01/03/2021 CSPC42 - Algos. 106119100

Rajneesh Pandey

Cycle Test - 1

Question (1) :

as, we have.

```
for (i=1; i<=n*n ; i++) {
        for (j=1 ; j*j <=n ; j++) {
                // const. O(1)
        }}
```

So, inner loop is running for ($\sqrt{n}$) times of outer loop (n^2) times

so,

time complexity = $O(n^2 . \sqrt{n})$.

Question (2)

slowest to fastest growing order,

$$1 < \log n < n \log n < n^2 \log n < n^3 < 2^n < 3^n < n! < n^n$$

Question (3)

we have,

```
Algo. A(n)
{ if (n==2) return 1
  else,
        return (A(n^0.5) + A(n^0.5));
```

here,

$$A(n) = \begin{cases} 1 ; & \text{if } (n=2) \\ A(\sqrt{n})*2 ; & \text{if } (n>2) \end{cases}$$

Recurrance Relation,

So, $\boxed{A(n) = 2A(\sqrt{n})}$; $A(2)=1$

so, $A(n) = 2A(n^{1/2}) = 4A(n^{1/4}) = \cdots\cdots = 2^k A(n^{1/2^k})$

so, when,

$$n^{1/2^k} = 2 \Rightarrow n = 2^{2^k}.$$

So,

$$\log n = 2^k \cdot \log 2.$$

$$\boxed{k = \log\log n}$$

substitute.

now, $A(n) = 2^{\log\log n}$

$$\boxed{A(n) = \log_2 n}$$

Question (4)

we have, $\{2, 8, 7, 1, 3, 5\}$.

Applying Quick sort:

for each unsorted position.

set first element as pivot.

StoreIndex = pivotIndex + 1

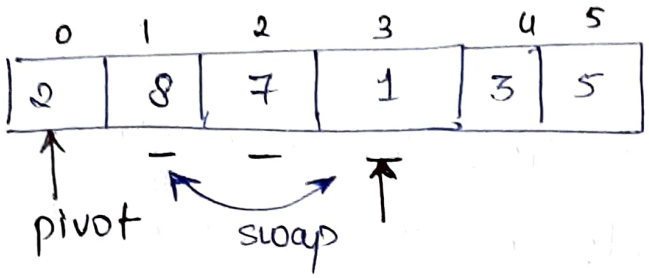
for i = pivotIndex + 1 to rightmostIndex

if element[i] < element[pivot]

swap(i, storeIndex); storeIndex++;

swap(pivot, storeIndex - 1);

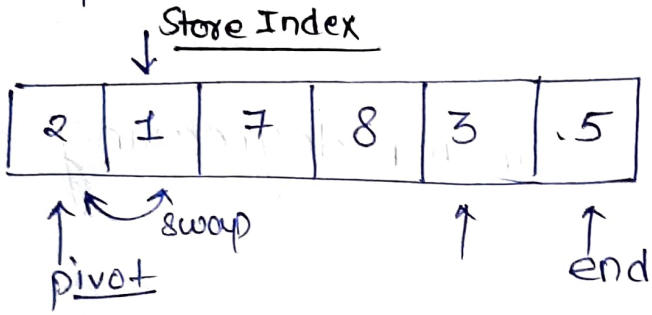
So,



store index = 1

compare $8 < 2$ (pivot), $7 < 2$ (pivot), $1 < 2$ (pivot) → true.

swap index 3 val to store index val.



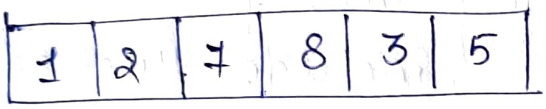
again continues.

loop ends.

So,

swap (pivot val, store_index val)

After one iteration



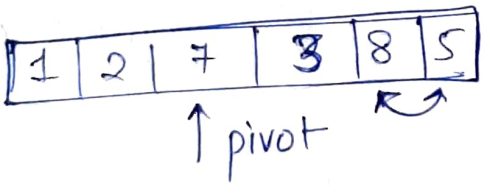
Pivot is at sorted position.

Now,

(7) → new pivot. store index = 3

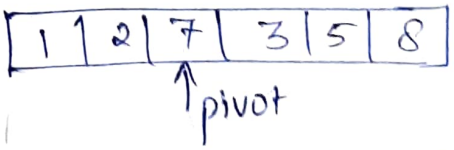
compare $3 < 7$ (pivot) → true

swap.



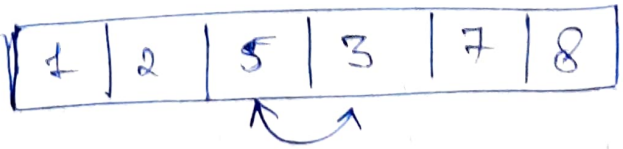
again $(5 < 7)$ true

⇒



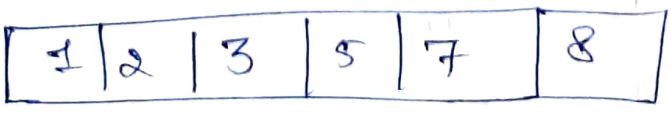
loop ends now, swapping pivot

After two iteration



now, check. $3 < 5$ true.

So, list is sorted



Question 7

A minimum spanning tree for a weighted, connected and undirected graph is a spanning tree with weight less than or equal to the weight of every other spanning tree.

Steps for Kruskals (Algo).

1) Sort all the edges in non-degreasing order of their weight.

2) Pick the smallest edge. check if it forms a cycle with the spanning tree formed so far if cycle is not formed, include this edge. Else discard it.

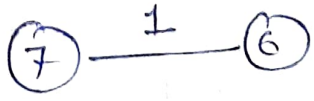
3) Repeate step #2 until there are (V-1) edges in the spanning tree.

tracing the algo :

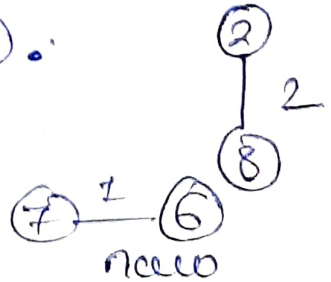
weight	Src	Dest
1	7	6
2	8	2
2	6	5
3	0	1
4	2	5
5	8	6
6	2	3
7	7	8
8	0	7
9	1	2
10	3	4
11	5	4
12	1	7
13	3	5
14		

Now, pick all edges one by one

(1).

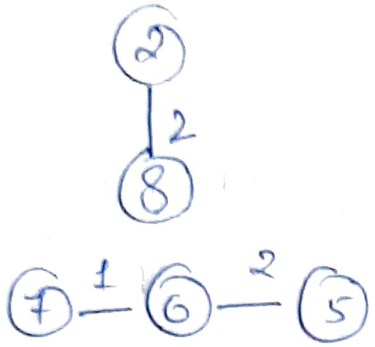


(2).

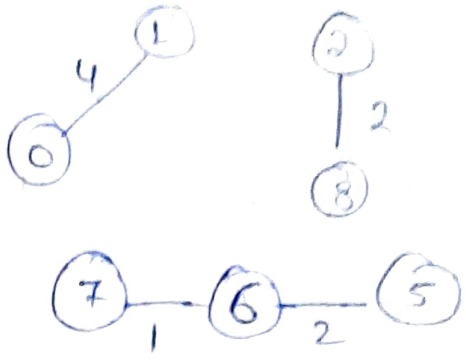


now,

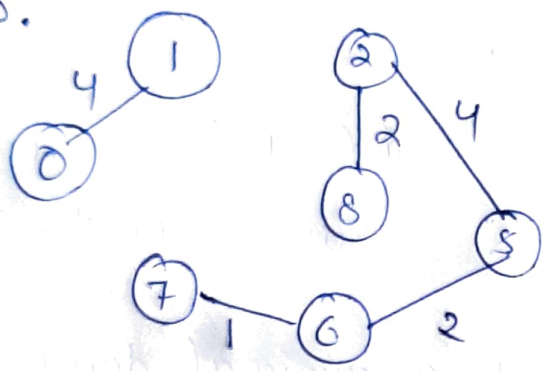
3.



4.

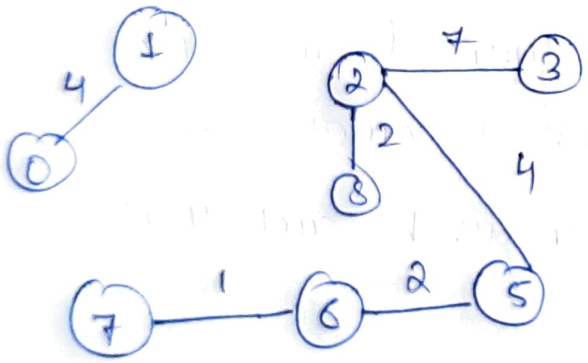


5.



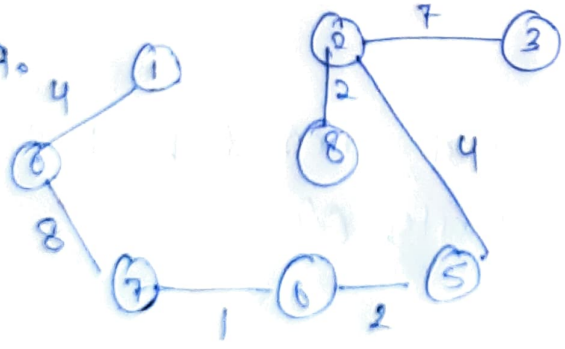
6. pick edge 8-6 but it result in cycle discard it

7.



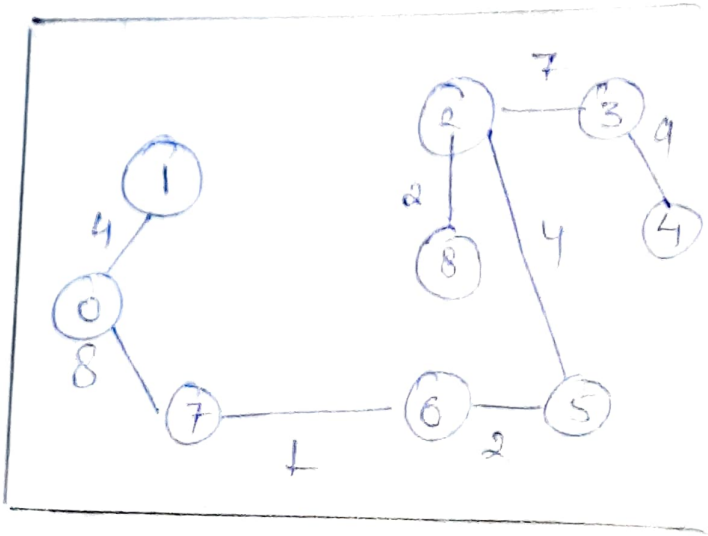
8. again 7-8 forming cycle. discard it

9.



10. (1-2) forms cycle

11. Hence the minimum spanning tree



Question 5

profit :	40	33	11	10	3
Deadline :	2	1	2	3	1
Jobs :	J_5	J_3	J_4	J_1	J_2

value of max deadline = 3.

0 — 1 — 2 — 3

find a time slot : such that slot is empty and is deadline and i is greatest. Put the particular job in this slot and mark it as filled.

so take J_5

0 — 1 $\overset{J_5}{—}$ 2 — 3

take J_3

0 $\overset{J_3}{—}$ 1 $\overset{J_5}{—}$ 2 — 3

J_4 is 2, but it already filled so, ignored

take J_1

0 $\overset{J_3}{—}$ 1 $\overset{J_5}{—}$ 2 $\overset{J_1}{—}$ 3

all the slots are filled, so ignore the rest jobs

optimum solution :

$$\text{profit} = J_3 + J_5 + J_1$$
$$= 33 + 40 + 10$$

profit = 83 .

Question 6

We can see that.

tank will be full for starting $(l+1)$ days because water taken out is less than water being filled.

After that, each day water in the tank will decrease by 1 more lit.

and on $(l+1+i)^{th}$ day $(c-(i)(i+1)/2)$ lit water will will remain before taking dringking water.

Now, we need to find a minimal day $(l+1+k)$, in which even after filling the tank by l liters we have water less than l in tank.

ie $(l+1+k-1)^{th}$ day tank becomes empty so.

our goal is to find minimum k such that

$$c - k(k+1)/2 <= l$$

So, using binary search and then $l+k$ will be our ans. time complexity $O(\log c)$.

```
lo = 0, hi = 1e4,
  while (lo < hi)
    { mid = (lo + hi) / 2 ;
      if ( cumululatesum (mid) >= (c-l))
                hi = mid.
      else,
                lo = mid+1; }
  return (l+lo);
```