Page Genius
Date

Quick Sort.

Quick Sort

12 + X

93 0, 000

Divide - partition the array A[p....q-1] and A[q+1.....r] by ~~recursive calls to quick sort~~. such that each element of A[p....q-1] is less than or equal to A[q], which is in turn, less than or equal to each element of A[q+1...r]. compute index q as a part of partitioning in

→ all numbers are greater than 10

⑩ 80 90 60 30 20

↑
sorted position.

all numbers are less than 9

⑥ ③ ⑤ ④ ② ① ⑨

↓
sorted position

4 6 7 ⑩ 16 12 13 14

↓sorted position

← smaller → larger

Procedure of quick sort.

	0	1	2	3	4	5	6	7	8	9
A	10	16	8	12	15	6	3	9	5	

→ ∞ (maximum number) and will act as end of ~~array~~ list

follows divide and conquer.

~~[illegible]~~ 0 + (255 1a 0) 'b' "~~s~~tring":

10 - select first element as pivot.
pivot = 10
10 should come at place where left hand side of 10 should be smaller than 10 and RHS of 10 should be larger then 10.

∴ i = ~~10~~ ~~(pivot)~~ l
↳ will search for numbers less than 10 (pivot)
↳ at most this will stop at last end

j = ∞ (end)
= h
↳ will search for numbers greater than 10. (pivot
↳ at most it will stop at ~~1st~~ element. pivot element

④ partitioning procedure.

increment i until you find greater than
decrement j untill you find smaller than or equal to pivot

A

10	16	8	12	15	6	3	9	5	∞
	i							j	

since both satisfies condition swap there position

A

10	5	8	12	15	6	3	9	16	∞
0	1	2	3	4	5	6	7	8	9

increment i untill you find element greater than pivo

i = 2 < 10
i = 3 > 10

also decrement j untill smaller than or equal to pivot

A

10	5	8	12	15	6	3	9	16	∞
			i			j			

now we need to swap i to j

	0	1	2	3	4	5	6	7	8	9
A	10	5	8	9	15	6	3	12	16	∞
					i		j			

increment i => i = 4 = $\boxed{15}$ > 10 stop

decrement j => j = 6 = $\boxed{3}$ < 10 stop

and then swap.

A	10	5	8	9	3	6	15	12	16	∞
	0	1	2	3	4	5	6	7	8	9

increment i = 5 = $\boxed{6}$ < 10 ∴ therefore again increment i = 6 = 15 > 10 ∴ stop.

decrement j = 5 = $\boxed{6}$ < 10 stop.

now since i > j (i crossed j) do not interchange. it means we found the position of pivot. the A[j] position will be position of pivot. swap j with pivot.

A	6	5	8	9	3	10	15	12	16	∞
	0	1	2	3	4	5	6	7	8	9
						j				

not sorted — j ↓ sorted — not sorted

now we need to perform quick sort recursively on either side.

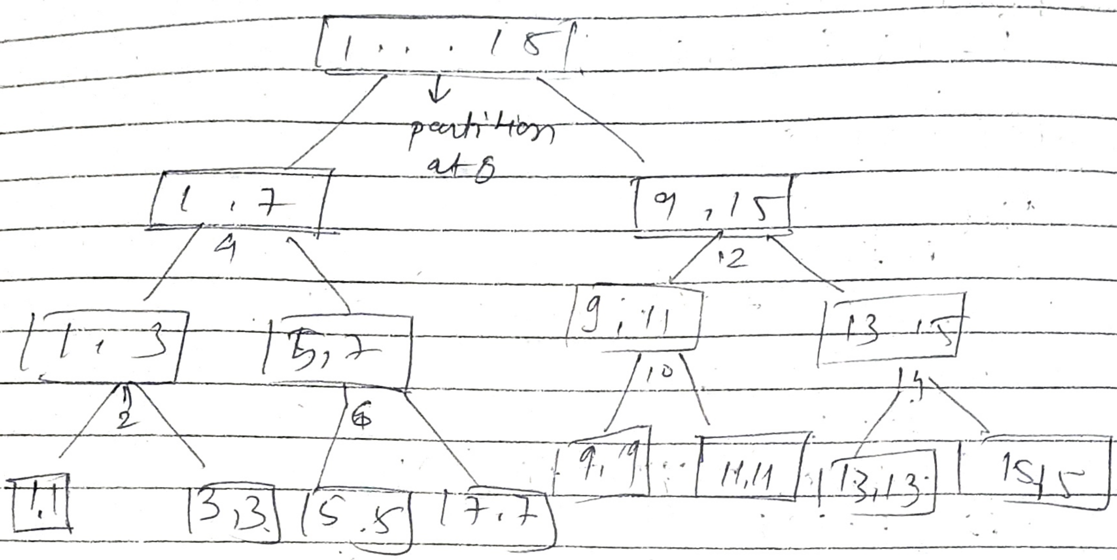
algorithm

```
partition (l, h){
        pivot = A[l]
        i = l; j = h;

    while ( i < j ){
        do {
            i++;
        }while ( A[i] ≤ pivot );
        do {
            j--;
        } while (A[j] > pivot);
        if (i < j){
                swap(A[i], A[j]);
    }

    swap(A[l], A[j])                       // for pivot swap
    return j;
}
```

```
Quicksort(l, h){                              time.
    if (l<h){                              — (n)
        j = partition(l, h);               — almost n every
        Quicksort(l, j);                   —   partition
        Quicksort(j+1, h);                         ↓
    }                                         total level
}                                               logn
                                                 ↓
                                             each level
                                               nlogn
```

Quicksort Analysis:

~~let~~ let take 15 element



∴ time complexity O(n log n)

↗ assume partitioning is always done in middle

Best case   worst case -

17

2 4 8 10 16 ~~18~~ 18

worst case will be that list is already sorted due to this partitioning will alway happening in beginning of list.

$O(n^2)$

$$\left.\begin{matrix} n \\ n-1 \\ n-2 \\ \vdots \\ 1 \end{matrix}\right\} \text{already sorted list}$$

$$= \frac{n(n+1)}{2}$$

$$= O(n^2)$$

improving worstcase → always select middle element

→ randomly select element as pivot.

always worst time taken $O(n^2)$.

it ill take logn to n stack size

↓ ↓

best worst

⊛ Strassen's Algorithm matrix multiplica

$$A \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}_{2\times 2} \times B \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}_{2\times 2} = C \begin{bmatrix} & \\ & \end{bmatrix}_{2\times 2}$$

$$C_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$$

simple algorithm for multiplying.

```
for(i=0; i<n; i++){
    for(j=0; j<n; j++){
        C[i,j]=0
        for(k=0; K<n; k++){
            C[ij] = A[i,k] * B[k,j]
        }
    }
}
```

$O(n^3) \rightarrow n^3$

4 formula ↓

multiplication

$C_{11} = A_{11} B_{11} + A_{12} B_{21}$

$C_{12} = A_{11} B_{12} + A_{12} B_{22}$

$C_{21} = A_{21} B_{11} + A_{22} B_{21}$

$C_{22} = A_{21} B_{12} + A_{22} B_{22}$

now let's see this with 4×4

$$A = \begin{array}{r} \\ \\ A_{21}\left\{ \begin{array}{c} \\ \\ \end{array}\right. \end{array} \left[\begin{array}{cc|cc} \multicolumn{2}{c|}{\overbrace{a_{11} \quad a_{12}}^{A_{11}}} & \multicolumn{2}{c}{\overbrace{a_{13} \quad a_{14}}^{A_{12}}} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ \hline a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & \multicolumn{2}{c}{\underbrace{a_{43} \quad a_{44}}_{A_{22}}} \end{array}\right] , \quad B = \left[\begin{array}{cc|cc} \multicolumn{2}{c|}{\overbrace{b_{11} \quad b_{12}}^{B_{11}}} & \multicolumn{2}{c}{\overbrace{b_{13} \quad b_{14}}^{B_{12}}} \\ b_{21} & b_{22} & b_{23} & b_{24} \\ \hline b_{31} & b_{32} & b_{33} & b_{34} \\ \multicolumn{2}{c|}{\underbrace{b_{41} \quad b_{42}}_{B_{21}}} & \multicolumn{2}{c}{\underbrace{b_{43} \quad b_{44}}_{B_{22}}} \end{array}\right]$$

$$\frac{4}{2} \times \frac{4}{2} \qquad\qquad \frac{4}{2} \times \frac{4}{2}$$

Algorithm –

dimension

matrix addition

```
algorithm MM ( A, B, n ) {
        if n(≤2) {
            C = 4 formula ;
        }
        else {
            mid = $\frac{n}{2}$
            mm ( $A_{11}$, $B_{11}$ $\frac{n}{2}$ ) + mm ($A_{12}$, $B_{21}$, $\frac{n}{2}$)
            mm ($A_{11}$, $B_{21}$, $\frac{n}{2}$ ) + mm ($A_{1}$ $B_{22}$ $\frac{n}{2}$
            mm ( $A_{21}$, $B_{11}$, $\frac{n}{2}$ ) + mm ( $A_{22}$, $B_{21}$, $\frac{n}{2}$ )
            mm ( $A_{21}$, $B_{22}$ $\frac{n}{2}$ ) + mm ($A_{22}$, $B_{21}$ $\frac{n}{2}$)
        }
}
```

Page No - 78

time complexity

$$T(n) = \begin{cases} 1 & n \leq 2 \\ 8\,T\left(\frac{n}{2}\right) + n^2 & n > 2 \end{cases}$$

↓ for adding

↓ for multiplication.

→ masters theorem.

$a = 8$ $f(n) = n^2$

$b = 2$ $= n^k$

$= k = 2$

$\log_b a = \log_2 8 = 3$

$\log_b a > k$

$\Rightarrow \Theta(n^3)$

check page no 79   TH cormen algo.

⊛ Greedy Method

- approach for solving problem as DAC etc.
this method is used for solving optimization problem(which require either minimum result or maximum result).

feasible soln- solving given constraints

optimal soln - satisfying objective of problem

To solve optimisation problem we have
1. greedy method
2. dynamic programming
3. branch and bound.

1. Greedy Method

n=5

a	a_1	a_2	a_3	a_4	a_5
	1	2	3	4	5

```
algorithm greedy (a, n){
    for i=1 to n do{
        x = select (a);
        if feasible(x) then {
            solution = solution + x;
        }
    }
}
```

Fractional knapsack problem.

bag capacity
m = 15
↓
knapsack.

~~object~~	~~0~~	~~1~~	~~2~~	~~3~~	~~4~~	~~5~~	~~6~~	~~7~~
~~profit P~~		~~10~~	~~5~~	~~15~~	~~7~~	~~6~~	~~18~~	~~3~~
~~weights W~~		~~2~~	~~3~~	~~5~~	~~7~~	~~1~~	~~4~~	~~1~~

objects -	1	2	3	4	5	6	7
profit (p)	5	10	15	7	8	9	4
weight (w)	1	3	5	4	1	3	2
p/w	5	3.3	3	1.75	8	3	2

w = 15
weight for bag

there are three ways we can pick the objects

1.) by selecting most profitable objects
2.) by selecting least weighted object
3) - by selecting the object whose profit to weight ratio is higher

1. by max profit

object	profit	weight	remaining weight
3	15	5	15-5=10
2	10	3	10 - 3 = 7
6	9	3	7 - 3 = 4
5	8	1	4 - 1 = 3
4	$\boxed{7}$	$\boxed{4}$	here weight is 4 but space left is 3 so we can't select this one
	$7 \times \frac{3}{4} = \frac{21}{4}$	3	but as it's fractional this object is divisible
	$= 5.25$		w = 0

Changed.

now check total profit i.e. 47.25

2) by min weight

object	profit	weight	remaining weight
1	5	1	15-1=14
5	8	1	14-1=13
7	4	2	13-2=11
2	10	3	11-3=8
6	9	3	8-3=5
4	7	4	5-4=1
3			

now remaing weight is 1
but object 3 is having 5 weight

$\boxed{15 \times \frac{1}{5} = 3}$ $\boxed{1}$ $1-1=0$

now total profit. 46

3) by max ratio of profit by weight

object	profit	weight	remaining weight
5	8	1	15-1=14
1	5	1	14-1=13
2	10	3	13-3=10
3	15	5	10-5=5
6	9	3	5-3=2
7	4	2	2-2=0

total profit : 51

by following third approach we got the max profit.

④ Job sequencing with dead lines

n=5   # every job need 1 unit of time   # maximization problem.

jobs	J_1	J_2	J_3	J_4	J_5
profit	20	15	10	5	1
deadlines	2	2	1	3	3

need to pick jobs which gives most profit.

slots hours for job   0 — 1 — 2 — 3

1) select in order of profit

$$0 \xrightarrow{J_2} 1 \xrightarrow{J_1} 2 \xrightarrow{J_4} 3$$

set $\{J_2, J_1, J_4\}$

sequence $J_1 \rightarrow J_2 \rightarrow J_4$

$\underbrace{J_2 \rightarrow J_1}_{} \rightarrow J_4$

both are
ready
to wait
for 2 hour.

total profit   20 + 15 + 5 = 40

Optimal Merge Pattern · greedy method

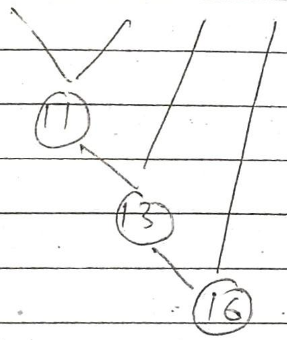
let have two sorted list.

A	B	C
3	5	3
8	9	5
12	11	8
20	16	9
m ↗	n ↗	11
		12
		16
		20

total time taken
O(m+n)

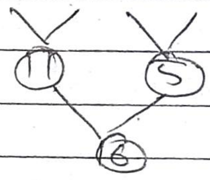
let's take this merging for more than two list

list	A	B	C	D
size	6	5	2	3



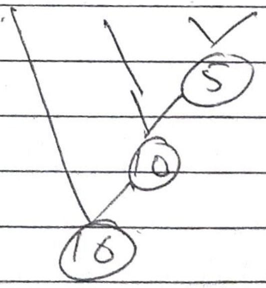
total = 40

A	B	C	D
6	5	2	3



11+16+5
= 32

A	B	C	D
6	5	2	3



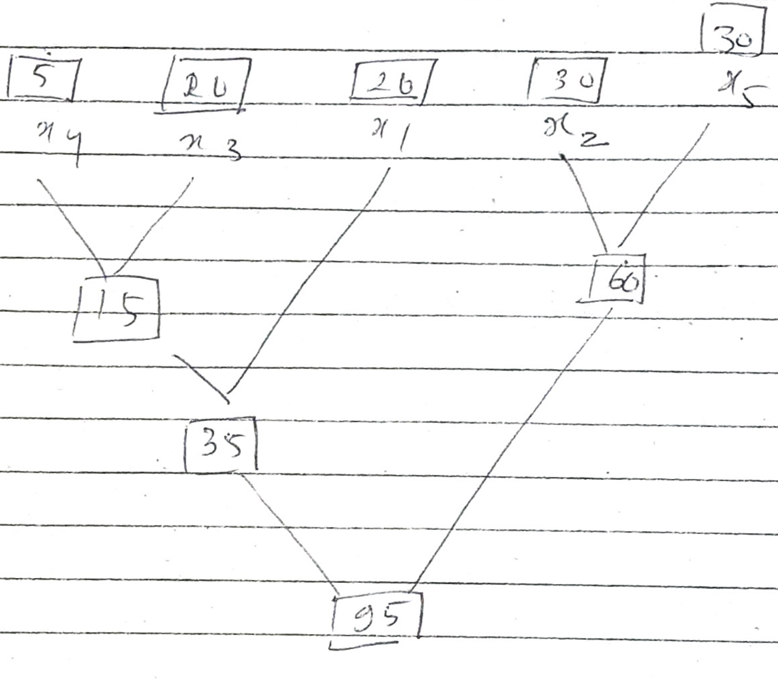
total = 31

always merge pair of small size list to get the best result.

eg:

list	x_1	x_2	x_3	x_4	x_5
size	20	30	10	5	30

(always select smaller. lets sort



total size ⇒ 15 + 35 + 95 + 60
= 205

$= \sum d_i \times x_i$

Huffman Coding

it is used for reducing the size of data or message

message - B C C A B B D D A E C C B B A E D D C C

length of message = 20

it'll be send by using ASCII code and those are 8 bit

A	65	01000001
B	66	01000010
C	67	01000100
D	68	01000101
E	69	0100010U

now total no. of bits will be 8 X 20 = 160 bits

total size of message (without encoding)

#fix size code

Character	count/frequency ↓		code
A	3	3/20	000
B	5	5/20	001
C	6	6/20	010
D	4	4/20	011
E	2	2/10	100

in - 1 bit we can store either 0 or 1 ie two

-- 2 bit $2^n = 4$

→ but we need to store 5 char

∴ 3 bit needed

now change the ASCII code for message such as

B = 001 C = 010 etc

now size of message is 20×3=60 bits
but to decode the encoded message we need the table also. now size of table -

5×8 bits ↑ char or symbol ↑ ASCII

5×3 bit ↑ code

40 + 15 = 55

now total size ⇒ msg + table
= 60+55
= 115 bits

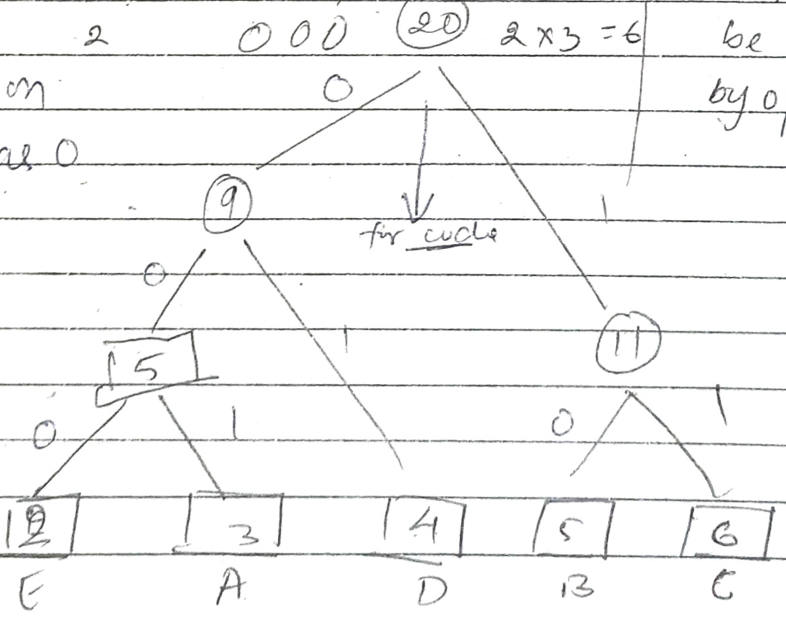
variable size code / Huffman coding.

↓ we don't have to take fix size of code some characters and alpha may be appearing less and some may be appearing more by optimal merge pattern
↓
greedy approach

Char	count	code	count X bit size size
A	3	0 01	3X3=9
B	5	10	5×2=10
c	6	11	6×2=12
D	4	01	4×2=8
E	2	000	2×3=6

every edge on L.H.S mark as 0 and on RHS mark as 1

now for code go from top to ottom



now size of message. = 9 + 10 + 12 + 8 + 6
= 45 bits.

size of table or tree = 8 × 5
bit char
= 40 bits

code
= 12 bits
total

= 40 + 12
= 52

total size = 97 bits (52+45)

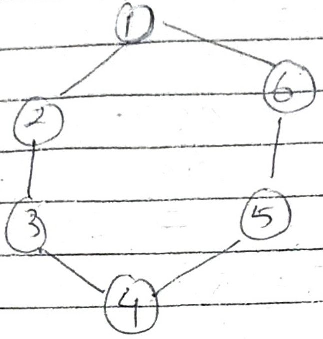
bet.

Prims & kruskals algorithm

minimum spanning tree

spanning tree is subset of graph

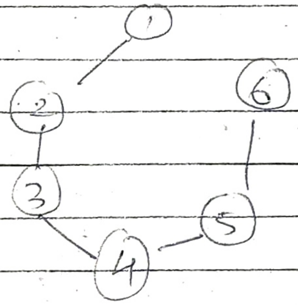
- need to take all vert
- subset of edge



$G = (V, E)$

$|V| = n = 6$ vertices

$n-1 = 5$ edges
↓
for spanning tree



$S = (V, E')$ $S \subseteq G$

where $V' = V$

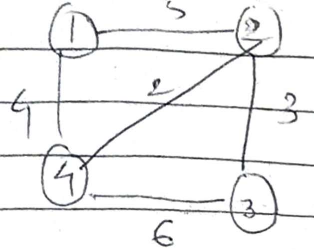
$E' = |V| - 1$

- for a given graph how many spanning trees can be generated

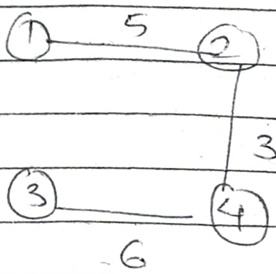
$|E| = 6$ in previous eg.

${}^6C_5$ ways = 6 ways.

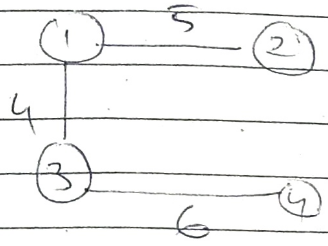
(*) ${}^{|E|}C_{|V|-1}$ - no. of cycles for most spanning.



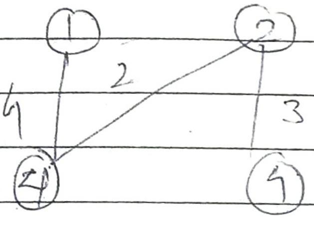
weighted graph



cost = 14



cost = 15



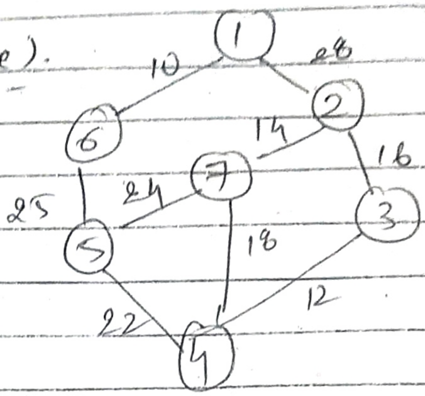
cost = 9

now to find minimum spanning tree without finding all the cost we got two algorithm
1) prims algorithm
2) Kruskals algorithm

for non connected graph we can't find spanning tree at all.
for prim's

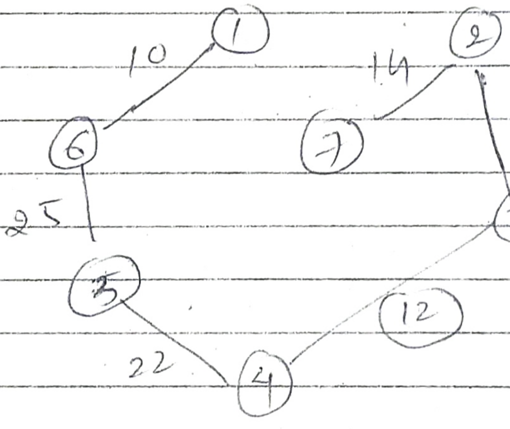
prim's algorithm

1) initially select the smallest weight ~~or~~ (minimum cost edge).



2) and then select the smallest connected

- total number of vertices are 7 so we can select 6 edge.



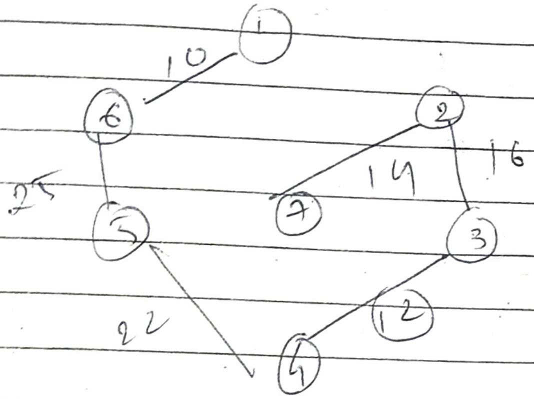
total cost = 99

#kruskal's method

with previous eg.

1) alway select the minimum weight edge,

2) if it is making a cycle do not include that edge

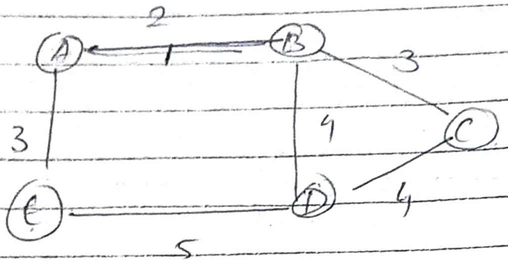


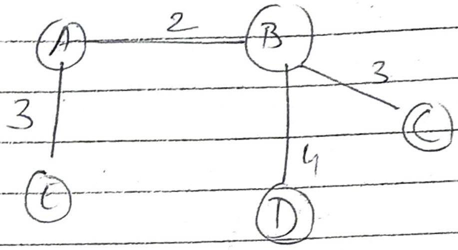
cost = 99.

total time $O(|V||E|)$
$\Rightarrow O(n \cdot e)$
$\Rightarrow O(n^2)$

but it can be improve if we use min heap. then the Time complexity $O=(n \log n)$

eg.





Cost = 12