0287

Binary Search

```
Alg  B_Search (a, i, l, x)
// a is array, i is start pos, l is last pos, x is search ele
{
  if (i == l) {
      if (x == a[i])
            return i;
      else
            return 0;
  }
  else {
      mid = (i+l)/2;
      if (x == a[mid])
            return mid;
      else if (a[mid] > x)
            return B_Search (a, i, mid-1, x);
      else
            return B_Search (a, mid+1, l, x);
  }
}
```

$$T(n) = \begin{cases} c_1 & n = 1 \\ T(n/2) + c_2 & \end{cases}$$

$= \boxed{T(n/4) + c_3} + c_2$

$= T(n/8) + c_4 + c_3 + c_2$

$\vdots$

$= T(1) + c_k + c_{k-1} + c_{k-2} + \cdots c_3 + c_2$

$n = 2^k$

$\log n = \log 2^k \qquad \log n = k \log_2 2$

$k = \log n$

$\log n \rightarrow O(\log n)$

- $O(1)$ - best case
- $O(n)$ - worst case

MERGE SORT

(Ex) [179, 254, 285, 310, 351, 423, 450, 520, 652, 861]

Time complexity

$$T(n) = \begin{cases} c_1 & n = 1,\ c_1 \text{ is a constant} \\ 2T(n/2) + c_2 n & n > 1,\ c_2 \text{ is a constant} \end{cases}$$

Assume $n = 2^k$, then $T(n) = 2T(n/2) + c_2 n$

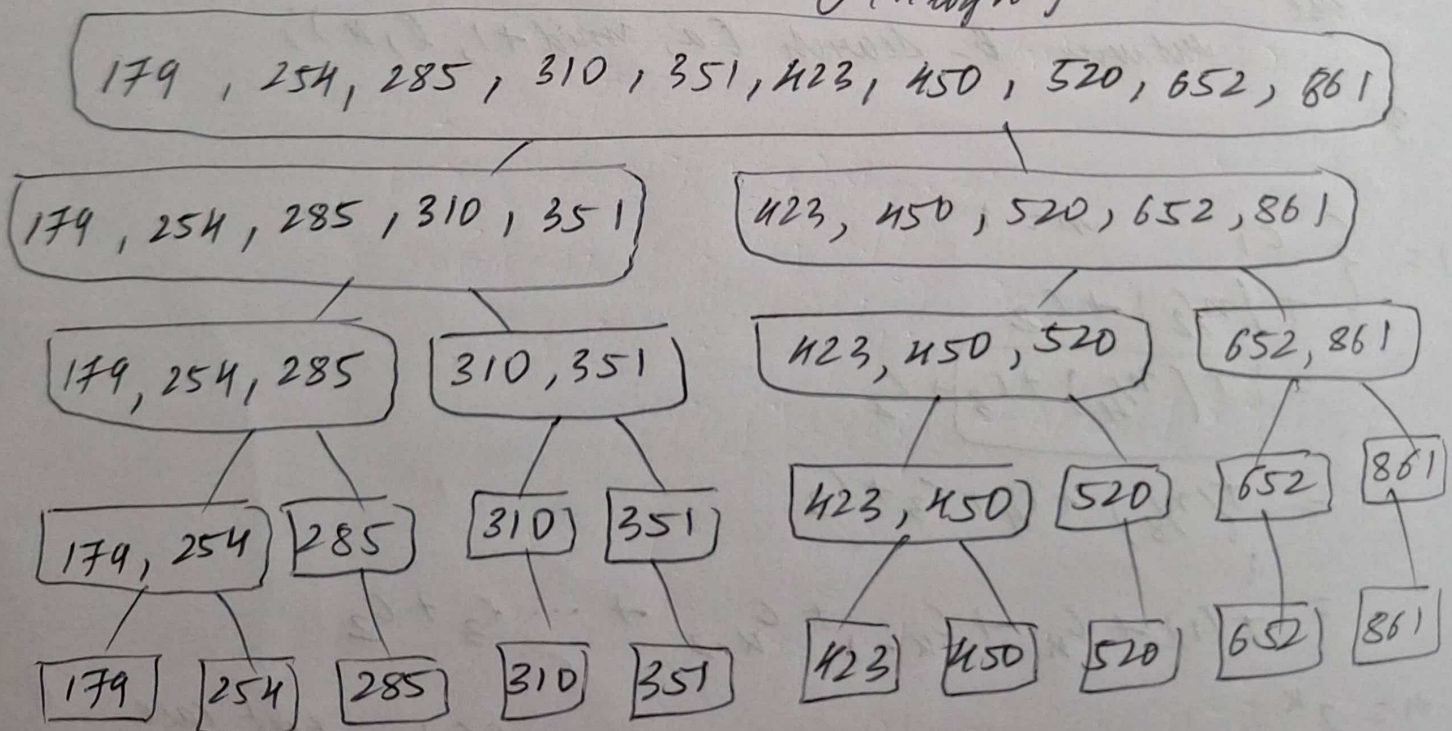
$$= 2\left(2T\left(\frac{n}{4}\right) + c_2 n_2\right)$$

$$= 2^2 T\left(\frac{n}{4}\right) + c_2 n + cn$$

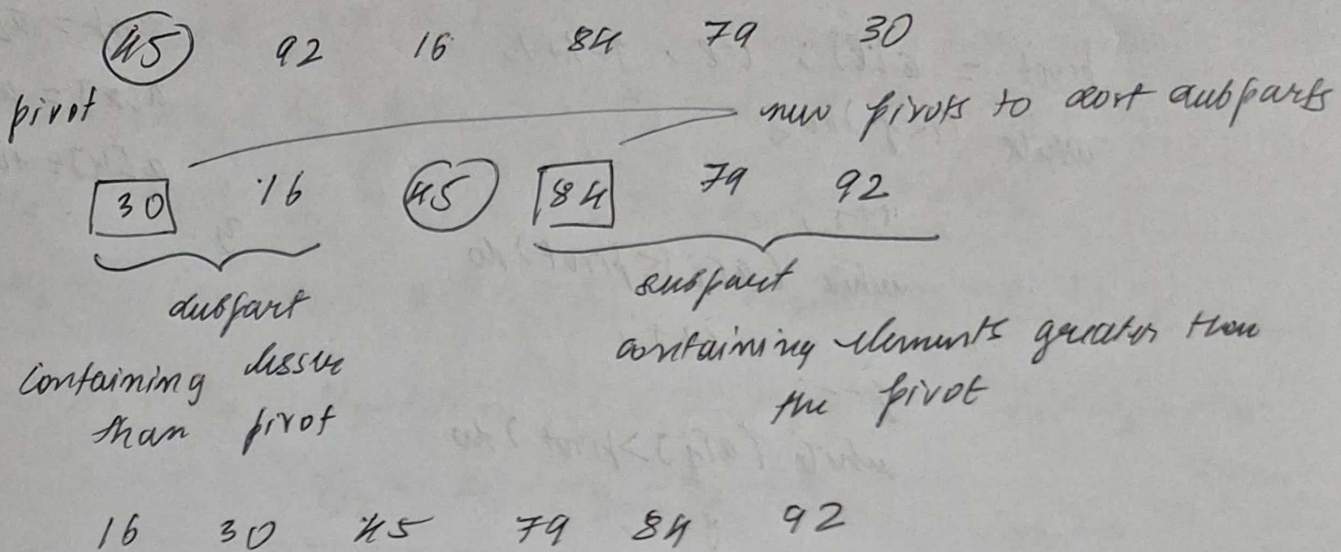
$$= 2^2 T\left(\frac{n}{4}\right) + 2c_2 n$$

$$= 2^k + T\left(\frac{n}{2^k}\right) + kc_2 n$$

$$= O(n \log n)$$



QUICK SORT




```
QuickSort (low, high) {
    if (low < high) {
        i = Partition (low, high)
```

Time complexity - Worst case : $O(n^2)$ Pivot is smallest or largest.

$$T(n) = \begin{cases} +(1) & \text{if } n = 1 \\ T(n-1) + c_n & \text{if } n > 1 \end{cases}$$

Best case - $O(n \log n)$

Avg case

```
partition (l,h) {
    prot = a[l]; i=l; j=h+1;
    while (i<j) do {
        i++;
        while (a[i]<prot) do
            i++;
        j--;
        while (a[j]>prot) do
            j--;
        if (i<j) then interchange (i,j)
    }
    interchange (l,j) return
}
```

```
interchange (x,y) {
    temp = a[x];
    a[x] = a[y];
    a[y] = temp;
}
```

Strassen's Matrix Multiplication

basic matrix multiplication — A & B $n \times n$ matrices. C is product $n \times n$

$T(n) = O(n^3)$

```
void matrix_mult() {
    for (i=1; i<=N; i++) {
        for (j=1; j<=N; j++) {
            for (k=1; k<=N; k++)
                c[i,j] = c[i,j] + A[i,k] * B[k,j];
        }
    }
}
```

$$T(n) = \begin{cases} c_1 & \text{if } n<=2 \\ 8T(n/2) + 4n^2 & \text{if } n>2 \end{cases}$$

Formulas for Strassen's Algorithm

$M_1 = (A_{11} + A_{22}) * (B_{11} + B_{22})$

$M_2 = (A_{21} + A_{22}) * B_{11}$ $\quad M_3 = A_{11} * (B_{12} - B_{22})$

$M_4 = A_{22} * (B_{21} - B_{11})$ $\quad M_5 = (A_{11} + A_{12}) * B_{22}$

$M_6 = (A_{21} - A_{11}) * (B_{11} + B_{12})$ $\quad M_7 = (A_{12} - A_{22}) * (B_{21} + B_{22})$

$C_{11} = M_1 + M_4 - M_5 + M_7$ $\quad C_{12} = M_3 + M_5$ $\quad C_{21} = M_2 + M_4$

$C_{22} = M_1 + M_3 - M_2 + M_6$

$$T(n) = \begin{cases} c_1 & n <= 2 \\ 7T(n/2) + c_2 n^2 & n > 2 \end{cases}$$

$T(n) = O(n^{\log_2 7})$

$T(n) = c_1 n^{2.81 \log_2 2}$

$T(n) = c_1 n^{\log_2 7}$

$= O(n^{\log_2 7})$

$\sim O(n^{2.81})$

(2)

$$\begin{bmatrix} 7 & 9 \\ 8 & 4 \end{bmatrix} \quad \begin{bmatrix} 3 & 5 \\ 6 & 2 \end{bmatrix}$$

22-07-29

Finding min & max

```
MinMax (i, j, min, max) {
    if (i == j)
        min = i
    else if (i == j-1) {
        if (a[i] > a[j]) {
            max = i
            min = j
        }
        else {
            max = j
            min = i
        }
    }
    else {
        mid = (i+j)/2
        if Except.
        minmax (i, mid, min, max)
        minmax (mid+1, j, min1, max1)
        if (min > min1)
            min = min1
        if (max1 > max)
            max = max1
    }
}
```

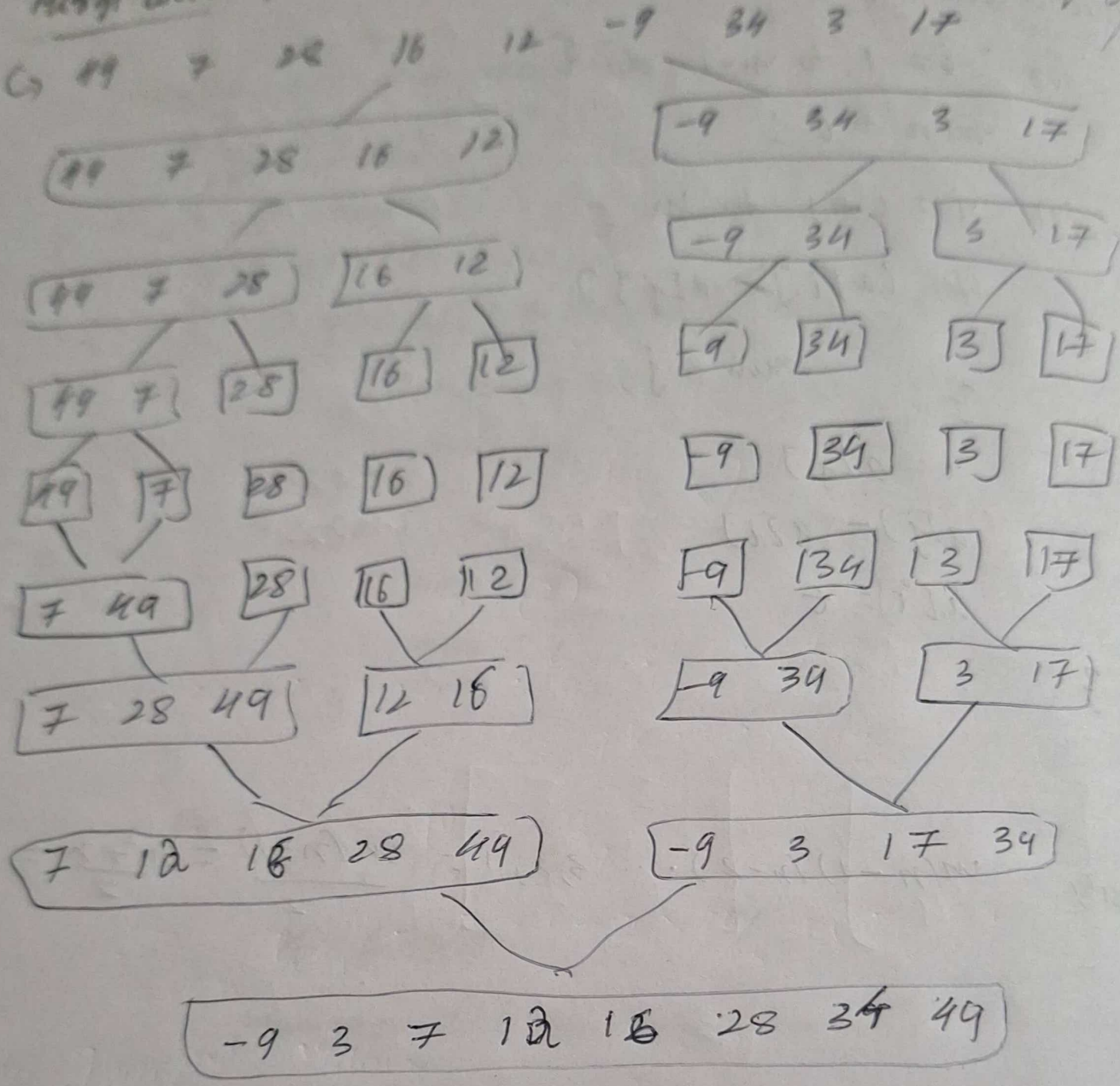
Selection Sort

```
selectionSort(A) {
    for i = 1 to n-1 do {
        min = i;
        for j = i+1 to n {
            if (a[i] > a[j])
                min = j;
        }
        t = a[j]
        a[j] = a[i]
        a[i] = t
    }
}
```

complexity $n(n-1)(n-2)\cdots 3,2,1 = \frac{n(n+1)}{2} = \frac{n^2+n}{2} = O(n^2)$

28-09-2024

⑥ Merge sort & Quick sort



Quick Sort

49 7 28 16 12 -9 34 3 17

pivot

17 3 34 -9 12 16 28 7 49

pivot subpart (lesser than pivot)

7 16 12 -9 3 17 34 28 49

sub sub

3 -9 7 16 12 17 28 34 49

✓ -9 3 7 12 16 17 28 34 49

Greedy

- Knapsack Problem └ optimal storage tapes └ job sequence with dead lines └ single source shortest path

Knapsack problem consists of n objects and one bag, each object i has a weight w_i and a profit p_i and the bag has capacity m. If fraction x_i ($0 \le x_i \le 1$) of object i is placed into the knapsack then a profit of $p_i x_i$ is earned. The objective is to obtain a filling of bag that maximizes the total profit earned. $\sum_{1 \le i \le n} p_i x_i$

subject to weight $\sum_{1 \le i \le n} w_i x_i \le m$.

Here profits and weights are positive nos.

Consider the following instance of knapsack prob. $n=3$, $w=20$. Profits are P_1, P_2, P_3

Items	A	B	C
Profits (P)	25	24	15
Weights (w)	18	15	10

x_1 is having unit profit $\frac{25}{18}$ with w_1 as 18

$w - 18 = 20 - 18 = 2$

$x_2 = \frac{2}{15}$ $\quad x_3 = 0$

$\sum P_i x_i = 25 * 1 + 24 * \frac{2}{15} + x_3 * 0 =$

$25 + \frac{48}{15} + 0 = \frac{25 + 3x}{0} = 28.2$

$x_3 = 0 \quad x_2 = \frac{20}{18} \quad x_1 = 0$

$\sum_{1 \le i \le n} P_i x_i = 15 \times 1 + 2 \times \frac{20}{15} + 0 = 31$

Profit / weight ratio

(3) $x_1 = \frac{25}{18} = 1.38$ (1) $x_2 = \frac{24}{15} = 1.6$

(2) $x_3 = \frac{15}{10} = 1.5$

$P_1 x_1 + P_2 x_2 + P_3 x_3$

$\frac{5}{10} \times 15 = 0 + 24 \times 1 + 0.75 = 31.5$

$(0, 1, \frac{1}{2})$

Algorithm of Knapsack problem

Algorithm Greedy(p, w, n, m, x)

// p[1:n] w[1:n]

// p[i]/w[i] ≥ p[i+1]/w[i+1]

// m is the knapsack size and x[1:n] is solution vector.
(bag capacity)

// n x[1:n]

{

for i = 1 to n do

x[i] = 0.0

u = m;

if (w[i] > u) then break.

x[i] = 1.0;

u = u - w[i];

}

if (i ≤ n) then x[i] = u / w[i+1]

}

JOB SEQUENCE WITH DEAD LINES

Jobs	1	2	3	4
Profits	100	10	15	27
deadlines	2	1	2	1

(b)

Task	DL	Profit
T1	7	15
T2	2	20
T3	5	30
T4	3	18
T5	4	14
T6	5	10
T7	1	23
T8	7	16
T9	3	25

Arrange the elements decreasing

Task	DL	Profit	
T3	5	30	✓
T9	3	25	✓
T7	2	23	✓
T2	2	20	✓
T4	3	18	X
T8	7	16	✓
T1	7	15	✓
T5	4	10	✓
T6	5	10	X

1	2	3	4	5	6	7
20	23	25	18	30	15	16

$30 + 25 + 23 + 20 + 18 + 16 + 15$

$= 147$

$T_1 + T_2 + T_3 + T_5 + T_7 + T_8 + T_9$

```
Alg  Greedy job (d, j, n) {
     J = {1}
     for i = 2 to n {
          if (all jobs in J ∪ {i} can be completed by
                                        their deadlines) then
          {     J = J ∪ {i};
          return J;
     }
```

```
Alg  Job Seq (d, j, n)
// d[i] ≥ 1        1 ≤ i ≤ n  are the deadlines n ≥ 1
// the jobs are ordered such that p[1] ≥ p[2] ≥ ... p[n]
// J[i] is the i^th job in optimal solution 1 ≤ i ≤ k
// at termination  d[J[i]] ≤ d[J[i+1]]  1 ≤ i ≤ k
  {
     d[0] = J[0] = 0;
     J[1] = 1;
     k = 1;
     for i = 2 to n do {
          // consider jobs in non-increasing order of
          // p[i]  based on profits
          // find position for i & check feasibility of
          // insertion.   r = k;
          while ((d[J[r]] > d[i] and d[J[r]] ≠ r) {
               do r = r - 1;
          if ( d[J[r]] < d[i]) and (d[i] > r)) then {
```

```
for q = k to r+1 step -1 do
    s[q+1] = s[q];
    s[r+1] = i;
    k = k+1;
}
return k;
```

Assignment

What is the solution generated by the job sequencing with deadlines $n = 7$ $(p_1, p_2, \ldots p_7) = (3, 5, 20, 18, 1, 6, 30)$

$(d_1, d_2, \ldots d_7) = (1, 3, 4, 3, 2, 1, 2)$

Task	DL	P
T_1	1	3
T_2	3	5
T_3	4	20
T_4	3	18
T_5	2	1
T_6	1	6
T_7	2	30

Task	D_L	P	
T_7	2	30	✓
T_3	4	20	✓
T_4	3	18	✓
T_6	1	6	✓
T_2	3	5	✗
T_1	1	3	✗
T_5	2	1	✗

1	2	3	4
6	30	18	20

$T_1 + T_3 + T_4 + T_6 = 47$

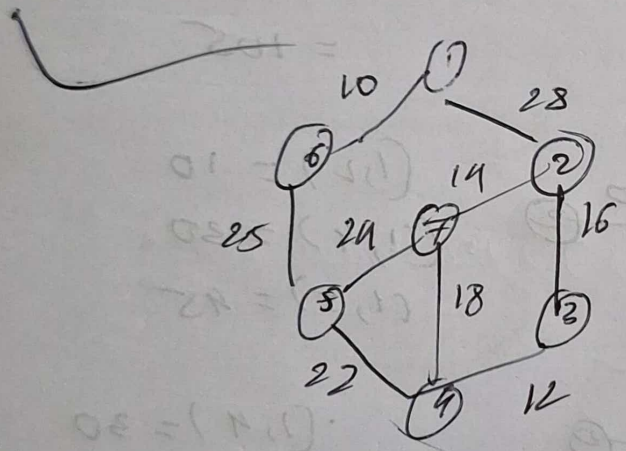
Minimum cost Spanning Tree

A spanning tree is a tree which includes all vertices in a graph. If the graph consists of n vertices. then possibilities are n^{n-2}

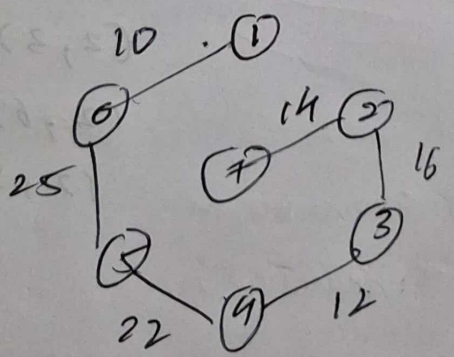
Cost of a spanning tree is a sum of costs on its edges.

Minimum cost spanning tree for a weighted connected & undirected graph. is a spanning tree with weights less than or equal to the weight of other spanning trees.

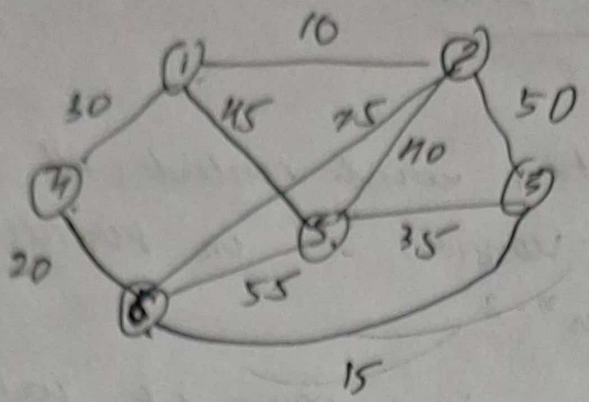
- Prim's
- Kruskal's

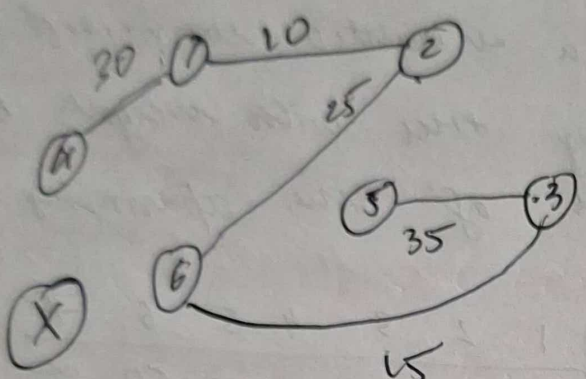


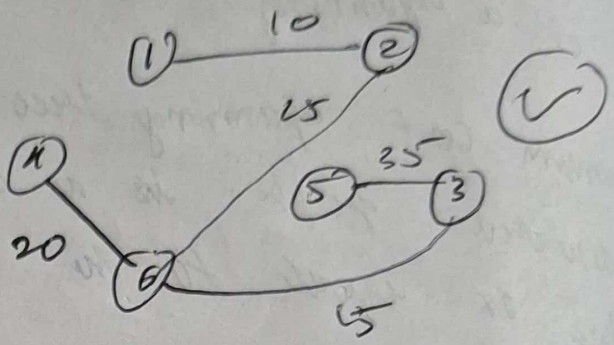
·	1	2	3	4	5	6	7
1	0	28	∞	∞	∞	10	∞
2	28	0	16	∞	∞	∞	14
3	∞	16	0	12	∞	∞	∞
4	∞	∞	12	0	22	∞	18
5	∞	∞	∞	22	0	25	24
6	10	∞	∞	∞	25	0	∞
7	∞	14	∞	18	24	∞	0



10+25+22+12+16+14 = (99)





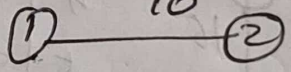


$10+30+25+15+35 = 115$

$20+15+35+15+10$

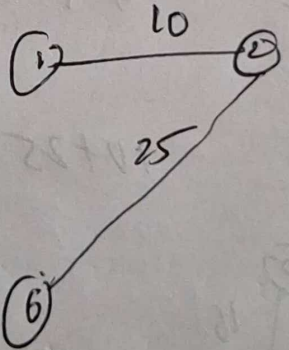
$= 105$

Step (1)



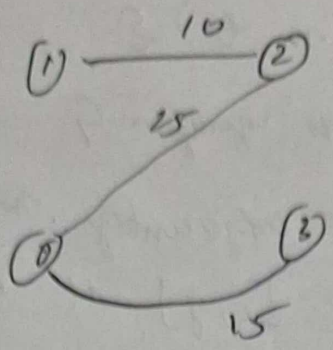
(1,2) = 10
(1,4) = 30
(1,5) = 45

Step (2)



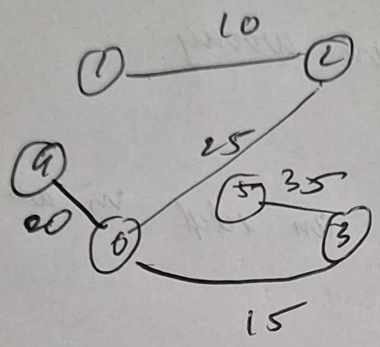
(1,4) = 30
(1,5) = 45
(2,3) = 50
(2,6) = 25
(2,5) = 40

step ③



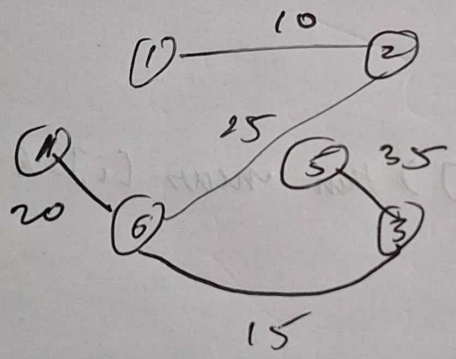
(1, 4) = 30
(1, 3) = 45
(2, 3) = 50
(2, 5) = 40
(6, 4) = 20
(6, 3) = 15
(6, 5) = 35

step ④



(1, 4) = 30
(1, 5) = 45
(2, 3) = 50
(2, 5) = 40
(6, 4) = 20 ← (6, 5) = 55
(3, 5) = 35
(3, 2) = 50

step ⑤



(1, 4) = 30 ×
(1, 5) = 45
(2, 3) = 50
(2, 5) = 40
(6, 5) = 55 ×
(3, 5) = 35
(3, 2) = 50

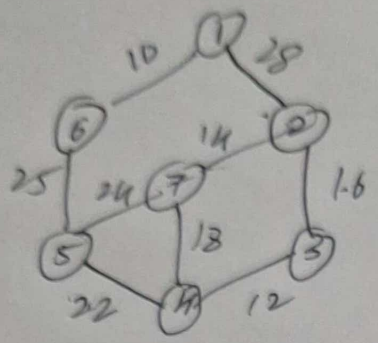
```
Algorithm Prim (E, cost, n, t) {
// E is the set of edges in the graph G
// cost [1:n, 1:n] is cost adjacency matrix of an undirected
// graph such that cost of [i,j] is either the positive
// real number or Infinity if no edge is available
// A minimum spanning tree is computed and stored as
// a set of edges in an array
// t [1: n-1, 1:2]
// (t [i,1], t [i,2]) is an edge in an min cost
   spanning tree.

   mincost = cost(
   t [1,1] = k [1,2] = t;
For i=1 to n do {
   if (cost [i,j] < cost [i,k]) then near [i] = k;
   else near [i] = k;
   near [k] = near [i] = 0;
}
for i=2 to n-1 do {
   Let j be an index such that near [j] ≠ 0 and
   cost [j, near [j]] is minimum
   t [i, 1] = t [i, 2] = near [j];
   mincost = mincost + cost [j, near [j]];
   near [j] = 0;
```

for k = 1 to n do

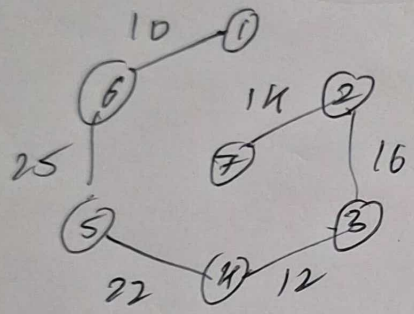
if (near[k] = 0 and cost(k, near[k]) > cost[k, j])

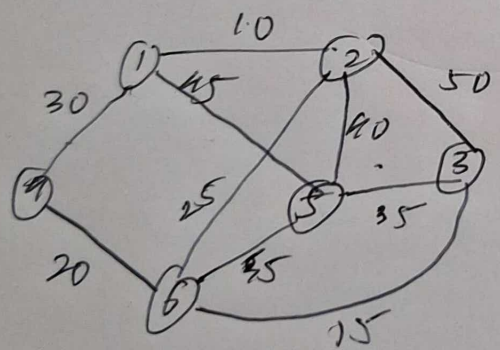
KRUSKAL'S



(1,6) - 10
(4,3) - 12
(2,7) - 14
(2,3) - 16
(4,7) - 18
(5,4) - 22
(5,7) - 24
(6,5) - 25

S-1 (1,6) - 10
S-2 (4,3) - 12
S-3 (2,7) - 14
S-4 (2,3) - 16
S-5 (4,7) - 18 ✗
S-7 (5,7) - 24
S-8 (6,5) = 25
S-9 (1,2) = 28 ✗
TC = 99





(1,2) = 10 ✓
(6,3) - 15 ✓
(4,6) - 20 ✓
(6,2) - 25 ✓
(4,1) - 30 ✗
(5,3) - 35 ✓
(5,2) - 40 ✗
(1,5) - 45 ✓
(2,3) - 50 ✗
(6,5) - 55 ✗

TC = 150