* Binary Search Iterative Method

```
int BS (A, n, key) {
    l = 1 , h = n
    while (l ≤ h) {
     mid = (l+h)/2 ;
      if (key = = A [mid]  return mid;
      if (key < A [mid]  h = mid - 1;
      else  l = mid + 1;
    }
}
```

min → O(1)

max → O(logn)

BS (Recursive)

```
Algorithm BS(A, l, h, key) {                    → T(n)
    if (l == h) {
        if (A[l] == key) return l;
        else return 0;
    }
    else {
        mid = (l+h)/2;                          — 1
        if (key == A[mid]; return mid;          — 1
        if (key < A[mid] return BS(l, mid-1, key);   } T(n/2)
        else return BS(mid+1, h, key);
    }
}
```

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

$T(n) = T(n/2) + 1$

$T(n/2) = T(n/2^2) + 1$

$T(n/2^2) = T(n/2^3) + 1$

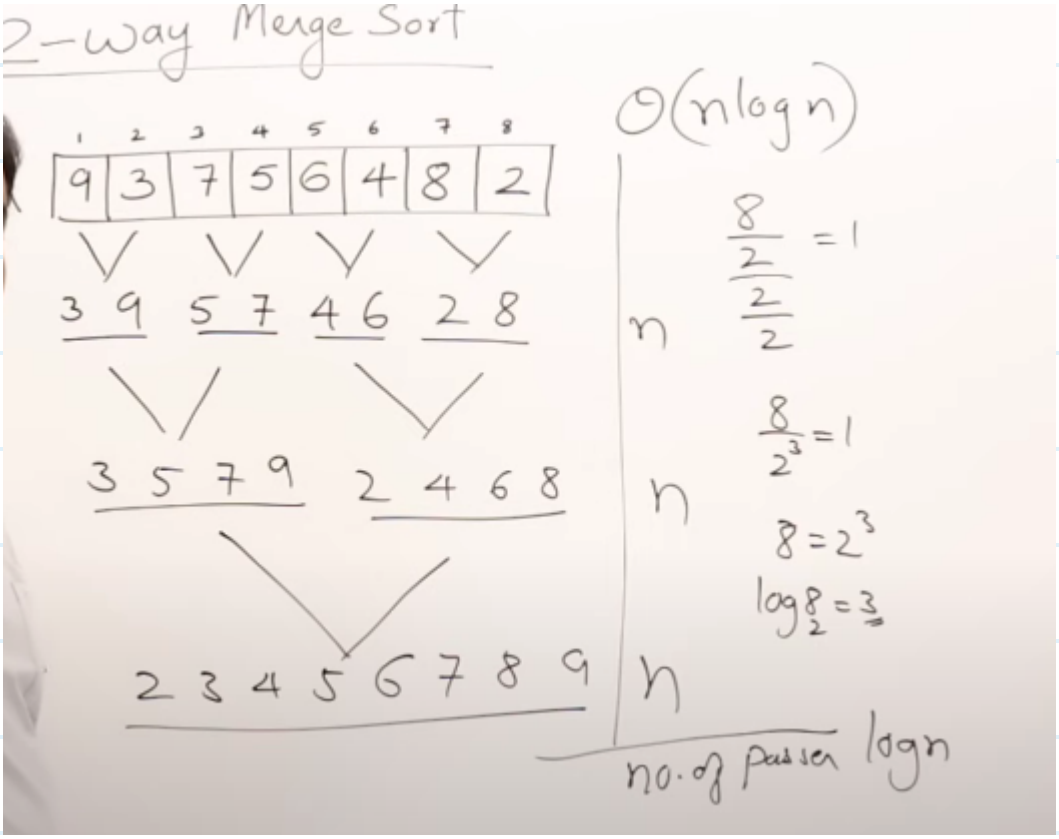
$\frac{n}{2^k} = 1$

$2^k = n \Rightarrow k = \log n \qquad O(\log n)$

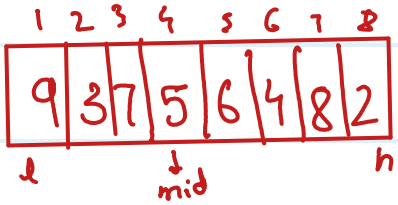
* Merge Sort

```
Algorithm Merge (A, B, m, n) {
    i=1, j=1, k=1
    while (i<=m && j<=n) {
        if (A[i] < B[j])
            C[k++] = A[i++];
        else
            C[k++] = B[j++]
    }
    for (i=1; i<=m; i++) C[k++] = A[i];
    for (j=1; j<=n; j++) C[k++] = B[j];
}
```

O (nlogn)

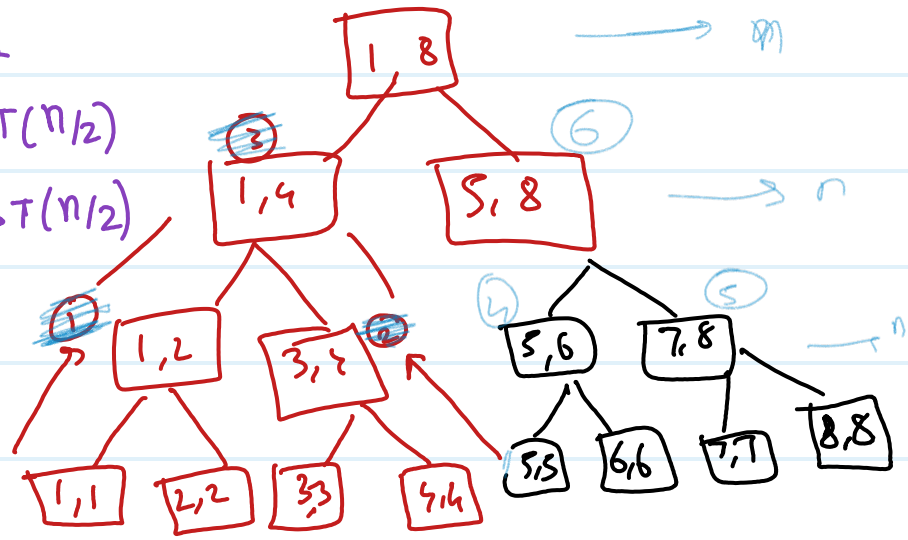


Merge Sort (Recursive Way)




```
Algorithm MergeSort(l, h) {          → T(n)
    if (l < h) {
        mid = (l + h)/2;             → 1
        MergeSort (l, mid);          → T(n/2)
        MergeSort (mid+1, h);        → T(n/2)
        Merge (l, mid, h);           → n
    }
}
```

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



$\Theta(n \log n)$

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

$a = 2, \; b = 2, \; f(n) = n, \; k = 1, \; p = 0$

$\log_2 2 = 1 = k$

Case II: $p > -1 \rightarrow \Theta(n^k \log^{p+1} n)$

$\therefore \; \Theta(n \log n)$

* Quick Sort

```
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]);
    return j;
}
```

```
Quick Sort (l, h) {
    if (l < h) {
      j = partition (l, h);
      Quicksort (l, j);
      Quicksort (j+1, h);
    }
}
```

O (nlogn)
↳ Best case

O (n^2)
↳ Worst case

* Strassen's Matrix Multiplication

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

$O(n^3)$

```
Algorithm MM (A, B, n) {
    if (n ≤ 2) {
        c = 4 formulas
    } else {
        mid = n/2
        MM(A11, B11, n/2) + MM(A12, B21, n/2)
        MM(A11, B12, n/2) + MM(A12, B22, n/2)
        MM(A21, B11, n/2) + MM(A22, B21, n/2)
        MM(A21, B12, n/2) + MM(A22, B22, n/2)
    }
}
```

$$T(n) = \begin{cases} 1 & n \le 2 \\ 8\,T(n/2) + n^2 & n > 2 \end{cases}$$

$a = 8,\ b = 2\quad f(n) = n^2$

$k = 2\quad p = 0$

$\log_2 8 = 3 > k = 2$

$\therefore \Theta(n^3)$

Strassen's MM

$P = (A_{11} + A_{22})(B_{11} + B_{22})$

$Q = (A_{21} + A_{22}) B_{11}$

$R = A_{11} (B_{12} - B_{22})$

$S = A_{22} (B_{21} - B_{11})$

$T = (A_{11} + A_{12}) B_{22}$

$U = (A_{21} - A_{11})(B_{11} + B_{12})$

$V = (A_{12} - A_{22})(B_{21} + B_{22})$

$C_{11} = P + S - T + V$

$C_{12} = R + T$

$C_{21} = Q + S$

$C_{22} = P + R - Q + U$

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

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

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

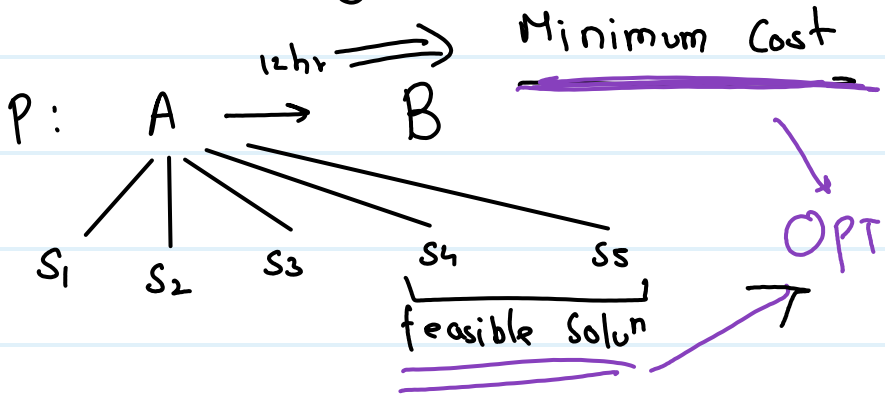
$C_{22} = A_{21} \times B_{12} + A_{22} \times B_{21}$

$$T(n) \begin{cases} 1 & n \leq 2 \\ 7T(n/2) + n^2 & n > 2 \end{cases}$$

$\log_2 7 = 2.81$, $K = 2$

$O(n^{2.81})$

* Greedy Method



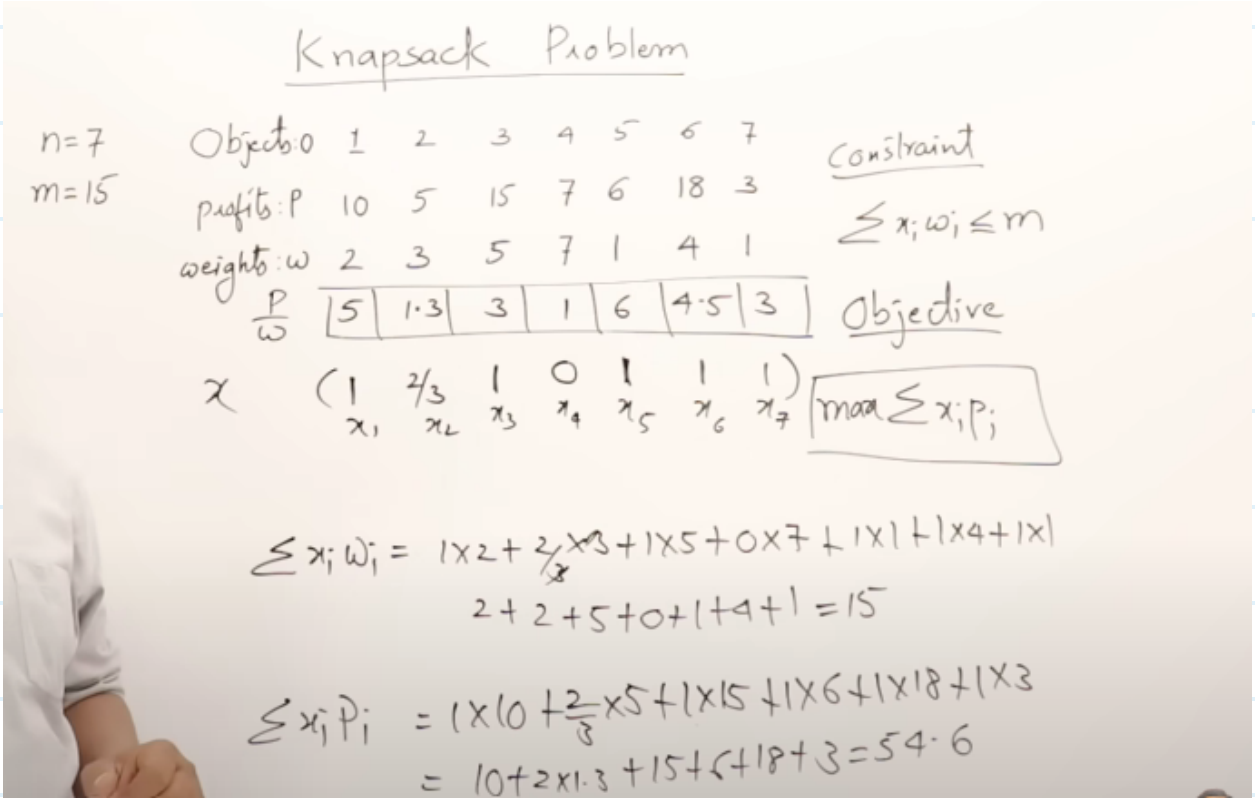
Minimum Cost

OPTIMAL SOLUTION

↳ Optimization Problems

```
Algorithm Greedy (a, n)
    for int i = 1 to n do {
        x = select (a);
    if feasible (x) then
        soln = soln + x;
    }
}
```

* Knapsack Problem



Algorithm: Fractional Knapsack (S,W)

S = {i1, i2, i3 ... in} set of items
each item is assigned a weight wi, & value vi →
The max wt of the knapsack is W

```
FOR item i=1 →n
    xi ← 0                      // selection
    pi ← vi /wi                 //profit /wt ratio
END FOR

cw ← 0    i ← 1                 //cw = current wt
WHILE cw < W & i<=n
  remove highest profit pi from S

    if (cw+wi) <= W
        xi ← 1                  cw += wi
    else
        xi ← (W - cw) /wi
        cw = W
    endif
  i → i+1
END WHILE
return (x)
```

Sorting : O(n log n)
Iterate : O(n)
Total complexity : O(n log n)
SC: O(1)

* Job sequencing with Deadlines

Job Scheduling

```
// Initialize
profit = 0
list [] = 0

FOR i = 1 → n
    k = d[i]
        WHILE k > 0
            IF list[k] = 0
                profit ← profit + p[i]
                list[k] = job[i]
                break
            ENDIF
        k--
    END
END FOR
```

4 3 2 1

~~2~~ 3 2 1

WC Complexity : $O(n^2)$

Q. $n = 5$ $(p_1, p_2 .. p_5) = (20, 15, 10, 5, 1)$
$(d_1, d_2 .. d_5) = (2, 2, 1, 3, 3)$

Jobs	1	2	3	4	5
p	20	15	10	5	1
d	2	2	1	3	3

max days = no. of slots

J1 takes 2 days ⇒ 2nd slot

J_2 takes 2 days ⇒ " occ.

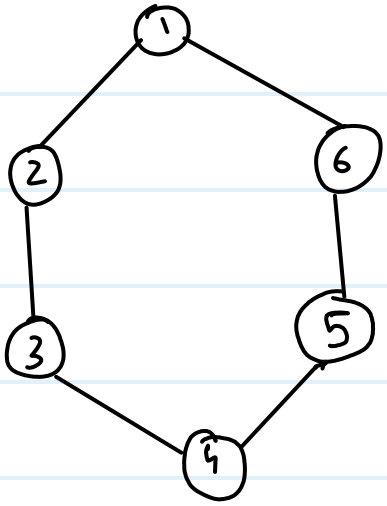
∴ slot 1

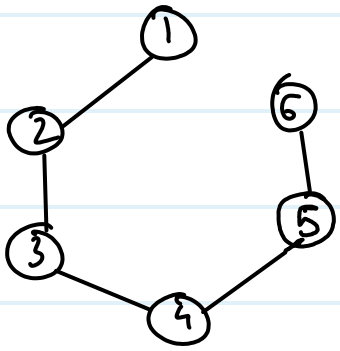
J_3 takes 1 day ⇒ 1st slot occ

∴ discard

J_4 takes 3 days ⇒ 3rd slot

Minimum Cost Spanning Tree





$|V| = n = 6$

$|V| - 1 = 5$

$G = (V, E)$

$V = \{1, 2, 3, 4, 5, 6\}$

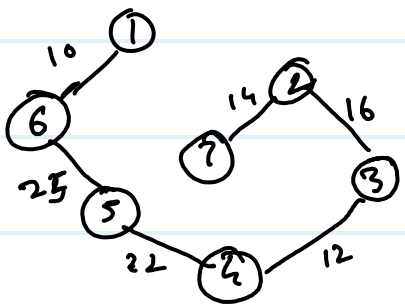
$E = \{(1,2)(2,3)(3,4)\ldots\}$

$|E| = 6$

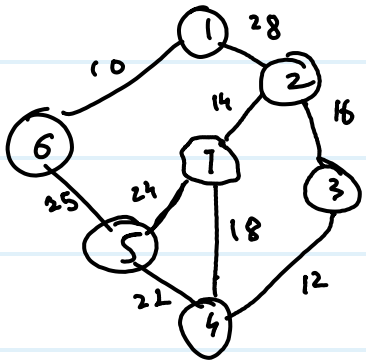
${}^{6}C_{5} = 6$

$|E|\ C_{|V|-1}$ – no. of cycles

- Prim's Algorithm



Min cost = 99



Prim's Algorithm

n = no. of nodes in a graph
nEdges = no of nodes in MST
min = min wt of edge
sum = total min cost

selected [] = 0

Algorithm :
// Initialize
selected [1] = 1
nEdges = 1
sum = 0

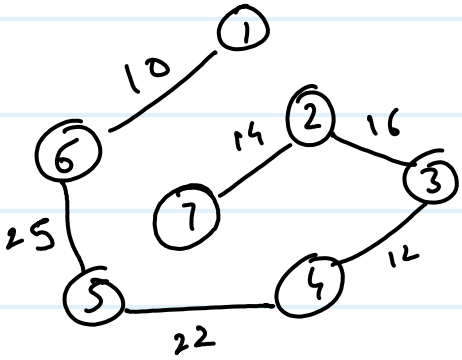
WHILE nEdges < n
min = ∞

// Find min edge connecting node to unselected node
FOR i = 1 → n
IF (selected [i] == 1)

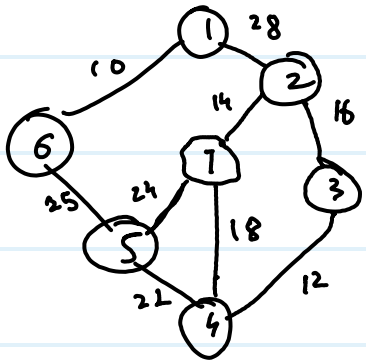
FOR j = 1 → n
IF (selected [j] == 0 & adj [i] [j] exists)
min = adj [i] [j]
row = i
col = j
END IF
END FOR
ENDIF
Print row, col, adj [row] [col];
END FOR
selected [col] = 1
nEdges ++
sum = sum + adj [i] [j];
END WHILE
Print sum;

* Select node → find its min edge & keep repeating till n-1 edges.
↳ O(n³)

- Kruskal's Algorithm



Min cost = 99



$|E|$

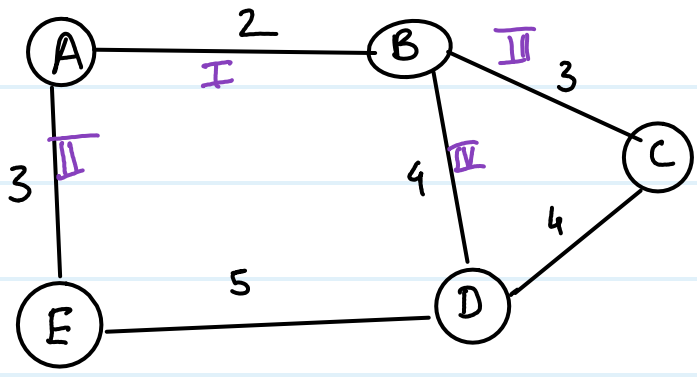
$|V|-1$

$\theta(|V||E|)$

$\theta = (n.e) = \theta(n^2)$

Using Min Heap $\theta(n \log n)$

Kruskal's Algorithm may work for non connected graphs



∴ Min cost

$= 2+3+3+4 = \underline{\underline{12}}$

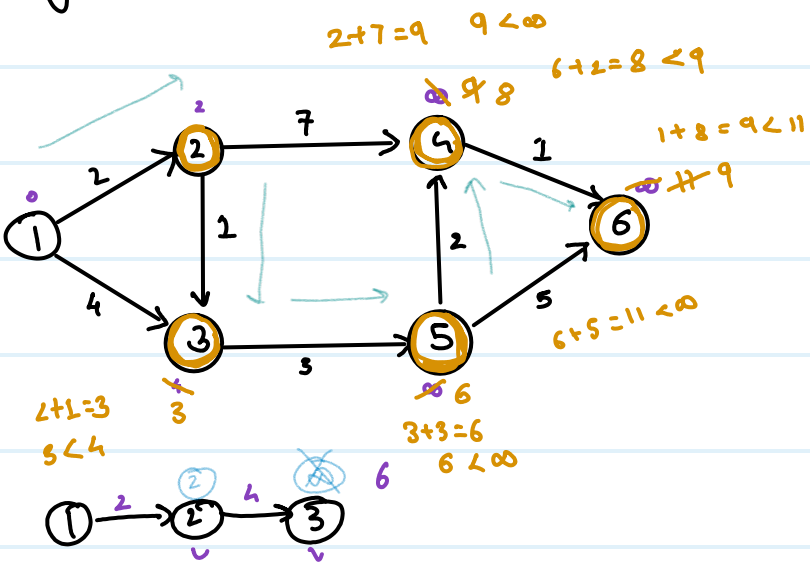
Kruskal's Algorithm

```
tree =
while (tree has < n-1 edges && E is not empty)
{
    select (u,v) edge with min cost
    remove (u,v) from E

    if (u,v) forms cycle
        discard (u,v)
    else
        add (u,v) to tree
    End if
}
```

TC: $O(E \log V)$

- # Dijkstra Algorithm



Relaxation

if $(d[u] + c(u,v) < d[v])$

$d[v] = d[u] + c(u,v)$

v	d[v]
2	2
3	3
4	8
5	6
6	9

$n = |V|$ $|V|$

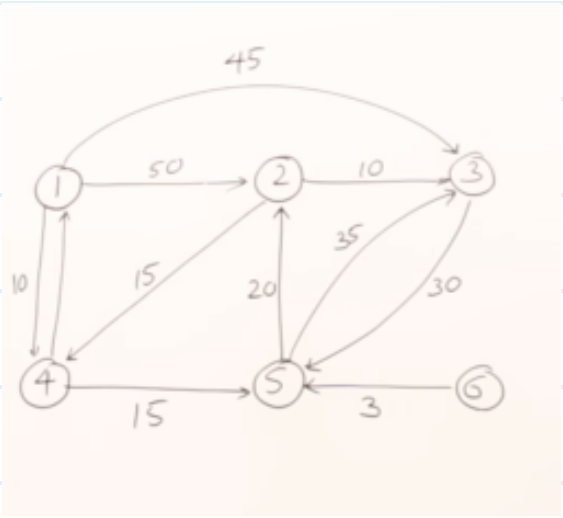
$n \times n = n^2$

worst case

$\Theta(n^2)$

Selected Vertex	2	3	4	5	6
4	50	45	10	∞	∞
5	50	45	10	25	∞
2	45	45	10	25	∞
3	45	45	10	25	∞
6	45	45	10	25	∞

45+10 = 55 > 45
↳ no change



Algo:

```
Dijkstra (v0, W, dist, n)
  // v0 → s. node   W → adj matrix   dist → array to store short. path
     n → no. of nodes
  {
    FOR i = 1 → n        //initialize all visited vertices → 0
    {                          distances from v0
      S[i] = 0
      dist[i] = W[v0, i]
    }                                  → O(n)

    S[v0] = 1
    dist[v0] = 0

    FOR (j=2 → n)                → O(n)
      choose vertex u from vertices not in S
                              such that dist[u] = min

    S[u] = 1

    FOR[ each v adj to u with S[v] = 0) { → O(n)
      if (dist[v] > dist[u] + W[u,v]) {
          dist[v] = dist[u] + W[u,v],
      }
    }
```

TC: $O(n^2)$

SC: $O(n^2)$

Algo MM (A, B, n) {

if (n <= 2) return {

$C_{11} = a_{11} \times b_{11} + a_{12} \times b_{21}$

$C_{12} = a_1$

↳ $T(n) = 8T(n/2) + O(n^2)$

Strassen M

$P = (A_{11} + A_{22})(B_{11} + B_{22})$

$Q = B_{11}(A_{21} + A_{22})$

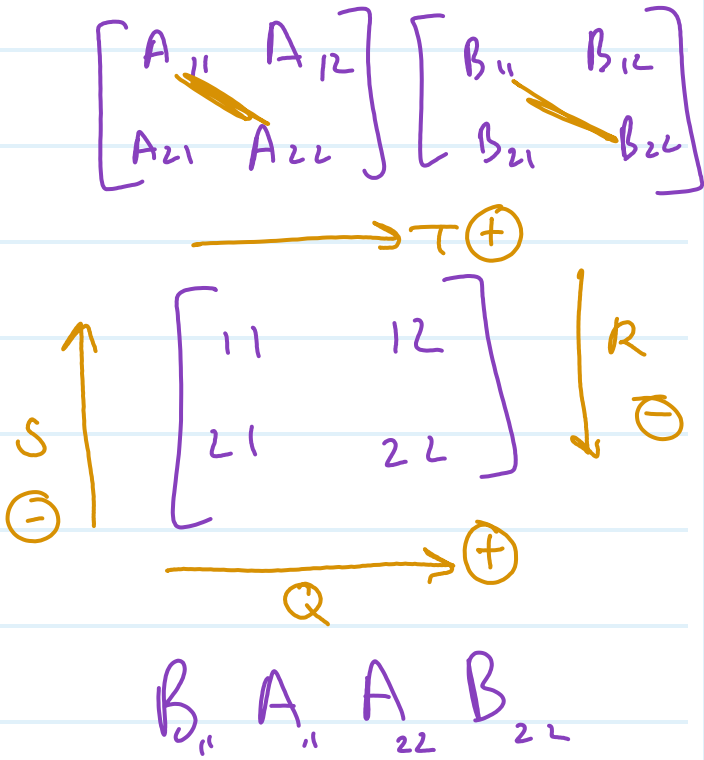
$R = A_{11}(B_{12} - B_{22})$

$S = A_{22}(B_{21} - B_{11})$

$T = B_{22}(A_{11} + A_{12})$

$U = (A_{21} - A_{11})(B_{11} + B_{12})$

$V = (B_{21} + B_{22})(A_{12} - A_{22})$



B_{11} A_{11} A_{22} B_{22}

$C_{ij} \Rightarrow C_{11} = P + S - T + V$

$C_{12} = R + T$

$C_{21} = Q + S$

$C_{12} = P + R - Q + U$

$\Rightarrow$ $T(n) = 7T(n/2) + O(n^2)$

↳ $O(n^{2.81})$