradiction

Let us assume $a_j' \neq a_j$ for j be the Largest index.

We have $a_j' < a_j$

Now $L = \sum_{i=0}^{j} a_i 2^i = \sum_{i=0}^{j} a_i' 2^i$

$$L = \sum_{i=0}^{j-1} a_i' 2^i + a_j' 2^j \leq \sum_{i=0}^{j-1} 2^i + a_j' 2^j$$

(as $a_i' < 2$)

$$= \underbrace{\sum_{i=0}^{j-1} 2^i} + a_j' 2^j$$

$= GP$

$$= \left(\frac{2^j - 1}{2-1}\right) + a_j' 2^j$$

$$= 2^j - 1 + a_j' 2^j$$

$$= 2^j [a_j' + 1] - 1$$

$$\leq 2^j a_j' - 1$$

$$\leq \underbrace{2^j a_j} - 1 \quad \text{as } a_j' < a_j$$

$$\leq \left(\sum_{i=0}^{j} a_i 2^i\right) - 1$$

$$= L - 1$$

Which is Contradiction.
Thus proved.

Rubbing time Analysis

Algo

count = 0 Count is number of coins

for i = 1000 to 0

$k = \left\lfloor \frac{L}{2^i} \right\rfloor$

$L = L \bmod 2^i$

count = count + k

We can see that above for loop will max of 1001 times and inside it mod and division takes constant time.

So Total time = O(k)

where k = 1000

in our case

Ques 1.5 Input : It consists of length $l_1, l_2 \dots l_n$ and access probability $p_1, p_2, \dots, p_n$ for n files $F_1, F_2 \dots F_n$.

Problem : To order these files on a tape so as to minize the Expected access time.
If files are stored in order $F_{s(1)}, \dots, F_{s(n)}$, then Expected access time is

$$\sum_{i=1}^{n} \left(p_{s(i)} \cdot \sum_{j=1}^{i} l_{s(j)} \right)$$

Output : order of files.

a) Algo 1 → Order the files from shortest to longest on the tape.
That is, $l_i < l_j \Rightarrow s(i) < s(j)$

The gives algo 1 is not correct.
We will prove using counterexample.
Let $f_1 = (l_1, p_1) = (1, 0.2)$
$f_2 = (l_2, p_2) = (2, 0.7)$
$f_3 = (l_3, p_3) = (3, 0.1)$

Our algo will order files in order

f_1	f_2	f_3

Expected time = $1 \times 0.2 + 0.7 \times \{1+2\} + 0.1 \times \{1+2$
= 2.9

But optimum is

f_2	f_1	f_3

Expected time = $0.7 \times 2 + (1+2) \times 0.2 + 0.1 \times (1+2+3)$
= 2.6 smaller than 2.9

Thus algo is not correct greedy.

b) ~~Algo2~~ → Order the files from most likely to be accessed to least likely to be accessed
That is $p_i < p_j \Rightarrow s_i^{-1} > s_j^{-1}$

The given algo2 is not correct.
We will prove using counterexamples
Let $f_1 = (l_1, p_1) =$ (1, 0.4)
$f_2 = (l_2, p_2) =$ (2, 0.5)
$f_3 = (l_3, p_3) =$ (3, 0.1)

Our algo will order files in order

f_2	f_1	f_3	

Expected time = $0.5 \times 2 + 0.4 \times 3 + 0.1 \times 6$
= 2.8

But optimum is

f_1	f_2	f_3

Expected time = $0.4 \times 1 + 0.5 \times 3 + 6 \times 0.1$
= 2.5 smaller than 2.8

Thus algo is not correct greedy.

c) Algo 3 → Order the files from the smallest ratio of the length over access probability to the largest ratio of the length over access probability. That is,

$$\frac{l_i}{p_i} < \frac{l_j}{p_j} \Rightarrow s_i < s_j$$

Proof) We will prove by contradiction

Let us assume O be a optimum solution, different from greedy approach. ~~There~~ Then there must be two consecutive files that are not in sorted order.

That is

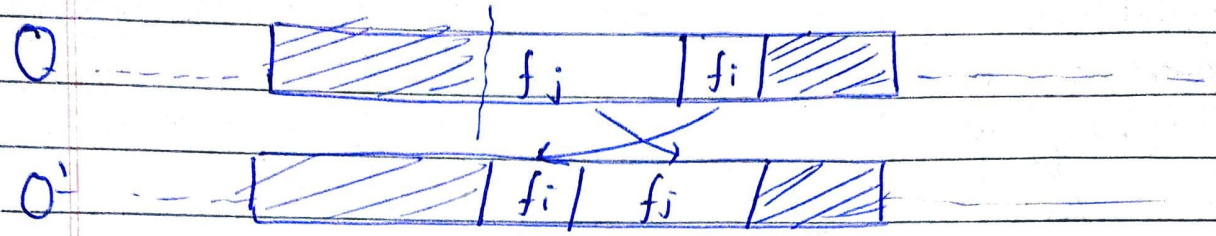
$$O = \{ \ldots\ldots f_j, f_i, \ldots\ldots \} \quad j > i$$

Then $\frac{s_j}{p_j} \geq \frac{s_i}{p_i}$

$$p_i s_j - s_i p_j \geq 0$$

$$p_j s_i - p_i s_j \leq 0$$

Now suppose if we swapped these two files. Then let the order obtained be O'.



We only swap f_i, f_j and remaining files will maintain their order

Then in O' f_j will moves s_i units backward

So ~~cost~~ time to access will increase by $p_j s_i$.

The file f_i will moves s_j units closer to front of tape, so time to excess it decreases by $s_j p_i$.

So cost will get change by

$$\text{Time}(O') - \text{Time}(O) = p_j s_i - p_i s_j \leq 0$$

$\therefore$ $\text{Time}(O') \leq \text{Time}(O)$ which contradicts

Hence proved

Rubbing time Analysis

In Step 1 we are dividing Each l_i by p_i so it takes $O(n)$.

Then we will sort in order of ratio $\frac{l_i}{p_i}$ which will take $O(n \log n)$

Total time $= O(n \log n + n)$

$= \boxed{O(n \log n)}$

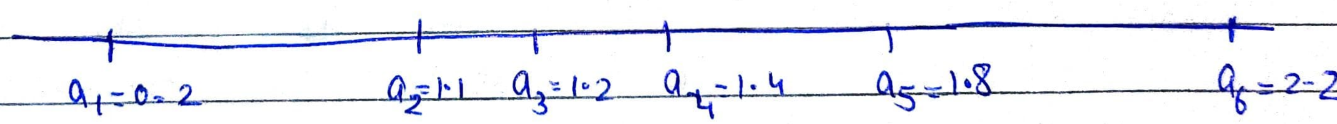
Ques 6) Input : Collection $A = \{a_1, a_2, \ldots, a_n\}$ of n points on the real Line.

Output : Min cardinality collection S of unit intervals that cover Every point in A.

To prove : whether given algo always or disprove produces an optimal sol.

a) Algo 1) let I be the interval that covers the most no. of points in A. Add I to the sol. set S. Then recursively continue on the points in A not covered by I.

The algo is not correct. We will prove by counterexample.



given algo outputs.

| a_1 | | a_2 a_3 a_4 a_5 | | a_6 |

3 intervals

But optimum is 2

| a_1 a_2 a_3 | | a_4 a_5 a_6 |

Hence given algo is not correct.

b) Algo 2) let a_j be the smallest [leftmost] point in A. Add the interval $I = [a_j, a_j + 1]$ to the sol. set S. Then recursively continue on the points in A not covered by I.

We will sort points in ascending order i.e $a_1 < a_2 < a_3 \text{ ---- } < a_n$.

proof) let us assume that there exist an optimal set of intervals which contains the interval $[a_1, a_1 + 1]$.

Let O = optimum set of intervals and there is an interval $[b, b+1]$ which contains a_1 st $b < a_1$. As there are no points to the left of a_1 so the interval $[b, a)$ is empty.

So we can replace interval $[b, b+1]$ by $[a_1, a_1 + 1]$ in O such that After new set of intervals be O_{new} is also optimal $\{ |O| = |O_{new}| \}$.

Thus we can build a optimal sol. of the subproblems (O') after removing all the points in interval $[a_1, a_1 + 1]$.

Then $O' \cup [a_1, a_1+1]$ is optimum.

Similarly we can prove that our greedy works for all the points greater than a_1+1.

Hence proved

Running Time)

Step 1) first we are sorting, it takes - $O(n \log n)$

Step 2) Then we are appending interval of unit length in let say set S starting with left most point it takes atworst $O(n)$.

Total Time $= O(n \log n + n)$

$= \boxed{O(n \log n)}$