Multistage graphs

Algorithm FGraph (G, k, n, P)

{

cost[n] := 0.0;

for j := n-1 to 1 step -1 do

{

let r be a vertex such that (j, r) is an edge of G and c[j, r] + cost[r] is minimum;

cost[j] = c[j, r] + cost[r];

d[j] := r;

}

P[1] := 1, P[k] := n;

for j := 2 to k-1 do P[j] = d[P[j-1]];

}

Formula:-

$$bcost(i,j) = \min_{\substack{l \in n-1 \\ (l,j) \in E}} \{ bcost(i-1, l) + c(l, j) \}$$

Algorithm BGraph (G, k, n, P)

{

bcost[1] = 0.0;

for j := 2 to n do

{

let r be such that (r, j) is an edge of G and c[j, r] + bcost[r] + c[r, j] is minimum;

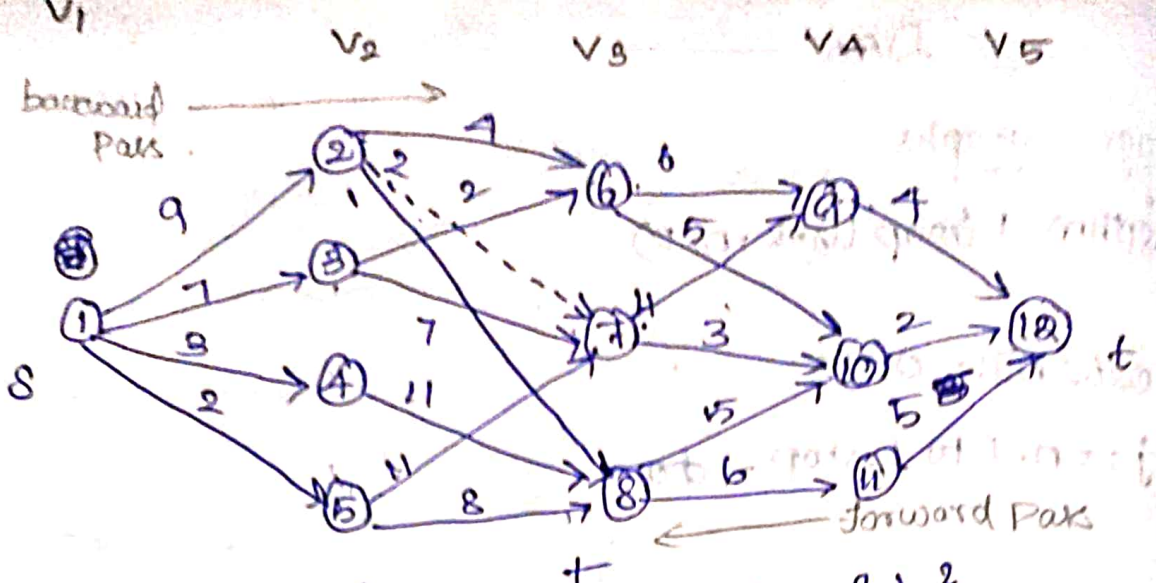
bcost[j] := bcost[r] + c[r, j];

d[j] := r;

}

P[1] := 1; P[k] := n;

for j := k-1 to 2 do P[j] = d[P[j+1]];



$$cost(i,j) = \min \{ c(j,l) + cost(i+1,l) \}$$

$i \le v_{i+1}$

$(i,j) \in E$

$|V_5| = 1 \quad |V_3| = 3 \quad |V_2| = 4$

$|V_4| = 3 \quad |V_1| = 1$

K = Stage of Graph

$G = (V, E) \quad K \ge 2.$

$K = 5$

l = vertices

Formula :- (Forward Pass).

$$\rightarrow cost(i,j) = \min \{ c(j,l) + cost(i+1,l) \}$$

(1) $cost(k-1,j) = c(j,k)$ for $k = 5$

$cost(5,12) = 0.$

$\boxed{cost(5-1, j) = c(j,k)}$

$cost(4,9) = 4$

$cost(4,10) = 2$

$cost(4,11) = 5.$

(2)

$$cost(3,6) = \left\{ \begin{array}{l} c(6,9) + cost(4,9) \\ 6 + 4 = 10 \\ c(6,10) + cost(4,10) \\ 5 + 2 = 7 \end{array} \right\} = 7.$$

$$cost(3,7) = \left\{ \begin{array}{l} c(7,9) + cost(4,9) \\ 4 + 4 = 8 \\ cost(7,10) + cost(4,10) \\ 3 + 2 = 5 \end{array} \right\} = 5.$$

$$cost(3,8) = \left\{ \begin{array}{l} c(8,10) + cost(4,10) \\ 5+2 = 7 \\ c(8,11) + cost(4,11) \\ 6+5 = 11 \end{array} \right\} = 7$$

Above vertical.

$$cost(2,2) = \left\{ \begin{array}{l} c(2,6) + cost(3,6) \quad cost(3,7) \quad c(3,8) \\ + cost(3,8) \quad 2 \quad + 5 \\ 4+7=11 \quad 7 \quad 1+7 \\ \quad\quad\quad\quad\quad\quad\quad\quad 8. \end{array} \right\}$$

$$= 7.$$

$$cost(2,3) = \left\{ \begin{array}{l} c(3,6) + cost(3,6), c(3,7) + cost(3,7) \\ 2+7=9 \quad\quad 7+5 = 11 \end{array} \right\} = 9$$

$$cost(2,4) = \left\{ \begin{array}{l} c(4,8) + cost(3,8) \\ 11+7 \end{array} \right\} = 18.$$

$$cost(2,5) = \left\{ \begin{array}{l} c(5,7) + cost(3,7), c(5,8) + cost(3,8) \\ 11+5 = 16 \quad\quad 8+7 \end{array} \right\} = 15$$

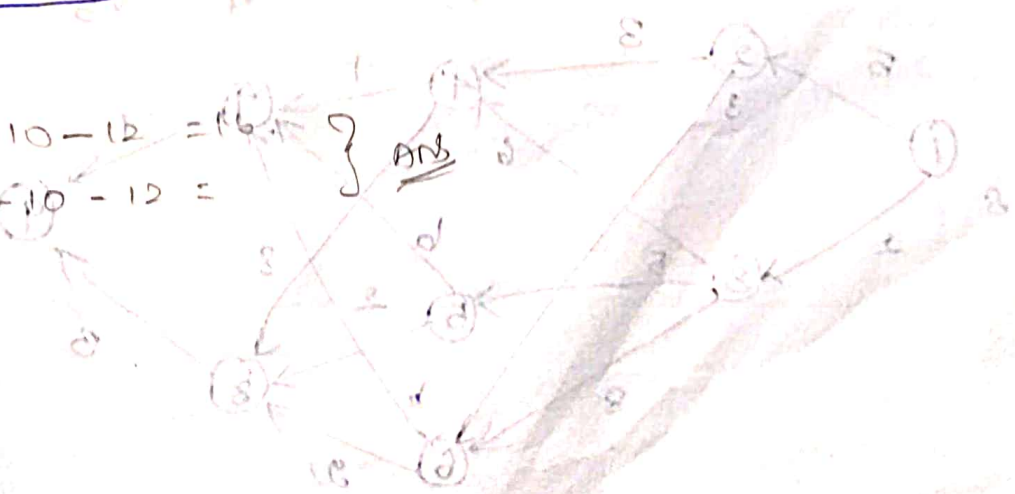
$$cost(1,1) = \left\{ \begin{array}{l} c(1,2) + cost(2,8), \quad c(1,3) + cost(2,3), \\ 9+7 = 16 \quad\quad 7+9 = 16 \\ c(1,4) + cost(2,4), \quad c(1,5) + cost(2,5) \\ 3+18 = 21 \quad\quad 2+15 = 17 \end{array} \right\}$$

$cost(1,1) = 16.$

Sol: (A(SO)

1 - 2 - 7 - 10 - 12 = 16 } Ans

1 - 3 - 6 - 10 - 12 = }



Backward path:-

Bcost (8,8) = 9

Bcost (2,3) = 7

Bcost (8,4) = 3

Bcost (8,5) = 2;

$$Bcost(3,6) = \left\{ \begin{array}{l} Bcost(8,8) + c(3,6) \\ 9+4 = 13 \\ Bcost(8,3) + c(3,6) \\ 7+2 = 9 \end{array} \right\} = 9.$$

$$Bcost(3,7) = \left\{ \begin{array}{l} Bcost(8,2) + c(2,7) \\ 9+2 = 11 \\ Bcost(2,3) + c(3,7) \\ 7+4 = 11 \end{array} \right\} = 11.$$

$$Bcost(8,8) = \{10, 14, 10\} = 10.$$

Bcost(4,9) = {15, 15) = 15

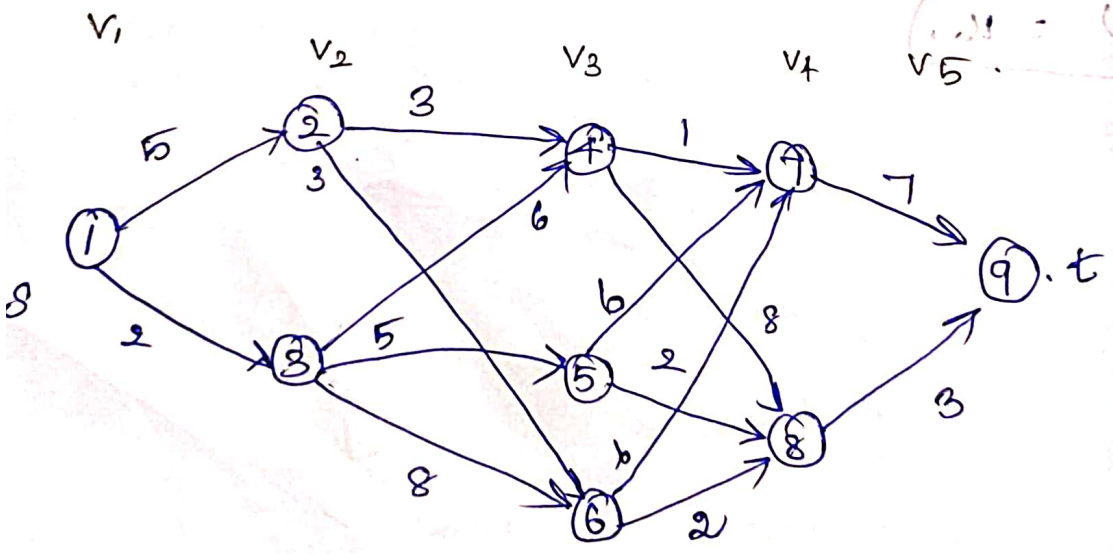
Bcost (4,10) = {14, 14, 15} = 14,

Bcost (4,11) = {10+6} = 16.

B'cost (5,12) = 16 → min)

Bcost (8,12) : 16

Sum :-



Multistage Graphs

Definition:

*A multistage graph $G=(V,E)$ is a directed graph where vertices are partitioned into k (where $k \geq 2$) disjoint sets V_i, $1 \leq i \leq k$.

*The set V_1 and V_k are such that $|V_1| = |V_k| = 1$.

*The vertex s is the source, and t the sink. Let $c(i,j)$ be the cost of edge $[i,j]$. The cost of a path from s to t is the sum of the costs of the edges on the path.

*The Multistage graph problem is to find a minimum-cost path from s to t.

Forward approach Formula:

$$cost(i,j) = \min\{c(j,l) + cost(i+1,l)\}$$

$l \in V_{i+1}$

$[j,l] \in E$.

Backward approach formula:-

$$bcost(i,j) = \min\{bcost(i-1,l) + c(l,j)\}$$

$l \in V_{i-1}$

$[l,j] \in E$

Forward Algorithm:

```
Algorithm FGraph (G, k, n, P)
// The input is a k-stage Graph G = (V, E) with n vertices
// indexed in order of stage. E is a set of edges and c[i,j]
// is the cost of (i,j). P[1:k] is a minimum-cost path.
{
  cost[n] := 0.0;
  for j := n-1 to 1 step -1 do
  { // compute cost[j].
  Let r be a vertex such that (j,r) is an edge
  of G and c[j,r] + cost[r] is minimum;
  cost[j] := c[j,r] + cost[r];
  d[j] := r;
  }
  // Find a minimum-cost path.
  P[1] := 1; P[k] := n;
  for j := 2 to k-1 do P[j] := d[P[j-1]];
  }
```

The input is a k-stage graph $G = (V, E)$ with n vertices indexed in order of stages. E is a set of edges and c[i,j] is the cost of [i,j].

P[1:k] is a minimum-cost path.

Algorithm BGraph (G, k, n, P)

// Same function as FGraph

{

bcost[1] := 0.0;

for j := 2 to n do

{// compute bcost[j].

Let r be such that [r, j] is an edge of G and bcost[r] + c[r, j] is minimum;

bcost[j] := bcost[r] + c[r, j];

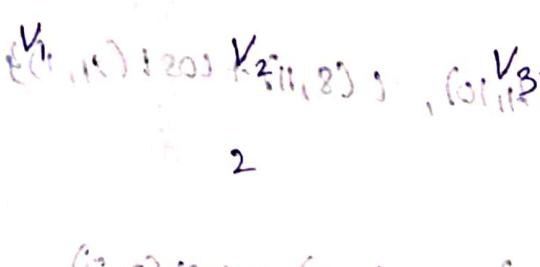
}

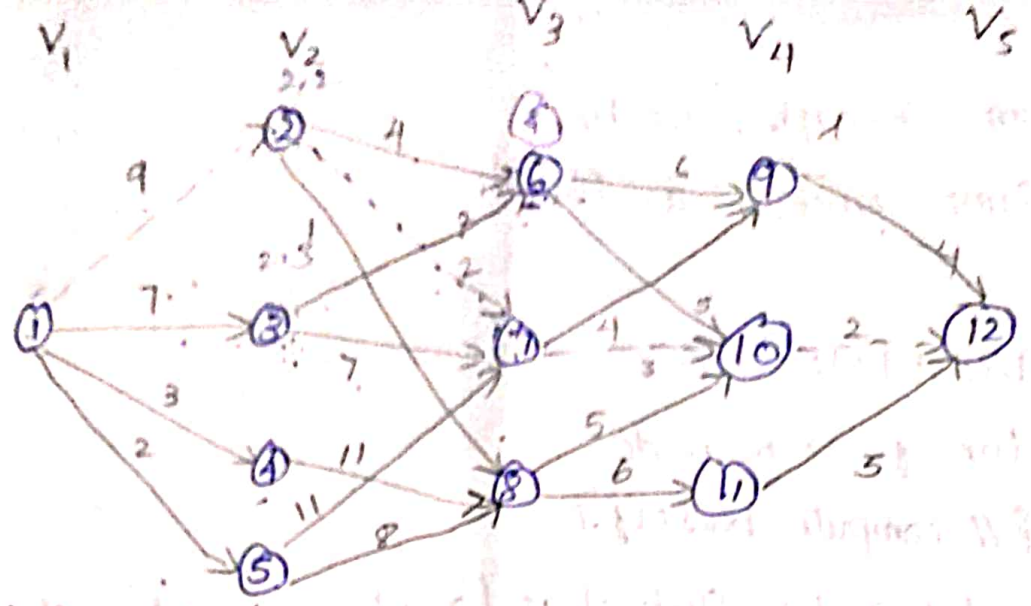
P[1] := 1; P[k] := n;

For j := k-1 to 2 do P[j] := d[P[j+1]];

}

Ex:-





$cost(i,j) = \min\{c(j,l) + cost(i+1,l)\}$

$cost(4,9) = 4$

$cost(4,10) = 2$

$cost(4,11) = 5$

$cost(3,6) = \min\{c(6,9) + cost(4,9), c(6,10) + cost(4,10)\}$

$= 4+4, 3+2$

$= 7$

$cost(3,7) = \min\{c(7,9) + cost(4,9), c(7,10) + cost(4,10)\}$

$= 4+4, 3+2$

$= 5$

$cost(3,8) = \min\{c(8,10) + cost(4,10), c(8,11) + cost(4,11)\}$

$= 5+2, 6+5$

$= 7$

$cost(2,2) = \min\{c(2,6) + cost(3,6), c(2,7) + cost(3,7), c(2,8) + cost(3,8)\}$

$= 4+7, 2+5, 1+7$

$= 7$

$cost(2,3) = \min\{c(3,6) + cost(3,6), c(3,7) + cost(3,7)\}$

$= 2+7, 7+5$

$= 9$

$cost(2,4) = \min\{c(4,8) + cost(3,8)\}$

$= 11+7$

$= 18$

cost(3,5) = min {c(5,7) + cost(3,7), c(5,8) + cost(3,8)}

= 11+5, 8+7

= 15.

cost(1,1) = min {c(1,2) + cost(2,2), c(1,3) + cost(3,8),
c(1,4) + cost(2,4), c(1,5) + cost(2,5)}

= 9+7, 7+9, 3+18, 2+15

= 16

bcost(i,j) = min {bcost(i-1,l) + c(l,j)}

bcost(2,2) = 9

bcost(2,3) = 7

bcost(2,4) = 3

bcost(2,5) = 2

bcost(3,6) = Min {bcost(2,2) + c(2,6), bcost(2,3) + c(3,6)}

= 9+4, 7+2 = 9

bcost(3,7) = min {bcost(2,2) + c(2,7), bcost(2,3) + c(3,7),
bcost(2,5) + c(5,7)}

= 9+2, 7+7, 2+11

= 11

bcost(3,8) = Min {bcost(2,2) + c(2,8), bcost(2,4) + c(4,8),
bcost(2,5) + c(5,8)}

= 9+1, 3+11, 2+8 = 10

bcost(4,9) = Min {bcost(3,6) + c(6,9), bcost(3,7) + c(7,9))

= 9+6, 11+4

= 15

bcost(4,10) = min {bcost(3,6) + c(6,10), bcost(3,7) + c(7,10),
bcost(3,8) + c(8,10)}

= 9+5, 11+3, 10+5

= 14

bcost(A,11) = Min {bcost(3,8) + c(8,11)}

= 10+6

= 16

bcost(5,12) := Min {bcost(4,9) + c(9,12), bcost(4,10) + c(10,12)

{bcost(4,11) + c(11,12)}

= 15+4, 14+2, 16+5

= 16

1-2-7-10-12.

Algorithm kruskal(E, cost, n, t) T(:- O(E log E) O(|E| log |V|)

{

construct a heap out of the edge costs using Heapify;

for i := 1 to n do parent[i] := -1;

i = 0; mincost := 0.0;

while ((i < n-1) and (heap not empty)) do

{

Delete a minimum cost edge (u,v) from the heap

and reheapify using adjust;

j := find(u); k := find(v);

if (j ≠ k) then

{

i := i+1;

t[i,1] := u; t[i,2] := v;

mincost := mincost + cost[u,v];

union(j,k);

}

}

if (i ≠ n-1) then write "No spanning tree");

else return mincost;

}

All-Pairs Shortest Path :-

Algorithm Allpaths (cost, A, n)

// cost [1:n, 1:n] is the cost adjacency matrix of graph with

// n vertices; A[i,j] is the cost of a shortest path from vertex i to vertex j. cost[i,i] = 0.0, for $1 \le i \le n$.

{

for i = 1 to n do,

for j = 1 to n do,

A[i,j] := cost[i,j];

for k := 1 to n do,

for i := 1 to n do,

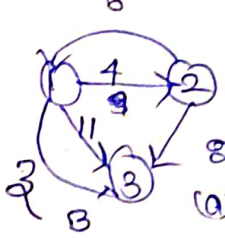
for j := 1 to n do,

A[i,j] := min(A[i,j], A[i,k] + A[k,j]);

$$A(i,j) = \min_{1 \le k < n} \{ \min \{ A^{k-1}(i,j) + A^{k-1}(k,j) \} \}$$

$$\Rightarrow A^k(i,j) = \min \{ A^{k-1}(i,j), A^{k-1}(i,j) + A^{k-1}(k,j) \},$$

↳ To calculate formula. $k \ge 1$



(a) → Example diagram.

A^0	1	2	3
1	0	4	11
2	6	0	2
3	3	∞	0

(b) A^0

A^1	1	2	3
1	0	4	11
2	6	0	2
3	3	7	0

(c) A^1

A^2	1	2	3
1	0	4	6
2	6	0	2
3	3	7	0

(d) A^2

A^3	1	2	3
1	0	4	6
2	5	0	3
3	3	7	0

(e) A^3

$A^1[3,2) = \min\{\infty, 3+4\} = 7$

$A^1(1,2) = \min\{4, 0+4\} = 4$

Ans
Length of A.

$A^2(1,2) = \min(A^1(1,2), A^1(1,2) + A^1(3,2))$

$A^2(1,2) = \min(4, 4+2) = 6.$

$A^3(2,1) = \min(6, 2+3) = 5.$

$t = \Theta(n^3)$

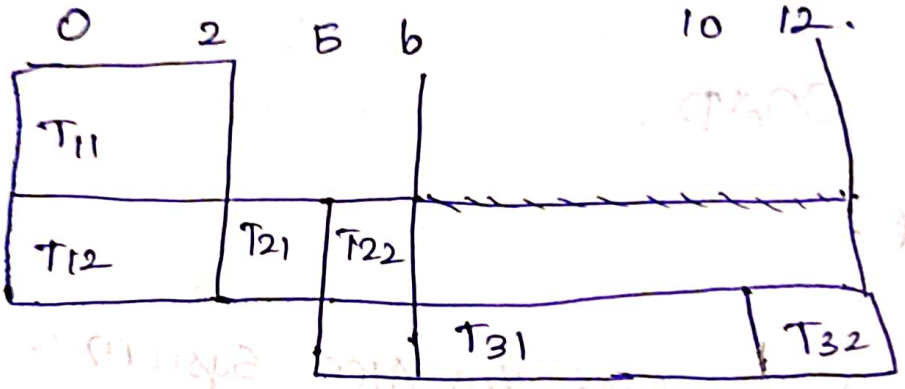
DDA.

Flow shop scheduling :- SM/1 (Multiple Job multiple Process)

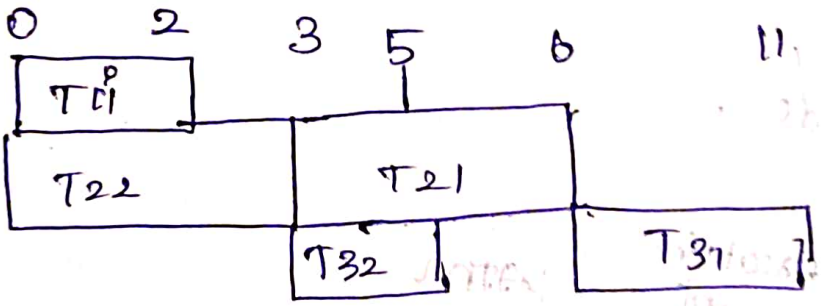
$$j = \begin{bmatrix} 2 & 0 \\ 3 & 3 \\ 5 & 2 \end{bmatrix}$$ Task. (Job)

Job 1 = 10.
Job 2 = 12.
waiting $J_2 = 7$.
waiting $J_1 = 0$.

time



time



Job 1 = 10.
waiting $J_1 = 0$.
Job 2 = 10.
waiting $J_2 = 1$

{ $F(s) = \max_{1 \le i \le n} \{ f_i(s)^2 \}$. $f_1(s) = 10$. , $f_2(s) = 11$ =

mean flow time.

{ $MFT(s) = \frac{1}{n} \sum_{1 \le i \le n} f_i(s)$. $f_2(s) = 12$. , $f_2(s) = 5$

i = Job number. = n - of job requi -

j : Job task = m - task.

Tij = Pj → Processor.

↳ time required to complete task — Eij.

OFT - Optimal finishing Time.

OMFT - Optimal mean finishing time.

POMFT - Premetive optimial mean flow time.

The Traveling salesperson problem :

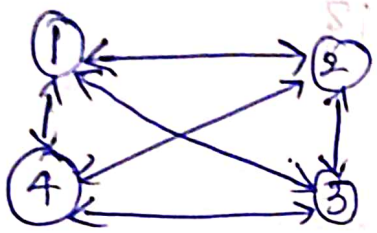
$$g(1, V-\{1\}) = \min_{2 \le k \le n} \{ c_{1k} + g(k, V-\{1,k\}) \}$$

We obtain (for $i \neq 1$)

$$g(i,s) = \min_{j \in s} \{ c_{ij} + g(j, s-\{j\}) \}.$$

$$N = \sum_{k=0}^{n-2} (n-1) \binom{n-2}{k} = (n-1)2^{n-2}.$$

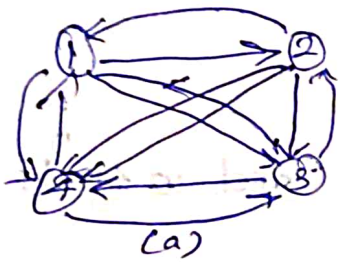
Directed Graph :



(a)

	1	2	3	4
1	0	10	15	20
2	5	0	9	10
3	6	13	0	12
4	8	8	9	0

(b)



(a)

$G = (V, E)$ $(i,j) \in E$

$c_{ij} > 0$

c_{ij} : $(i,j) \in E$

$c_{ij} = \infty$

$n = |V| = 4$ $\therefore n$ is number of vertices.

directed simple.

$\Rightarrow$ Tour of G is a simple cycle. (Hamiltonian cycle)

$\therefore$ Simple cycle :- Tour without repeat

If we find out the performance;

$\Rightarrow$ Sum of the cost edges on the Tour.

$$\boxed{g(i, \phi) = c_{i1}.} \quad |S| = \phi. \qquad 1 \le i \le n.$$

$\therefore$ Now Number is defined;

$g(4, \phi) = 8$

$g(2, \phi) = 5$

$g(3, \phi) = 6$

$g(4, \phi) = 8.$

$g(2,\{3\}) = \min\{c_{23} + g(3,\phi)\} = 15$

$g(2,\{4\}) = \min\{c_{24} + g(4,\phi)\} = 18$

$g(3,\{2\}) = \min\{c_{32} + g(2,\phi)\} = 18$

$g(3,\{4\}) = \min\{c_{34} + g(4,\phi)\} = 20$

$g(4,\{2\})$

$g(4,\{3\})$ ..

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

$g(2,\{3,4\}) = \min\{c_{23} + g(3,\{4\}), c_{24} + g(4,\{3\})\}$

$9 + 12 + 10 + 15 = 25$

$g(3,\{4\}) = \min\{c_{32} + g(2,\{4\}), c_{34} + g(4,\{2\})\}$

$13 + 18 + 12 = 43$

$= 25$

$g(3,\{2\}) = \min\{c_{42} + g(2,\{3\}), c_{43} + g(3,\{2\})\}$

$8 + 15 \qquad 9 + 11$

$= 23$

$g(1,\{2,3,4\}) = \min\{c_{12} + g(2,\{3,4\}), c_{13} + g(3,\{2,4\}), c_{14} + g(4,\{2,3\})\}$

$= \min\{12 + 25, 15 + 23, 10 + 25\}$

$= 35$

35 is the minimum cost

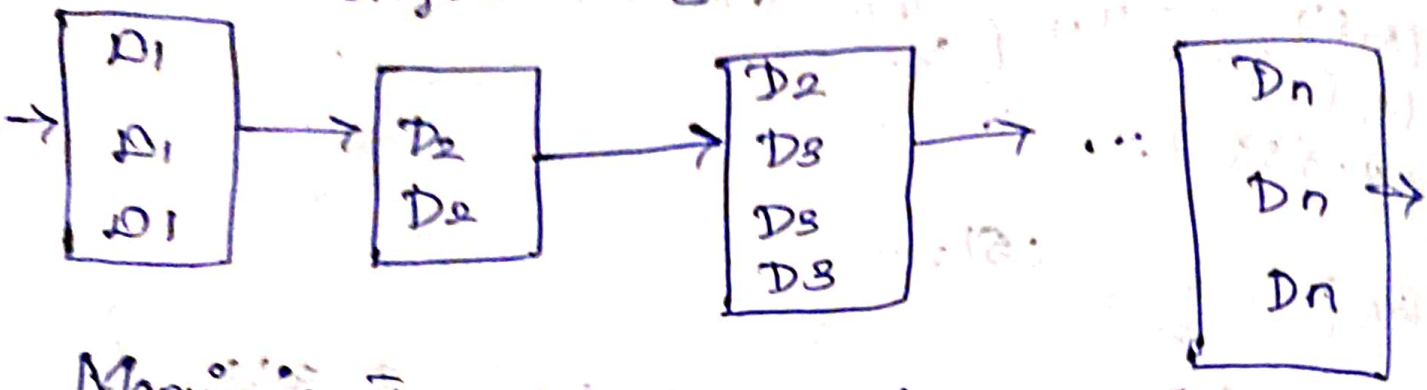
1 - 2 - 4 - 3 - 1

10
10
9
6
35

Minimum cost of Route = 35

Space $\Rightarrow O(n2^n)$ $T(n) = O(n^2 2^n)$

RELIABILITY DESIGN :



Maximize $\prod_{1 \le i \le n} \phi_i(m_i)$ ↳ overall function 8.

Subject to $\sum_{1 \le i \le n} c_i m_i \le c$

$m_i \ge 1$ and integer $1 \le i \le n$ → i^{th} device cost.

$u_i = \left\lfloor \left(c + c_i - \sum_{1}^{n} c_j \right) / c_i \right\rfloor$

↳ Total cost

R_i – reliability of device of P_i.

$r_1 = .9$ $r_2 = .85$ $r = .8$

$D_1 = .90$ $D_2 = 85$

Reliability Properbility of Entire system $= \prod r_i$

↳ Product

$= .9 \times .85 \times .8$

Ans ≥

0/1 knapsack :- —— DDA —— 21/10/20.

~~Final~~ formula :-

$$f_n(m) = \max \{ f_{n-1}(m), f_{n-1}(m), f_{n-1}(m - w_n) + p_n \}$$

for arbitrary $f_i(y)$; $i \geq 0$.

$$f_i(y) = \max \{ f_{(i)-1}(y), f_{i-1}(y - w_i) + p_i \}$$

$$S_1^i = \{ (P,W) \mid (P,R, W - w_i) \in S^i \}.$$

Algo :-

Algorithm DKP (P, w, n, m)

{

$S^0 := \{(0,0)\}$;

for $i := 1$ to $n-1$ do

{

$S^{i-1} := \{ (P,W) \mid (P - P_i, W - w_i) \in S^{i-1}$ and $w \leq m \}$;

$S^i := \text{MergePure}(S^{i-1}, S_1^{i-1})$;

}

$(PX, WX) :=$ last Pair S^{n-1};

$(PY, WY) := (P' + P_n, W' + w_n)$ where W' is the largest W in any pair in S^{n-1} such that $W + w_n \leq m$.

if $(PX > PY)$ then $x_n := 0$;

else

$x_n := 1$;

Trace BackFor $(x_{n-1}, \ldots, x)$;

}

Ans ⇒

UNIT-IV Basic Traversal & Searching Technique

⇒ Techniques for Binary Trees :

```
treenode = record
{
    type data ;
    treenode * lchild ;
    treenode * rchild ;
}
```

Algorithm Inorder (t) → Binary tree with height tree

```
{
 if t ≠ 0 then
 {
  Inorder (t → lchild);
  Visit (t); // Process.
  Inorder (t → rchild);
 }
}
Algorithm Preorder (t)
{
  if t ≠ 0 then
  {
    Visit (t);
    PreOrder (t → lchild);
    Preorder (t → rchild);
  }
}
Algorithm postorder (t)
{
 if t ≠ 0 then
   PostOrder (t → lchild);
   PostOrder (t → rchild);
    Visit (t);
  }
}
```

degree of node ⇒ no. of node (or) subnode.

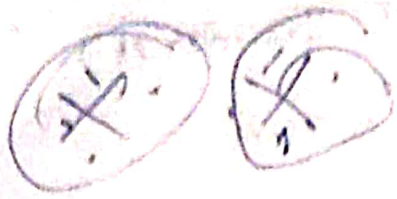
degree ≤ 2 is binary Tree.

A + B * C ↑ D - E / F

Inorder - A + B

Preorder - + A B

Postorder - A B + .



Traversal - Processing each and every node.

Searching → to check available node in Tree (or) Graph. find the location of given item.

1) A + B * C ↑ D - E / F

Possibility :

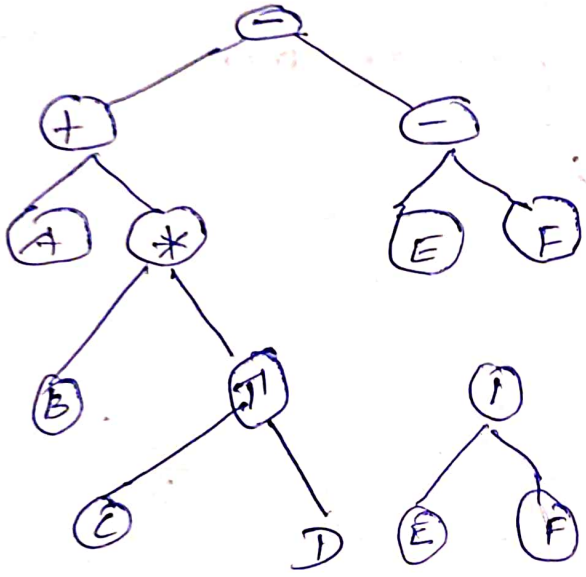
LRD

DLR

DRL - Preposition

RLD

RDL.



In-order :

⇒ (A - D) .

left data right.

B D A E C

B D A D E C I G E H.

Pre data :

~~LDR A B C AB D E D~~

A B D C E

A B D C E

Post :: D D E I G H P C A.

Result :

Pre :: + - A * B D * C D / E F
A n D B * C D . E F .

+ - A * B D C D / E F

A B + C D + * + E F / - .

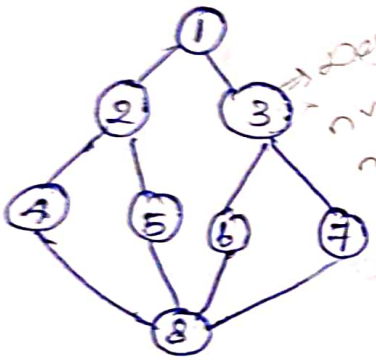
Technique for Graphs :

Algorithm for Graphs :

Algorithm BFS(V) → Queue → Data structure
{
u := v;
visited [v] := 1; → Global array (Linear array)
repeat
{
for all vertices w adjacent from u do.
{
if (visited [w] = 0) then
{
Add w to q (queue);
visited [w] := 1;
}
}
if q is empty then return ; Delete the next element
u from q ;
} until(false);
}

```
Algorithm BFT(G,n)
{
  for i := 1 to n do
    visited[i] := 0;
  for i := 1 to n do
    if (visited[i] = 0) then
      BFS(i);
}
Algorithm DFS(v)
{
  visited[v] := 1;
  for each vertex w adjacent from v do
  { if (visited[w] = 0) then
      DFS(w);
```

9.3

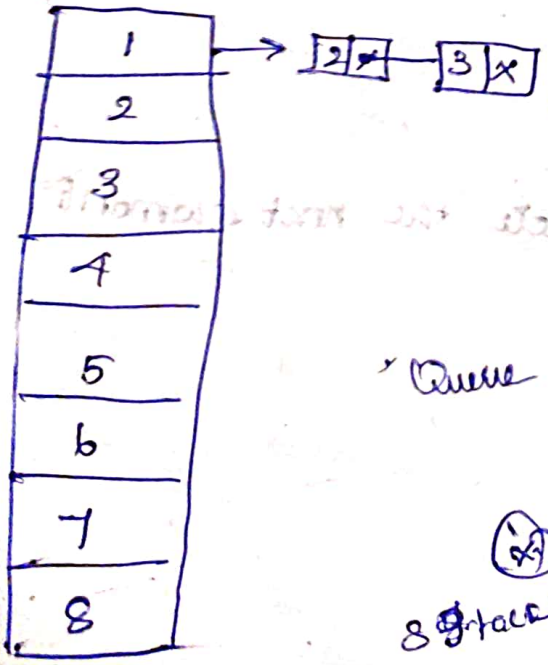


Depth first spanning tree

n vertices
n-1 edges
connected
spanning tree

Tree – has only one node way 7 the element.

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



We are only Traverse the Binary tree.

- Queue ⇒ BFS → Breath First Search
- ⇒ BFT → Breath First Traversal
- ⇒ DFS → Depth First Search
- ⇒ DFT → Depth First Traversal

v=1

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

u := 1 ✓ u := 2 u := 3

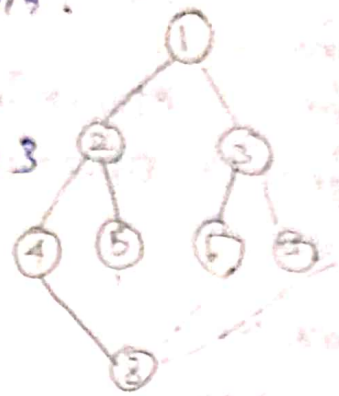
w = 2, 3 ✓ w = 1, 4, 5. w = 1, 6, 7.

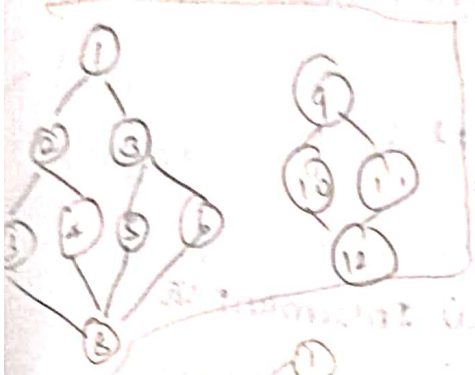
u : 4 ✓ u := 5 u : 6 u := 7

w = 2, 8 ✓ w = 2, 8 w = 3, 8 w = 7, 3

v = 8 [w = 4, 5, 6, 7]

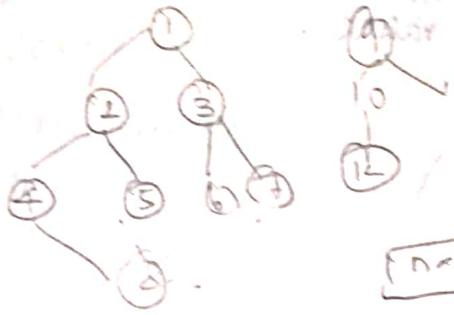
Q = 2, 3, 4, 5, 6, 7, 8





⇒ undirected graph

disconnected graph.



→ Same for Graph

n = 7 for matrix

adjancy matrix ⇒ O(n+5)

⇒ O(n²)

Depth First :- (we can use DFT & DST both are same)

	1	2	3	4	5	6	7	8
VISITED =	0	0	0	0	0	0	0	0

→ adjancy list

adjancy traversing.

v = 1 ✓ v = 2 ✓ v = 4.

w = 2, 3 w = 1, 4, 5 w = 2, 8

w = 8 ✓

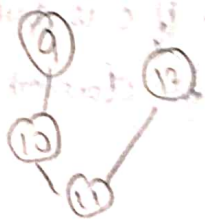
v = 4. v = 6

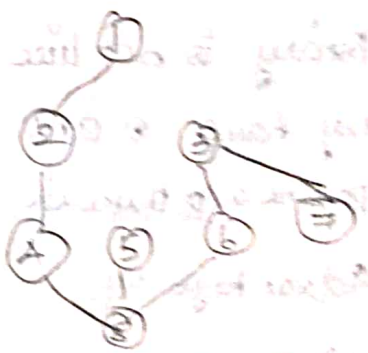
w = 4, 5, 6, 7, w = 3, 8

w = 2, 8

v = 6. v = 3 ✓ v = 7

w = 3, 7 w = 3, 8

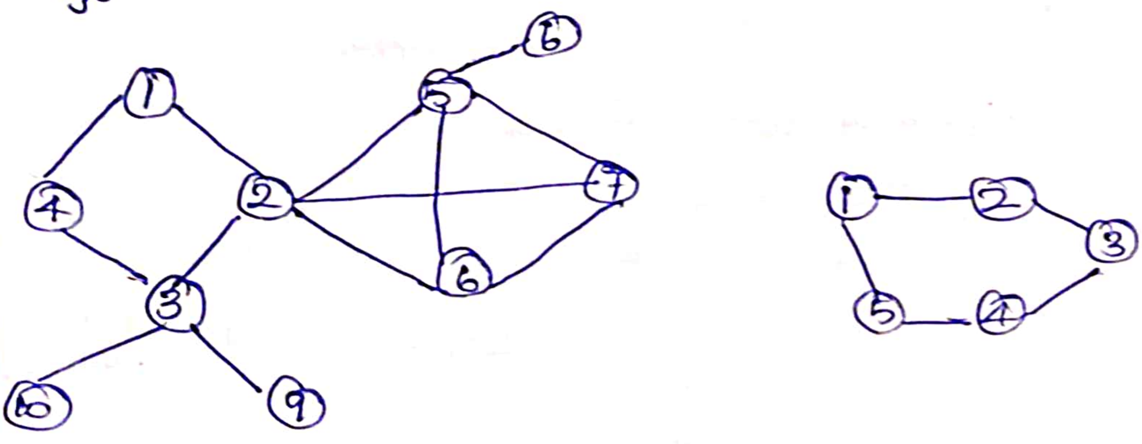




Depth first Spanning tree

Biconnected components and DFS :-

Algorithm dst (u, v) (start)
Explanation: (Depth first array.
{
    dfn[u] := num;
    L[u] := num;
    num := num + 1;
  for each vertex w adjacent from u do
  {
    if (dfn[w] = 0) then
    {
      dst (w, u);
      L[u] := min (L(u), L[w]);
    }
    else if (w ≠ v) then L[u] := min (L[u], dfn[w]).
}}



Graph G (a) (b)

$L[u] = \min \{ dfn[n], \min \{ L[w] \mid w \text{ is a child of } u \}, \min \{ dfn[w] \mid (u, w) \text{ is a back edge} \} \}$.

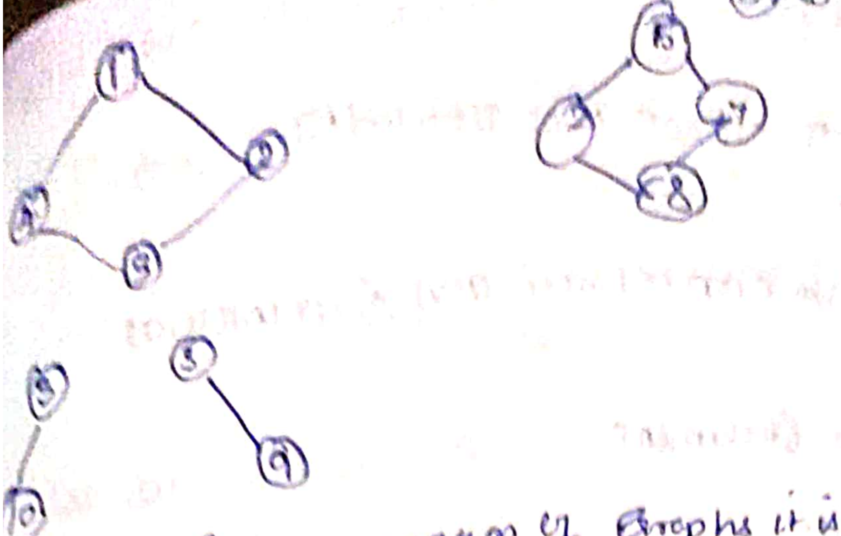
architectural point = 2, 3, 5.

A vertices v be a connected graph of G is a architectural point & together with edge.

→ optimization

A Graph containing no architecture point (optimization).

Binary contain Graph.

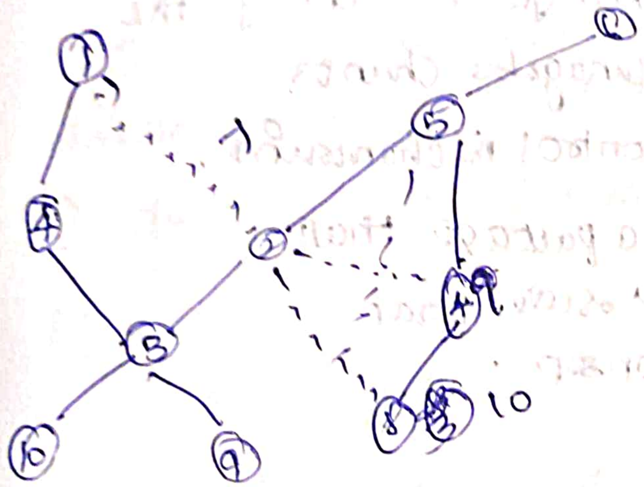


maximal connection of graphs it is strongly connected components. (Depth first search method)

scheme to connect construct. By Graph.

1) By need architecture two.

Final adjacency :-



Post order → left right data is known as Postorder.

Bipartite graph :- 248

$V_1 = \{v1, v2, v3, v4, v5, v6, v7\}$

$V_1' = \{v1, v2, v3\}$

$V_2 = \{$ ~~v1,v2,v3~~ v4, v5, v6, v7.