Analysis Insertion sort :

1) for loop – i

2) while loop – j

$$\sum_{i=0}^{n-1} \sum_{j=0}^{i-1} 1 = \sum_{0}^{n-1} i-\cancel{1}-0+\cancel{1} = \sum_{0}^{n-1} i = 0+1+2 \cdots n-2+n-1 = \frac{(n-1)(n-1+1)}{2}$$

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

ignore 1/2.

$\therefore n(n-1) = n^2 - n$

ignore n.

$\Rightarrow C(n) = \Theta(n^2)$.

Algorithm dfs (a, n, u, s, t)

// traverse the graph from source node in DFS.

// Input a : adjacency matrix (u, v)

n = number of nodes

u = source node (row)

// v = adjacent node (column)

s = nodes that are visited & not visited.

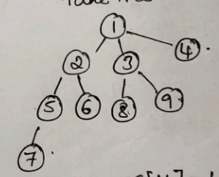
t : vector.

// output : DFS traversal of graph. (u,v) nodes reachable from u stored in vector t.

S[u] = 1 // source is visited so make it 1. // u = 0.

```
for every v adjacent to vertex u
  if v is not visited
     t[k][0] ← u
     t[k][1] → v
     k = k+1 ;
  dfs (a, n, u, s, t)
  end if
end for.
```

take tree



Adjacency Matrix.

	1	2	3	4	5	6	7	8	9
1	0	1	1	1	0	0	0	0	0
2	0	0	0	0	1	1	0	0	0
3	0	0	0	0	0	0	0	1	1
4	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	1	0	0
6	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0

S[u] = 1

S[1] = 0

S[2] = 0] Initial.

S[3] = 0

S[4] = 0

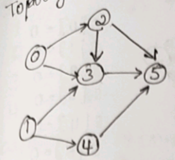
S[5] = 0

S[6] = 0

S[7] = 0

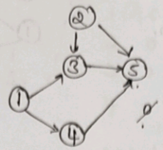
Topological Sort. [remove nodes whose indegree is 0]. consider visited.

Visited = 0, 1, 2, 3, 4, 5.

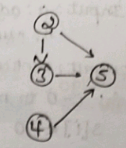


I.

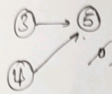
0	1	2	3	4	5.
0	0	1	3	1	3.



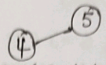
~~0~~	1	2	3	4	5.
	0	0	2	1	3.



~~0~~	~~1~~	2	3	4	5
		0	1	0	3.



~~0~~	~~1~~	~~2~~	3	4	5
			0	0	2.



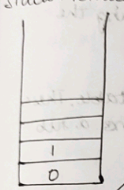
~~0~~	~~1~~	~~2~~	~~3~~	4	5
				0	1.

5.

~~0~~	~~1~~	~~2~~	~~3~~	~~4~~	5
					0.

[Topological Sorting → source removal method].

Stack Method.

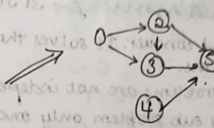


1) consider nodes with indegre 0.

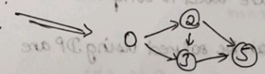
2) pop topmost ie 1.

3) next indegree 0 node = 4.

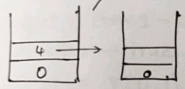
remove 4.



0	~~1~~	2	3	4	5
0		1	2	0	2

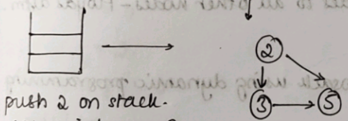


0	~~1~~	2	3	~~4~~	5.
0		1	2		2



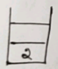
5) next indegree 0 = none

∴ remove 0.



~~0~~	~~1~~	2	3	~~4~~	5
		0	1		2.

4) push 2 on stack. cause indegree = 0.



5) pop 2.

6) push 3

7) pop 3 → 5

8) push/pop 5.

3 → 5

~~0~~	~~1~~	~~2~~	3	~~4~~	5
					0

∴ 1, 4, 0, 2, 3, 5.

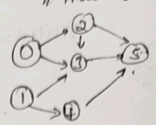
Topological sort of directed acyclic graph G = (V, E) where V = vertices E = edges.

• vertex u appears before v in the ordering.

u = source v = indegree (0) → adjacent node after u is removed.

Algorithm topological (n,a) / dfs(n,a).
// purpose : To obtain topological order.
// Input : a : adjacency matrix.
n : number of nodes in graph.
// output : vertices in topological order.

// Tracing.



S[0] = 0
S[1] = 0
S[2] = 0
S[3] = 0
S[4] = 0
S[5] = 0

```
for i=0 to n-1 do
    s[i] ← 0
end for.
    j←0.
for u←0 to n-1 do
    if (s[u]==0) call dfs(n,a) // topological(n,a)
end for.
```

dynamic programming. 31 July 2023.

↳ similar to divide and conquer. It solves the problem by combining the solution to sub problems.
↳ Applicable when sub problems are not independent.
↳ Algorithm solves every sub problem only once and is saved in a table. These values stored in table are used to compute the result every time a sub problem is encountered.
↳ various problems that can be solved using DP are
a) Binomial coeff.
b) knapsack
c) Find path matrix using warshalls algm.
d) compute shortest distance from all nodes to all other nodes – Floyds algm.

KnapSack Algm. (n, m, w, p, V)
// purpose : find the optimal solution for knapsack using dynamic programming.
// input : n : no/ of objects
m : capacity of knapsack
w : weight of all objects
p : profits of all objects.
// output : V : The optimal solution.

```
for i←0 to n do // no of items
    for j←0 to m do // capacity or weight
        if (i=0 or j=0)
            V[i,j] = 0
    else if (w[i] > j)
        V[i,j] = V[i-1,j]
    else
        V[i,j] = max( V[i-1,j] , V[i-1, j-w[i]] + p[i])
end if
end if end for end for.
```

Item (i)	(j)	
1	2	\$ 12
2	1	\$ 10
3	3	\$ 20
4	2	\$ 15.

capacity → W (weight) = 5.
$n = 4$ (items)

i \ j	0	1	2	3	4	5.
0	0	0	0	0	0	0 ← (1). (2), (3) ...
1	0	0	12	12	12	12.
2	0	10	12	22.	22	22.
3	0	10	12	22	30	32
4	0	10	12	25	30	37 → optimal solution.

1) $i=0$ $j=0$.
if ($i=0$ or $j=0$) →
$V[i,j]=0$

2) $i=0$ $j=1$
3) 4), 5) ...

6) $i=1$ $j=0$.
$i=1$ or $j=0$
⇒ $V[i,j]=0$.

7) $i=1$ $j=1$.
$i=0$ or $j=0$ ✗
$w[i]>j$
$w[1]>1$
weight of item1 = 2
∴ $2>1$ — true.
$V[1,1] = V[0,1]$
$= 0$

8) $i=1$ $j=2$.
$i=0$ or $j=0$ ✗.
$w[i]>j$
$2>2 \rightarrow$ ✗
else.
$V[1,2] = max(V[0,2], V[0, 2-2] + \overset{\text{profit of item1}}{12})$
$= max(0, 12)$
$= 12$

9) $i=1$ $j=3$
$2>3$ ✗.
else $[1,3] = max(V[0,3], V[0,1]+12)$
$= 12$.

10) $i=1$ $j=4$
$2>4$ ✗.
else $[1,4] = max(V(0,4), V(0,2)+12)$
$= 12$

11) $= 12$.

12) $i=2$ $j=0$.
$i=2$ or $j=0$ $= 0$

13) $i=2$ $j=1$
$W(i)=1$
$1>1$ ✗.
else $V[2,1] = max(V(1,1), V(1, 1-1)+10)$
$max(0, 0+10)$
$= 10$

14) $i=2$ $j=2$
$1>2$ ✗.
$V[2,2] = max(V(1,2), V(1,2-1)+10)$
$max(12, 10)$
$= 12$.

15) $i=2$ $j=3$
$1>3$ ✗.
$V[2,3] = max(V(1,3), V(1,3-1)+10)$
$= (12, 22)$
$= 22$.

16) $i=2$ $j=4$
$(2,4) = max(V(1,4) + V(1,3)+10)$
$= M(12+22)$
$= 22$

Analysis:

$$\sum_{i=0}^{n}\sum_{j=0}^{m} 1 = \sum_{i=0}^{n} m-0+1 = \sum_{i=0}^{n}(m+1) = \sum_{i=0}^{n}(m+1)\,1$$

constant

$$= m+1 \sum_{i=0}^{n} 1 = m+1\,(n+1)$$
$$= mn + m + n + 1$$

neglect smaller terms.

$C(n) = \Theta(mn)$.

Binomial Coefficient:

Algm. Binomial coefficient (n, k)

// purpose: to compute binomial coefficient using DP.

// Input: non negative integer $n \geq k \geq 0$

// output: value of ^nC_k.

```
for i <- 0 to n do.
  for j <- 0 to min(i,k) do.
    if (j=0 or i=j)
      c[i,j] <- 1
    else
      c[i,j] <- c[i-1,j-1] + c[i-1, j]
    end if
  end for
end for.
```

^nC_k

$^nC_k = {}^4C_2$

$= \frac{4!}{2!(4-2)!}$

$= \frac{4!}{2\times 2} = \frac{24}{4} = 6.$

Ex. $^4C_2$ $n=4$ $k=2$.

	0	1	2 → k
0	1		
1	1	1	
2	1	2	1
3	1	3	3
4 ↓ n	1	4	6

6 req. ans.

1) $i=0 \leftarrow 4$ $j \leftarrow 0$ to $\min(0,2) = 0$

$i=0$ $j=0$

$\Rightarrow i=j \Rightarrow C[0,0]=1$

$i=1$ $j=0$ to $\min(1,2)$

$j=0$ to 1.

$i=1$ $j=0$.

$i \neq j$ but $j=0$

$\therefore C[1,0]=1$

$i=1$ $j=1$

$i=j$ $\therefore C[1,1]=1$

$i=2$ $j=0$ to $\min(2,2)$
$(0,2)$

$i=2$ $j=0$

$i \neq j$ but $j=0$

$C[2,0]=1$

$i=2$ $j=1$.

$i \neq j$ $j \neq 0$.

$\therefore C[2,1] = C[1,0] + C[1,1]$

$= 1+1$

$= 2$

$i=3$ $j=0 \rightarrow 2$.

$i=3$ $j=0 \Rightarrow C(3,0)=1$

$i=3$ $j=1 \Rightarrow C[3,1] = C[2,0] + C[2,1]$

$= 1+2$

$= 3$

$i=3$ $j=2$.

$C[3,2]$

$= C[2,1] + C[2,2]$

$= 2+1 = 3$

$i=4$ $j=0 \Rightarrow 1$

$i=4$ $j=1$

$C[4,1] = C[3,0] + C[3,1]$

$= 4$

$i=4$ $j=2$.

$C[4,2] = C[3,1] + C[3,2]$

$= 3+3$

$= 6$.

analysis.

	0	1	2
0	1		
1	1	1	
k (2)	1	2	1
k+1 (3)	1	3	3
n (4)	1	4	6

→ Right Δ.

rectangle.

∴ Analysis.

$$\underbrace{\sum_{i=1}^{k}\sum_{j=1}^{i-1} 1}_{\text{Right }\Delta} + \underbrace{\sum_{i=k+1}^{n}\sum_{j=1}^{k} 1}_{\text{Rectangle}}$$

$$= \sum_{1}^{k} i-1-\cancel{1}+\cancel{1} + \sum_{k+1}^{n} k-\cancel{1}+\cancel{1}$$

$$= \sum_{1}^{k} i-1 + k\sum_{k+1}^{n} 1.$$

$$= 0+1+2+3\ldots k-1 + k[n-k-\cancel{1}+\cancel{1}].$$

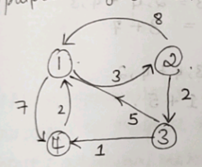
$$= \frac{(k-1)(k)}{2} + k(n-k)$$

$$= \frac{k^2-k}{2} + nk-k^2 = \frac{k^2-k+2nk-2k^2}{2} = \frac{2nk-k-k^2}{2}.$$

$$\frac{2nk}{2} - \cancel{\frac{k}{2}} - \cancel{\frac{k^2}{2}} = \underline{\underline{nk}}.$$

Floyd's Algorithm.

// purpose : to find shortest distance from all nodes to all other nodes in the graph.



adjacency Matrix. — if direct path - not present => put 999
└ if no loop - put = 0.

$$A^0 = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 3 & 999 & 7 \\ 2 & 8 & 0 & 2 & 999 \\ 3 & 5 & 999 & 0 & 1 \\ 4 & 2 & 999 & 999 & 0 \end{array}$$

↳ cost adjacency matrix.

* Write value of path if path exists.

$999 \simeq \infty$.

A' (1 as intermediate)

$$A' = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 3 & 999 & 7 \\ 2 & 8 & 0 & 2 & 15 \\ 3 & 5 & 8 & 0 & 1 \\ 4 & 2 & 5 & 999 & 0 \end{array}$$

// copy 1st row, 1st column.

// 2,2 → 0
2,3 → reach using 1.
2,3 = 2,1 + 1,3.
2 = 8 + 999.
2 = 999 (∞).
shortest = 2.

// 2,4
2,4 = 2,1 + 1,4
999 = 8 + 7
999 = 15.
shortest = 15.

// 3,2.
3,2 = 3,1 + 1,2.
999 = 5 + 3
999 = 8.
shortest = 8.

// 3,4.
3,4 = 3,1 + 1,4
1 = 5 + 7.
1 = 12.
shortest = 1.

// 4,2.
4,2 = 4,1 + 1,2.
999 = 2 + 3 = 5.

// 4,3.
4,3 = 4,1 + 1,3
999 = 2 + 999

A^2 (2 as intermediate) copy 2nd row 2nd column (use A1)

$$\begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 3 & 5 & 7 \\ 2 & 8 & 0 & 2 & 15 \\ 3 & 5 & 8 & 0 & 1 \\ 4 & 2 & 5 & 7 & 0 \end{array}$$

// 1,3 = 1,2 + 2,3.
999 = 3 + 2 = 5.

// 1,4 = 1,2 + 2,4
7 = 3 + 15.
7 = 18.
∴ choose 7.

// 3,1
5 = 3,2 + 2,1
5 = 8 + 8
5 = 16.

// 3,4
1 = 3,2 + 2,4.
1 = 8 + 15.
1 = 23

// 4,1
2 = 4,2 + 2,1.
2 = 5 + 8
2 = 13

// 4,3.
999 = 4,2 + 2,3.
999 = 5 + 2
999 = 7.

A^3 (3 as inter-mediate) copy 3 row 3 co

$$\begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 3 & 5 & 6 \\ 2 & 7 & 0 & 2 & 3 \\ 3 & 5 & 8 & 0 & 1 \\ 4 & 2 & 5 & 7 & 0 \end{array}$$

// 1,2 = 1,3 + 3,2.
3 = 5 + 8
keep 3.

// 1,4 = 1,3 + 3,4
7 = 5 + 1
7 = 6

// 2,1 = 2,3 + 3,1
8 = 2 + 5.
8 = 7.

// 2,4 = 2,3 + 3,4
15 = 2 + 1
15 = 3.

// 4,1 = 4,3 + 3,1
2 = 7 + 5.

// 4,2 = 4,3 + 3,2
5 = 7 + 8.

A4.

$$\begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 3 & 5 & 6 \\ 2 & 5 & 0 & 2 & 3 \\ 3 & 3 & 6 & 0 & 1 \\ 4 & 2 & 5 & 7 & 0 \end{array}$$

// 1,2 = 1,4 + 4,2.
3 = 6 + 5.

// 1,3 = 1,4 + 4,3
5 = 6 + 7.

// 2,1 = 2,4 + 4,1
7 = 3 + 2.
= 5.

// 2,3 = 2,4 + 4,3
2 = 3 + 7

// 3,1 = 3,4 + 4,1
5 = 1 + 2.
= 3

// 3,2 = 3,4 + 4,2.
8 = 1 + 5.

1 —3— 2
1 —5— 3 (1,2 + 2,3)
1 —6— 4 (1,2 + 2,3 + 3,4)

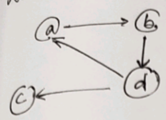
|| similarly follow.

Algm. Floyds (n, cost)

```
// purpose : To implement floyds algm.
// Input : cost adjacency matrix.
// output : Shortest distance. matrix nxn.
for i←0 to n-1 do
  for j←0 to n-1 do
    D[i,j] ← cost[i,j]
  end for
end for.

for k←0 to n-1 do
  for i←0 to n-1 do
    for j←0 to n-1 do
      D[i,j] = min( D[i,j], D[i,k]
                      + D[k,j
    end for
  end for
end for.
```

Warshalls algm : All pair shortest path. - Use Floyd only but entries are 0,1.



$$
\begin{array}{c|cccc}
 & a & b & c & d \\
\hline
a & 0 & 1 & 0 & 0 \\
b & 0 & 0 & 0 & 1 \\
c & 0 & 0 & 0 & 0 \\
d & 1 & 0 & 1 & 0
\end{array} = A^0
$$

// Floyds → cost matrix.
Warshalls → 0, 1's only
check if path there.
process similar.
if path → 1
else 0.

A^1

$$
\begin{array}{c|cccc}
 & a & b & c & d \\
\hline
a & 0 & 1 & 0 & 0 \\
b & 0 & 0 & 0 & 1 \\
c & 0 & 1 & 0 & 0 \\
d & 1 & 1 & 1 & 0
\end{array}
$$

// b,a = 0
// b,c = b,a + a,c.
= 0 + 0
= 0.

// b,d = b,a + a,d.
1 = 0 + 0.
as path exists,
b,d = 1.

// c,b = c,a + a,b.
0 = 0 + 1.
0 = 1.
$\boxed{\text{Choose 1}}$
will be replaced later.

// c,d = c,a + a,d
0 = 0 + 0.

// d,b = d,a + a,b
0 = 1 + 1.

// d,c = d,a + a,c
1 = 1 + 0.

final

$$
\begin{array}{c|cccc}
 & a & b & c & d \\
\hline
a & 0 & 1 & 1 & 1 \\
b & 1 & 0 & 1 & 1 \\
c & 0 & 0 & 0 & 0 \\
d & 1 & 1 & 1 & 0
\end{array}
$$

A^2

$$
\begin{array}{c|cccc}
 & a & b & c & d \\
\hline
a & 0 & 1 & 1 & 1 \\
b & 0 & 0 & 0 & 1 \\
c & 1 & 1 & 0 & 1 \\
d & 1 & 1 & 1 & 0
\end{array}
$$

// a,c = a,b + b,c
0 = 1 + 0

// a,d = a,b + b,d
0 = 1 + 1

// c,a = c,b + b,a
0 = 1 + 0

// c,d = c,b + b,a
0 = 1 + 0

// d,a = d,b + b,a
1 = 1 + 0

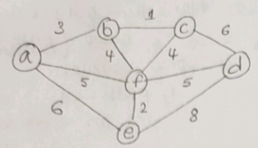
// d,c = d,b + b,c
1 = 1 + 0

// Algm Warshalls (n, A, P)
purpose : to compute transitive closure
// input : adjacency matrix.
// output : transitive closure

```
for i←0 to n-1 do
  for j←0 to n-1 do
    P[i,j] ← A[i,j].
  end for
end for.
```

```
for k←0 to n-1 do.
  for i←0 to n-1 do
    for j←0 to n-1 do
      P[i,j] = min( P[i,j], P[i,k]
                    + P[k,j
    end for
  end for
end for.
```

3-Aug Greedy Technique:



a) Prims Algorithm.

↳ creates a spanning subtree of main graph [includes all vertices and only the shortest paths b/w them].

Source = a.

a $\xrightarrow{\cancel{3}}$ b → // least.
a $\xrightarrow{\cancel{5}}$ f
a $\xrightarrow{\cancel{6}}$ e

b $\xrightarrow{\cancel{1}}$ c // least
b $\xrightarrow{\cancel{4}}$ f // least 1

c $\xrightarrow{6}$ d
4 ✗ f // least 2 — consider any 1.

f $\xrightarrow{5}$ d // least path to d.
2 ✗ e // least

Graph. (should not form cycles)

a —3— b —1— c
b —4— f
f —5— d
f —2— e

Total cost (minimum cost).
= 3 + 1 + 4 + 5 + 2.
= 15 //.

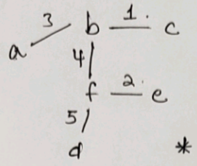
Tree Vertices	Remaining Vertices	Illustration.
a(–, –)	b(a,3)✓ c f(a,5) d e(a,6)	a —3— b
b(a,3)	c(b,1)✓ d. f(b,4) e(a,6)	a —3— b —1— c
c(b,1)	d(c,6) f(b,4)✓ e(a,6).	a —3— b —1— c b —4— f
f(b,4)	e(f,2)✓ e(a,6) d(c,6) d(f,5)	a —3— b —1— c b —4— f f —2— e
e(f,2)	~~e(a,6)~~ d(c,6) d(f,5)✓. d(e,8)	a —3— b —1— c b —4— f f —5— d f —2— e

b) Krushal's Algorithm.

- Write all costs / edges in increasing order.

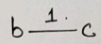
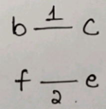
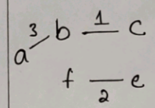
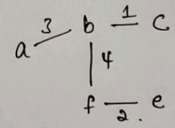
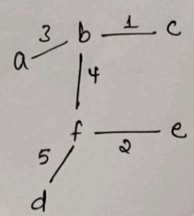
b – c = 1 ✓
f – e = 2 ✓
a – b = 3 ✓
b – f = 4 ✓
c – f = 4 × will form cycle.
a – f = 5 × forms cycle
f – d = 5 ✓
a – e = 6 ×
c – d = 6
d – e = 8.

f visited



= 3+1+4+2+5.
= 15//.

* once a vertice visited – neglet all other edges that pass through it to prevent formation of cycles.

Tree Edges.	Sorted list of Edges.	Illustration.
—	b-c = 1, f-e = 2, a-b = 3, b-f 4, c-f 4, a-f = 5, f-d = 5, a-e = 6, c-d = 6, d-e = 8	
bc 1.	ef 2, ab 3, bf 4, cf 4, af 5, df 5, ae 6, cd 6, de 8	
ef 2.	ab 3, bf 3, cf 4, af 5, df 5, ae 6, cd 6, de 8	
ab 3	bf 4, ~~cf 4~~, ~~af 5~~ neglet as f,e d visited., ~~df 5~~, ~~ae 6~~, ~~cd 6~~, ~~de 8~~	
bf 4, df 5	~~ae 6~~, ~~de 8~~ neglet as e visited., ~~cf = 4~~, ~~af 5~~, ~~df 5~~, ~~cd 6~~.	

minimum Total cost = 15//.