03-02-17 **UNIT-III : DYNAMIC PROGRAMMING**

→ Similar to Divide and Conquer Approach.
→ In Dynamic Programming, the problem is divided into sub-problems, the sub-problems are not independent (as in divide and conquer); it depends on some other sub-problems, the sub-problem is divided into sub-sub-problems.

→ Eg: 4 | 3 | 2 | 5 | 7 | 3 | 2 | 6
3 4 2 5 3 7 2 6
2 3 4 5 2 3 7 6

↳ In D and C, we are sorting 2 and 3 again in the 2nd run (independent)
↳ But in dynamic programming, we need not sort 2 and 3 again, it is already present.

Eg. Fibonacci Series

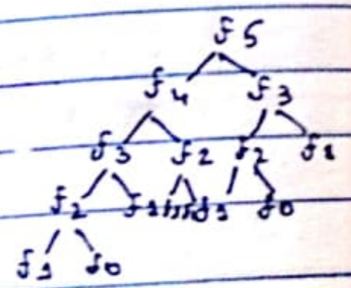
$f_n = f_{n-1} + f_{n-2}$

$f_5 = f_4 + f_3$

↳ $f_3 + f_2$ ↳ $f_2 + f_1$

↳ $f_2 + f_1$ ↳ $f_1 + f_0$ ↳ $f_1 + f_0$

[$f_1 = 1$; $f_0 = 0$]



Recursive call of $f_0 = 3$, $f_1 = 5$, $f_2 = 3$, $f_3 = 2$, $f_4 = 1$, $f_5 = 1$

↳ In Divide and Conquer Approach
But, in Dynamic Programming we can do it by saving some recursive calls

→ Dynamic Programming is a Tabular Method we construct a table we have the solutions of the sub-programs there. Whenever we need the solution of a sub-problem, we retrieve it from there instead of performing the task again.
→ Dynamic Programming is of two types:
i) Top Down Approach (Memorization)
ii) Bottom Up Approach (Tabular Method)

→ Eg. Fibonacci Series with memorization

f_0	0
f_1	1
f_2	$f_0 + f_1 = 1$
f_3	$f_1 + f_2 = 2$
f_4	$f_2 + f_3 = 3$
f_5	$f_3 + f_4 = 5$

→ Eg. Fibonacci Series with Bottom Up Approach
Construct a table (array) with size 'n'

A[0] = 0 A[1] = 1

A[i] = A[i-1] + A[i-2]

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

→ Multiple decisions in dynamic programming, not following a single input order ⇒ multiple/many decision sequences.
In greedy approach, we have only one input order and have only one decision sequence

→ If any problem contains overlapping sub-problems, we apply dynamic programming to solve these problems.
→ We must find optimality sequence for the overlapping sub-problems.

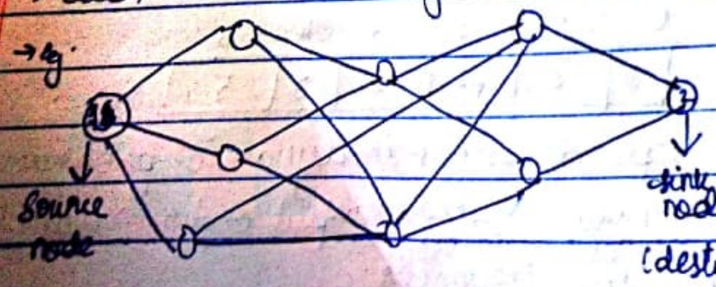
* → Principle of Optimality (Control Abstraction of Dynamic Programming)

- The principle of Optimality states that an optimal sequence of decisions has the property that whatever the initial state and decision are, the remaining decision(s) must constitute an optimal decision sequence with regard to state resulting from the first decision.

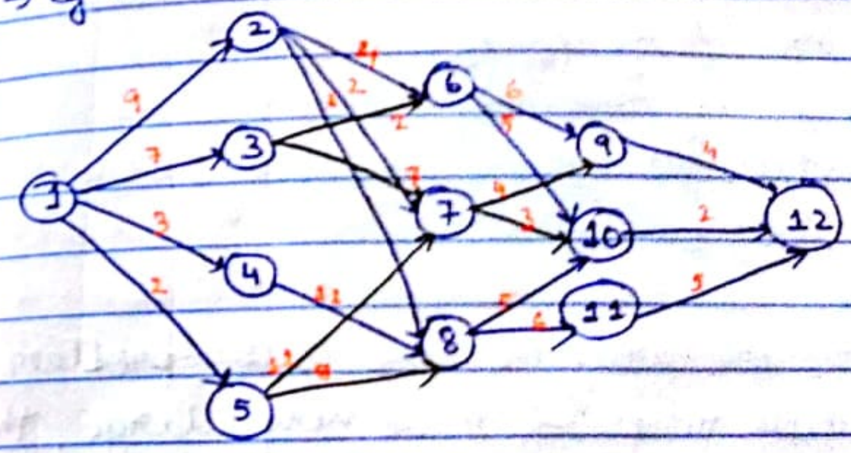
Multistage Graph

→ It is a directed graph G(V, E).
→ From the set of vertices, we must form 'k' distinct groups.
→ Each one is one stage.

→ Eg.



→ We must find the shortest path from source node to sink node.
→ We must find shortest paths from source to stage 1, stage 1 to stage 2, ... n.
→ V1, V2, V3, V4, V5, ... vertices at different stages
→ We have to find the minimum cost at each stage so that we can find the minimum cost from the source to the destination.
→ Eg. V1 V2 V3 V4 V5

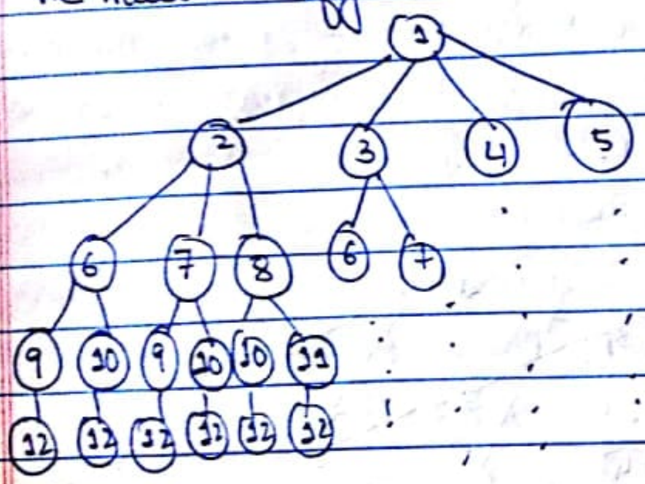


'k' stage graph
$|V_1| = |V_k| = 1$
We have to take (k-2) decisions to provide a solution for this problem
We have three approaches to provide a solution to the
↳ Greedy Approach (1, 5, 8, 10, 12 : (2+3+5+2) = 12)
↳ Brute Force Method
↳ Dynamic Programming
Once we make a decision in greedy approach, we

But without greedy approach, (select max cost after 1st edge), we get minimum cost ((1),(3),(7),(10) = 9+2+3+2 = 16)
So, greedy approach fails for the above problem.

Brute Force

We must verify each and every possibility



We are calling the same sub-problem recursively multiple times. If 'n' is very large, the time complexity is exponential time.
So, we reject this approach.

Dynamic Programming

We use backward approach. 12 - min cost = 10, 10 - min cost = 7 (Stage 5 - Stage 4 - Stage 3 - Stage 2 - Stage 1). We take (k-2) decisions for 'k' stage graph '-2=) destination and source

Cost(i,j) = Shortest path from node j in stage/level v_i to sink node 't'

c(i,j) = edge cost between node i and j
(no direct edge = ∞)

$$Cost(i,j) = \min_{l \in v_{i+1}, (j,l) \in E} \{ c(j,l) + Cost(i+1, l) \}$$

$$Cost(k-1, t) = \begin{cases} c(j,t) & \text{if } \langle j, t \rangle \in E \\ \infty & \text{if } \langle j, t \rangle \notin E \end{cases}$$

Forward Approach →

eg. Cost(3,6)

↳ 3 = level 3=i 6 = node = j

$l \in 4$

Cost(3,6) = min {Cost(5,9) + cost(4,9)}

Cost(k-1, t) = cost(5-1, 12) = cost(4,

→ Refer to previous 5 stage multi graph

k = 5 Cost(k-1, t) = c(j,t)

Cost(4,9) = 4 Cost(4,10) = 2 Cost(4,11) = 5

Cost(k+2, j)

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

= min {6+4, 5+2} = min {10, 7} = 7

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

= min {4+4, 3+2} = min {8,5} = 5

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

= min {5+2, 6+5} = min {7, 11} = 7

Cost (k-3, j)

Cost (2,2) = min {c(2,6) + cost(6,6), c(2,7) + cost(7,7), c(2,8) + cost(8,8)}

= min {11 + 7, 2 + 5, 1 + 7} = min {18, 7, 8} = 7

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

= min {2 + 7, 7 + 5} = min {9, 12} = 9

Cost (2,4) = min {c(4,8) + cost(8,8)} = min {11 + 7} = 18

Cost (2,5) = min {c(5,7) + cost(7,7), c(5,8) + cost(8,8)}

= min {11 + 5, 8 + 7} = min {16, 15} = 15

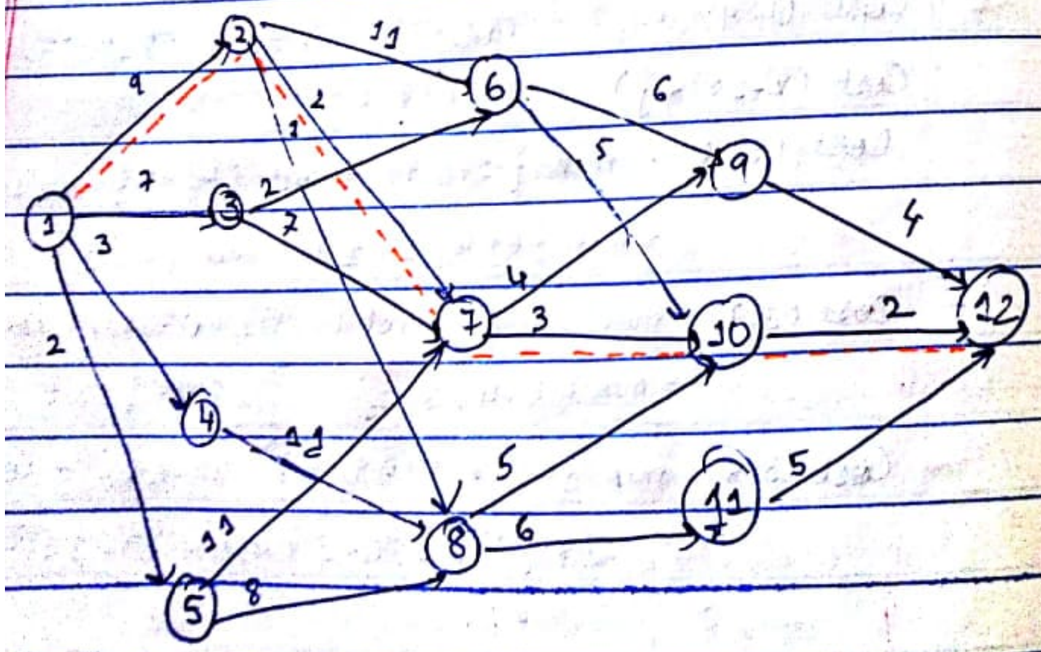
Cost (k-4, j)

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

= min {9 + 7, 7 + 9, 3 + 18, 2 + 15} = min {16, 16, 21, 17} = 16

∴ Min cost = (1,2), (2,7), (7,10), (10,12) = (9 + 2 + 3 + 2) = 16.

↳ Forward Approach



Backward Approach:

bcost (i,j) = min {bcost (i-1, l) + c(l,j)} , l ∈ Vi-1, (l,j) ∈ E
bcost (2,j) = c(1,j), if (1,j) ∈ E
= ∞, if (1,j) ∉ E

bcost (2,2) = min c(1,2) = 9

bcost (2,3) = c(1,3) = 7

bcost (2,4) = c(1,4) = 3

bcost (2,5) = c(1,5) = 2

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

= min {9 + 11, 7 + 2} = min {20, 9} = 9

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

= min {7 + 7, 2 + 11} = min {14, 13} = 13

bcost (3,8) = min {bcost (2,2) + c(2,8), bcost (2,4) + c(4,8), bcost (2,5) + c(5,8)} = min {9 + 1, 3 + 11, 2 + 8}

= min {10, 14, 10} = 10

bcost

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

= min {9 + 6, 13 + 4} = min {15, 17} = 15

bcost (4,10) = min {bcost (3,6) + c(6,10), bcost (3,7) + c(7,10), bcost (3,8) + c(8,10)} = min {9 + 5, 13 + 3, 10 + 5}

= min {14, 16, 15} = 14

bcost (4,11) = min {bcost (3,8) + cost (8,11)} = min {10 + 6} = min

= 16

~~bcost (4,11) = min {bcost (3,8) + c (8,11)} = min{~~

bcost (5,12) = min { bcost (4,9) + c(9,12), bcost (4,10) + cost(10,12)
, bcost (4,11) + cost (11,12) }
= min { 15+4, 14+2, 16+5 } = min {19, 16, 21} = 16

∴ Min Cost = 12 - 10 - 7 - 2 - 1
= (1,2), (2,7), (7,10), (10,12)
= 9 + 2 + 3 + 2 = 16

<< Backward Approach

→ Algorithm FGraph (G, k, n, p) //Forward approach
{
// If 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
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] 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]];
}

→ Algorithm BGraph (G, k, n, p) // Backward Approach
{
// 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];
d[j] := r;
}
// Find a minimum cost path
p[1] := 1; p[k] := n
for j = k - 1 to 2 do p[j] := d[p[j+1]];
}

15-02-'18

All-Pairs Shortest Path

→ Problem Definition: From the given directed graph G(V,E); construct cost matrix.

→ We construct cost adjacency matrix by:

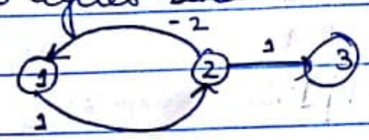
cost(i,i) = 0

cost(i,j) = length of edge, $(i,j) \in E$

cost(i,j) = ∞ , $(i,j) \notin E$, $i \neq j$

→ We assume that the directed Graph 'G' has no cycles ~~and~~ with no negative edges

Eg.



① - ③ => 1, -2, 1, -2, 1, -2 . . . . . .

-> ①, ②, ③, ②, ①, ② . . . . . .

We must remove the cycle with -ve edges then.

→ To find shortest path from node i to node j going through some intermediate node 'k' if node k exists, then the sub paths i to k and k to j are also shortest paths.

Repeath the above procedure for i to k, k to j and continue till there are no intermediate nodes or vertices

→ For this, we have to construct a matrix $A^n(i,j)$

n = no. of vertices i,j : source, destination nodes

→ To construct $A^n(i,j)$, we must construct $A^{n-1}(i,j)$

$A^0(i,j)$ = cost(i,j)

→ Then, we must construct $A^1(i,j)$

$A^1(i,j)$ = shortest path from vertex i to vertex j through intermediate vertex ①

→ We must calculate $A^k(i,j)$

$A^k(i,j)$ = shortest path from vertex i to vertex j through intermediate vertex (k)

To calculate $A^k(i,j)$, it depends on $A^{k-1}(i,k) + A^{k-1}(k,j)$ =) when (k) exists

If the intermediate vertex (k) does not exist,

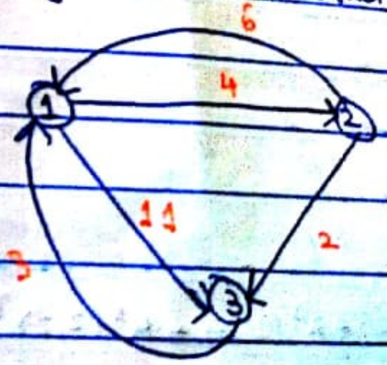
$A^k(i,j) = A^{k-1}(i,j)$

→ Combining the above two cases,

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

(k-1) => largest index, no other greater index than k-1 is used

→ Eg.



Aim: To find $A^3(i,j)$

First: Construct $A^0(i,j)$

A^0	1	2	3
1	0	4	11
2	6	0	2
3	3	∞	0

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

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

$A^1(1,3) = \min\{A^0(1,3), A^0(1,1)+A^0(1,3)\} = \min\{11, (0+11)\} = 11$

$A^1(2,1) = \min\{A^0(2,1), A^0(2,1)+A^0(1,1)\} = \min\{6, (6+0)\} = 6$

$A^1(2,2) = \min\{A^0(2,2), A^0(2,1)+A^0(1,2)\} = \min\{0, (6+4)\} = 0$

$A^1(2,3) = \min\{A^0(2,3), A^0(2,1)+A^0(1,3)\} = \min\{2, (6+11)\} = 2$

$A^1(3,1) = \min\{A^0(3,1), A^0(3,1)+A^0(1,3)\} = \min\{3, (3+11)\} = 3$

$A^1(3,2) = \min\{A^0(3,2), A^0(3,1)+A^0(1,2)\} = \min\{\infty, (3+4)\} = 7$

$A^1(3,3) = \min\{A^0(3,3), A^0(3,1)+A^0(1,3)\} = \min\{0, (3+11)\} = 0$

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

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

$A^2(1,1) = \min\{A^1(1,1), A^1(1,2)+A^1(2,1)\} = \min\{0, (4+6)\} = 0$

$A^2(1,3) = \min\{A^1(1,3), A^1(1,2)+A^1(2,3)\} = \min\{11, (4+2)\} = 6$

$A^2(2,1) = \min\{A^1(2,1), A^1(2,2)+A^1(2,1)\} = \min\{6, (0+6)\} = 6$

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

$A^2(2,3) = \min\{A^1(2,3), A^1(2,2)+A^1(2,3)\} = \min\{2, (0+2)\} = 2$

$A^2(3,1) = \min\{A^1(3,1), A^1(3,2)+A^1(2,1)\} = \min\{3, (7+6)\} = 3$

$A^2(3,2) = \min\{A^1(3,2), A^1(3,2)+A^1(2,2)\} = \min\{7, (7+0)\} = 7$

$A^2(3,3) = \min\{A^1(3,3), A^1(3,2)+A^1(2,3)\} = \min\{0, (7+2)\} = 0$

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

$A^3(1,1) = \min\{A^2(1,1), A^2(1,3)+A^2(3,1)\} = \min\{0, (6+3)\} = 0$

$A^3(1,2) = \min\{A^2(1,2), A^2(1,3)+A^2(3,2)\} = \min\{4, (6+7)\} = 4$

$A^3(1,3) = \min\{A^2(1,3), A^2(1,3)+A^2(3,3)\} = \min\{6, (6+0)\} = 6$

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

$A^3(2,2) = \min\{A^2(2,2), A^2(2,3)+A^2(3,2)\} = \min\{0, (2+7)\} = 0$

$A^3(2,3) = \min\{A^2(2,3), A^2(2,3)+A^2(3,3)\} = \min\{2, (2+0)\} = 2$

$A^3(3,1) = \min\{A^2(3,1), A^2(3,3)+A^2(1,3)\} = \min\{3, (0+6)\} = 3$

$A^3(3,2) = \min\{A^2(3,2), A^2(3,3)+A^2(3,2)\} = \min\{7, (0+7)\} = 7$

$A^3(3,3) = \min\{A^2(3,3), A^2(3,3)+A^2(3,3)\} = \min\{0, (0+0)\} = 0$

∴ Shortest Path Matrix $= A^3 =$

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

```
Algorithm AllPaths(A, cost, n)  // O(n^3)
{
    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]);
            }
        }
    }
}
```

P.T.O. →

31-01-18

Single Source Shortest Path: General Weights

→ Given directed graph may contain negative weights also.

$d[2] = 7$ $d[3] = 5$ ⇒ from 1 ⇒ 1st iteration

↳ By Greedy Approach

So, with General Weights and -ve edge cost in the graph, greedy approach fails.

→ In dynamic programming, we go for $d^{l}[u]$.

Our graph 'n' vertices, maximum no. of n-1 edges, we want to find the shortest path from node 'v' to node 'u' at most (n-1) edges.

$d^{l}[u]$ = shortest path from source vertex v to node u, at most l edges

So, we calculate $d^1[u], d^2[u], d^3[u], \cdots, d^{n-1}[u]$.

$d^k[u] = d^{k-1}[u]$ ⇒ At most k-edges, but they are not available

$d^k[u] = \min\{d^{k-1}[u], \min\{d^{k-1}[i] + cost[i,u]\}\}$ ⇒ At most k-edges, they are available

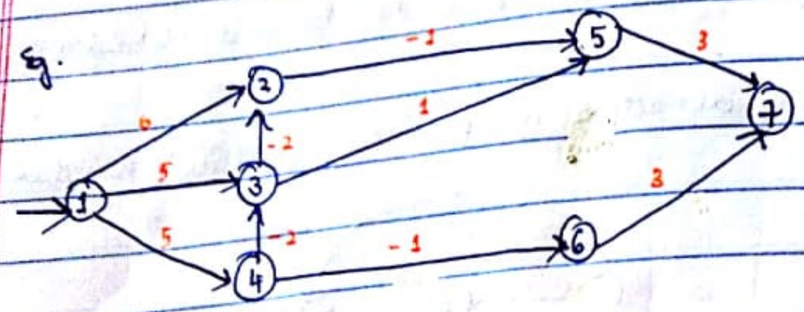
↳ Recurrence Relation

→ $d^k[u] = d^{k-1}[u]$ ⇒ Edges available

$= \min\{d^{k-1}[u], \min\{d^{k-1}[i] + cost[i,u]\}\}$ ⇒ Edge not available

→ Initial condition: $d^1[u] = cost[v,u] = cost[v,u]$

→ Eg.



$d^4[2]=6$, $d^3[3]=5$, $d^4[4]=5$, $d^4[5]=\infty$, $d^4[6]=\infty$, $d^4[7]=\infty$

$d^2[2] = \min\{d^2[2], \min_{i=1,3,4,5,6,7}\{d^2[1]+cost[1,2], d^2[3]+cost[3,2], d^2[4]+cost[4,2], d^2[5]+cost[5,2], d^2[7]+cost[7,3]\}\}$

$= \min\{6, \min\{0+6, 5-2, 5+\infty, \infty+\infty, \infty+\infty, \infty+\infty\}\}$

$= 3$

$d^3[3] = \min\{d^2[3], \min\{d^2[1]+cost[1,3], d^2[2]+cost[2,3], d^2[4]+cost[4,3], d^2[5]+cost[5,3], d^2[6]+cost[6,3], d^2[7]+cost[7,3]\}\}$

$= \min\{5, 0+5, 6+\infty, 5-2, \infty+\infty, \infty+\infty, \infty+\infty\} = 3$

edges ↓	vertices → 1	2	3	4	5	6	7
	0	6	5	5	∞	∞	∞
1	0	3	3	5	5	4	∞
2	0	3	3	5	2	4	7
3	0	1	3	5	0	4	5
4	0	1	3	5	0	4	3
5	0	1	3	5	0	4	3
6							

0/1 Knapsack (Subset Problem)

Knapsack

Fractional Knapsack | 0/1 Knapsack

Dynamic

$0 \leq x_i \leq 1$ | 1, 0, 1, 0, 1

Greedy

1, 1, 1, $\frac{5}{6}$, 0, 0, 0

$\frac{P_i}{w_i} > \frac{P_{i+1}}{w_{i+1}}$

$i_1, i_2, i_3, \ldots\ldots\ldots, i_n$ → items

$w_1, w_2, w_3, \ldots\ldots\ldots, w_n$ → weights

$P_1, P_2, P_3, \ldots\ldots\ldots, P_n$ → profits

$[(n+1) * (w+1)]$ → matrix

Base Conditions:

→ $B(0, w) = 0$; for changing w.r.t

→ $B(i, 0) = 0$; for changing i.r.t

→ i - current item, w_i = weight of current item, w = max weight allowed

$B(i, w) = B(i-1, w)$, if $(w_i > w)$

$B(i, w) = \max \begin{cases} B(i-1, w) \\ P_i + B(i-1, w-w_i) \end{cases}$

	0	1	2	3	...	...	W
0	0	0	0	0	..	..	0
1	0						
2	0						
3	0						
⋮	⋮						
⋮	⋮						
n	0						

→ weights

↓ items

Eg. W = 5

Wt	Profit
1	7
3	4
4	2

⇒ [(n+1) × (W+1)] = (3+1) × (5+1) = (4×6) matrix

	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	7	7	7	7	7
2	0	7	7	7	11	11
3	0	7	7	7	11	11

B(1,1) ⇒ $w_i > w$ ⇒ $1 > w$ ✗ (w = present max allowed)

$B(1,1) = \max\{B(1-1,1), 7 + B(1-1,1-1)\}$

$= \max\{0, 7+0\} = 7$

B(1,2) ⇒ $w_i > w$ ⇒ $1 > 2$ ✗ (restricted to 1 weight)

$B(1,2) = \max\{B(1-1,2), 7 + B(1-1,2-1)\}$

$= \max\{0, 7+0\} = 7$

B(1,3) ⇒ $w_i > w$ ⇒ $1 > 3$ ✗

$B(1,3) = \max\{B(1-1,3), 7 + B(1-1,3-1)\} = \max\{0, 7+0\} = 7$

B(1,4) ⇒ $w_i > w$ ⇒ $1 > 4$ ✗

$B(1,4) = \max\{B(1-1,4), 7 + B(1-1,4-1)\} = \max\{0, 7+0\} = 7$

B(1,5) ⇒ $w_i > w$ ⇒ $1 > 5$ ✗

$B(1,5) = \max\{B(1-1,5), 7 + B(1-1,5-1)\} = \max\{0, 7+0\} = 7$

B(2,1) ⇒ $w_i > w$ ⇒ $3 > 1$ ✓ ⇒ $B(i,w) = B(2-1,1) = B(1,1) = 7$

B(2,2) ⇒ $w_i > w$ ⇒ $3 > 2$ ⇒ $B(i,w) = B(2-1,2) = B(1,2) = 7$

B(2,3) ⇒ $w_i > w$ ⇒ $3 > 3$ ✗

$B(2,3) = \max\{B(2-1,3), 4 + B(2-1,3-3)\} = \max\{7, 4+0\} = 7$

B(2,4) ⇒ $w_i > w$ ⇒ ~~3 > 4~~ ~~4 > 3~~ $3 > 4$ ✗

$B(2,4) = \max\{B(2-1,4), 4 + B(2-1,4-3)\} = \max\{7, 4+7\} = 11$

B(2,5) ⇒ $w_i > w$ ⇒ $5 > 3$ ✗

$B(2,5) = \max\{B(2-1,5), 4 + B(2-1,5-3)\} = \max\{7, 4+7\} = 11$

B(3,1) ⇒ $w_i > w$ ⇒ $4 > 1$ ⇒ $B(i,w) = B(3-1,1) = B(2,1) = 7$

B(3,2) ⇒ $w_i > w$ ⇒ $4 > 2$ ⇒ $B(i,w) = B(3-1,2) = B(2,2) = 7$

B(3,3) ⇒ $w_i > w$ ⇒ $4 > 3$ ⇒ $B(i,w) = B(3-1,3) = B(2,3) = 7$

B(3,4) ⇒ $w_i > w$ ⇒ $4 > 4$ ✗

$B(3,4) = \max\{B(3-1,4), 2 + B(3-1,4-4)\} = \max\{11, 2+0\} = 11$

B(3,5) ⇒ $w > w$ ⇒ $4 > 5$ ✗

$B(3,5) = \max\{B(3-1,5), 2 + B(3-1,5-4)\} = \max\{11, 2+7\} = 11$

→ Algorithm to find out the items being considered into knapsack

for each row i from N to 0

if (B(i,w) - B(i-1, w - w[i] = P[i])

then;

{

item i is taken into knapsack

w:= w - w[i];

}

Tracing the algorithm for given example:

→ i = 3

B(3,5) - B(2,1) : 11 - 7 = 4 ≠ P[3] ≠ 2.

move to next step

→ i = 2

P(2,5) - B(1, 2) : 11 - 7 = 4 = P[2] = 4

item 2 is taken into knapsack

w:= 5 - 3 = 2

→ i = 1

P(1,2) - B(1-1, 2-1) = 7 - 0 = 7 = P[1] = 7

item 1 is taken into knapsack

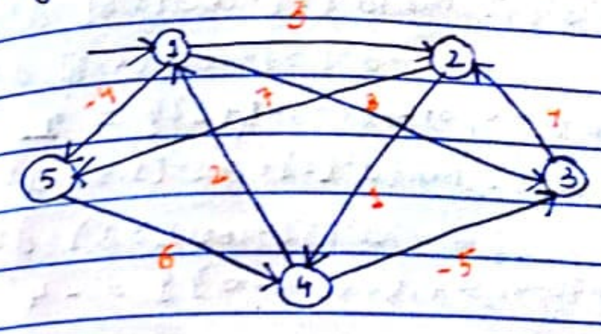
w:= 2 - 1 = 1

→ i = 0

Stop

Knapsack contains items 2 and 1

∴ Find the shortest path from node 1 to remaining vertices by applying Single Source Shortest Path (General weights) Algorithm:



$d^k[u] = \min \{ d^{k-a}[u], \min \{ d^{k-1}[i] + cost[i,u] \} \}$

- $d^1[2] = 3$ $d^1[3] = 8$ $d^1[4] = \infty$ $d^1[5] = -4$
- $d^1[1] = 0$
- $d^2[2] = \min \{ d^2[2], \min_{i=1,3,4,5} \{ d^1[1] + cost[1,2], d^1[$

$= \min \{ d^1[2], \min \{ d^1[1] + cost[1,2], d^1[3] + cost[3,2], d^1[4] + cost[4,2], d^1[5] + cost[5,2] \} \}$

$= \min \{ 3, \min \{ 0+3, 8+4, 2+\infty, -4+\infty \} \}$

$\{ 3, \min \{ 3, 12, \infty, \infty \}$

$= 3$

- $d^2[3] = \min\{d^1[3], \min\{d^1[1] + cost[1,3]; d^1[2] + cost[2,3]; d^1[4] + cost[4,3]; d^1[5] + cost[5,3]\}\}$
  $= \min\{8, \min\{0+8, 3+\infty, \infty+(-5), -4+\infty\}\} = 8$
- $d^2[4] = \min\{d^1[4], \min\{d^1[1] + cost[1,4]; d^1[2] + cost[2,4]; d^1[3] + cost[3,4]; d^1[5] + cost[5,4]\}\}$
  $= \min\{\infty, \min\{0+\infty, 3+1, 8+\infty, -4+6\}\} = 2$
- $d^2[5] = \min\{d^1[5], \min\{d^1[1] + cost[1,5]; d^1[2] + cost[2,5]; d^1[3] + cost[3,5]; d^1[4] + cost[4,5]\}\}$
  $= \min\{-4, \min\{0-4, 3+7, 8+\infty, \infty+\infty\}\} = -4$
- $d^3[1] = 0$
- $d^3[2] = \min\{d^2[2], \min\{d^2[1] + \overset{cost}{\min}[1,2]; d^2[3] + \overset{cost}{\min}[3,2]; d^2[4] + \overset{cost}{\min}[4,2]; d^2[5] + \overset{cost}{\min}[5,2]\}\}$
  $= \min\{3, \min\{0+3, 8+4, 1+\infty, -4+\infty\}\} = 3$
- $d^3[3] = \min\{d^2[3], \min\{d^2[1] + \overset{cost}{\min}[1,3]; d^2[2] + \overset{cost}{\min}[2,3]; d^2[4] + \overset{cost}{\min}[4,3]; d^2[5] + \overset{cost}{\min}[5,3]\}\}$
  $= \min\{8, \min\{0+8, 3+\infty, 2+(-5), -4+\infty\}\} = -3$
- $d^3[4] = \min\{d^2[4], \min\{d^2[1] + cost[1,4]; d^2[2] + cost[2,4]; d^2[3] + cost[3,4]; d^2[5] + cost[5,4]\}\}$
  $= \min\{2, \min\{0+\infty, 3+1, 8+\infty, -4+6\}\} = 2$
- $d^3[5] = \min\{d^2[5], \min\{d^2[1] + cost[1,5]; d^2[2] + cost[2,5]; d^2[3] + cost[3,5]; d^2[4] + cost[4,5]\}\}$
  $= \min\{-4, \min\{0-4, 3+7, 8+\infty, 2+\infty\}\} = -4$

$d^4[1] = 0$

$d^4[2] = \min\{d^3[2], \min\{d^3[1] + cost[1,2]; d^3[3] + cost[3,2]; d^3[4] + cost[4,2]; d^3[5] + cost[5,2]\}\}$
$= \min\{3, \min\{0+3, -3+4, 2+\infty, -3+\infty\}\} = 1$

$d^4[3] = \min\{d^3[3], \min\{d^3[1] + cost[1,3]; d^3[2] + cost[2,3]; d^3[4] + cost[4,3]; d^3[5] + cost[5,3]\}\}$
$= \min\{-3, \min\{0+8, 3+\infty, 2+(-5), -4+\infty\}\} = -3$

$d^4[4] = \min\{d^3[4], \min\{d^3[1] + cost[1,4]; d^3[2] + cost[2,4]; d^3[3] + cost[3,4]; d^3[5] + cost[5,4]\}\}$
$= \min\{2, \min\{0+\infty, 3+1, -3+\infty, -4+6\}\} = 2$

$d^4[5] = \min\{d^3[5], \min\{d^3[1] + cost[1,5]; d^3[2] + cost[2,5]; d^3[3] + cost[3,5]; d^3[4] + cost[4,5]\}\}$
$= \min\{-4, \min\{0-4, 3+7, -3+\infty, 2+\infty\}\} = -4$

vertices → / edges ↓	1	2	3	4	5
1	0	3	8	∞	-4
2	0	3	8	2	-4
3	0	3	-3	2	-4
4	0	1	-3	2	-4

3

0/1 knapsack, W = 5

v_i	P_i
2	3
3	4
4	5
5	6

→ $B(i,w) = \max \{ B(i-1,w);\ P_i + B(i-1, w-w_i) \}$

26-02-'18

Travelling Salesman Problem (TVSP) (Permutation Problem)

→ Salesman must start at one city, covers all other cities in the path and reach the same city again in a path of minimum length: Problem Statement.

→ G(V, E)

$c_{ij} > 0$ $c_{ij} = \infty$ => no path between i and j

→ Optimal Solution: $g(1, V-1)$

We must find $g(i, S)$

=> Salesman starts at vertex i, covers all the cities in set S and reaches back to vertex i.

→ $g(1, V-\{1,k\}) = \min\{c_{1k} + g(k, V-\{1,k\})\}$

↳ Recurrence Relation

$g(i,S) = \min\{c_{ij} + g(i, S-\{j\})\}$

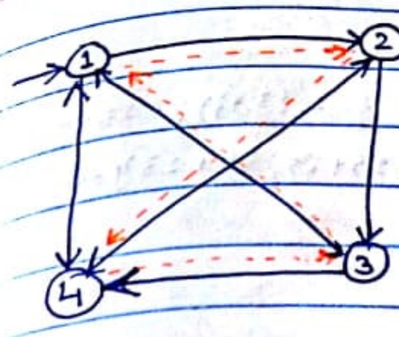
→ Initially, $S = \phi, S = 1, S = 2, \ldots\ldots, S = n-1$

$S = \phi$ → empty set

↳ Salesman starts at vertex (city) i, no cities in Set S, so he reaches vertex (city) i again

Base Case: $g(i, \phi) = c_{i1}$

→



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

• $|S| = 0$

$g(2, \phi) = c_{i1} = c_{21} = 5$

$g(3, \phi) = c_{i1} = c_{31} = 6$

$g(4, \phi) = c_{i1} = c_{41} = 8$

• $|S| = 1$

$g(2, \{3\}) = \min\{c_{23} + g(3, \phi)\} = \min\{9+6\} = 15$

$g(2, \{4\}) = \min\{c_{24} + g(4, \phi)\} = \min\{10+8\} = 18$

$g(3, \{2\}) = \min\{c_{32} + g(2, \phi)\} = \min\{13+5\} = 18$

$g(3, \{4\}) = \min\{c_{34} + g(4, \phi)\} = \min\{12+8\} = 20$

$g(4, \{2\}) = \min\{c_{42} + g(2, \phi)\} = \min\{8+5\} = 13$

$g(4, \{3\}) = \min\{c_{43} + g(3, \phi)\} = \min\{9+6\} = 15$

• $|S| = 2$

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

$= \min\{9+20, 10+15\} = \min\{29, 25\} = 25$

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

$= \min\{13+18, 12+13\} = \min\{31, 25\} = 25$

$g(4, \{2,3\}) = \min\{c_{42} + g(2, \{3\}), c_{43} + g(3, \{2\})\}$

$= \min\{8+15, 9+18\} = \min\{23, 27\} = 23$

• |S| = 3

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

$= \min\{10+25, 15+25, 20+23\} = \min\{35, 40, 43\}$

$= 35$

$c_{12} + g(2,\{3,4\})$

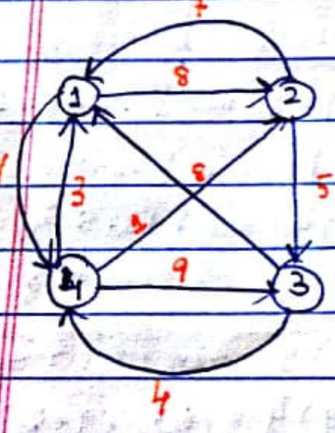
↳ $c_{24} + g(4,\{3\})$

↳ $c_{43} + g(3,\phi)$

∴ Path = 1 → 2 → 4 → 3 → 1 //By Backtracking

↳ Path length = 35 = minimum

Eg.



=>

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

• |S| = 1 |S| = 0

$g(2,\phi) = c_{i1} = c_{21} = 7$

$g(3,\phi) = c_{i1} = c_{31} = 8$

$g(4,\phi) = c_{i1} = c_{41} = 3$

• |S| = 2 |S| = 1

$g(2,\{3\}) = \min\{c_{23} + g(3,\phi)\} = \min\{5+8\} = 13$

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

$g(3,\{2\}) = \min\{c_{32} + g(2,\phi)\} = \min\{\infty+7\} = \infty$

$g(3,\{4\}) = \min\{c_{34} + g(4,\phi)\} = \min\{4+3\} = 7$

$g(4,\{2\}) = \min\{c_{42} + g(2,\phi)\} = \min\{1+7\} = 8$

$g(4,\{3\}) = \min\{c_{43} + g(3,\phi)\} = \min\{9+8\} = 17$

• |S| = 3 |S| = 2

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

$= \min\{5+7, \infty+17\} = \min\{12, \infty\} = 12$

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

$= \min\{\infty+\infty, 4+8\} = \min\{\infty, 12\} = 12$

$g(4,\{2,3\}) = \min\{c_{42} + g(2,\{3\}), c_{43} + g(3,\{2\})\}$

$= \min\{1+13, 9+\infty\} = \min\{14, \infty\} = 14$

• |S| = 1 |S| = 3

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

$= \min\{8+12, \infty+12, 2+14\} = \min\{20, \infty, 16\} = 16$

∴ 1 → 4 → 2 → 3 → 1 = Path, Path length = 16 = minimum

$c_{14} + g(4,\{2,3\})$

↳ $g(c_{42} + g(2,\{3\})$ => 1 → 4 → 2 → 3 → 1

↳ $c_{23} + g(3,\phi)$ //By Backtracking

02-03-'18

Reliability Design

→ **Problem Statement** : Design a system using several devices connected in serial order, to increase reliability of the system.

→ $D_1, D_2, D_3 \ldots\ldots, D_n$ are 'n' no. of devices connected in serial order. Each device has some reliability $r_1, r_2, r_3, \ldots, r_n$

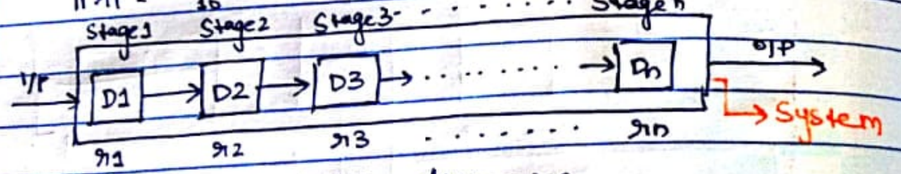
→ We give input to the device and we get output.

→ Total probability of the device is: $\prod_{1 \le i \le n} r_i$

→ Eg. Each device probability of reliability $r_i = 0.99$, $n = 10$

$\prod r_i = \prod_{10} 0.99 = (0.99)^{10} = 0.9044$

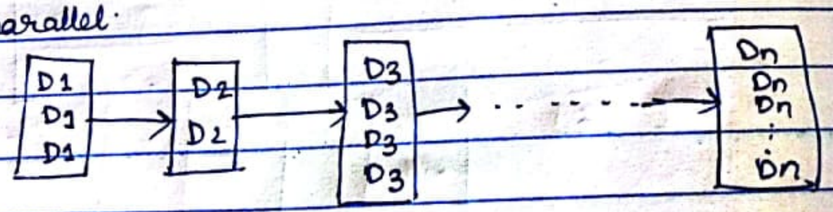
→



→ The collective reliability decreases.

To increase the probability of the system, we must take multiple copies of the same devices connected in parallel.

→ Eg.



→ We must find out how many devices we must take in each stage to increase reliability.

→ Each device has some cost c_i and reliability r_i

→ We have a budget C.

→ We must create a device with a constraint such that we must maximize the cost of the system within the budget C.

→ The formula for it is: $\sum_{i=1}^{n} c_i m_i \le C$

c_i = cost of device i, m_i = multiple copies of each device i;

r_i = reliability of device i, $1 - r_i$ = malfunction of device i;

$(1 - r_i)^{m_i}$ = malfunction of stage i;

$1 - (1 - r_i)^{m_i}$ = reliability of stage i;

→ We have to maximise $\phi_i(m_i)$

→ no. of multiple copies of same device for stage i

$\max \phi_i(m_i) = 1 - (1 - r_i)^{m_i} \quad [1 \le i \le n]$

$m_i \ge 1 \Rightarrow$ always

→ To calculate number of devices considered for each stage:

$u_i = \left\lfloor \dfrac{c - \sum_{j=1}^{n} c_j + c_i}{c_i} \right\rfloor$ ⇒ Range of devices (max) for each stage

→ We must maximise reliability of the total system, ϕ_i in this problem.

→ Suppose m_n devices are considered for stage 'n', our cost is: $c - c_n m_n$

→ We must multiply how many devices we consider in previous stage to it: $f_{n-1}(c - c_n m_n)$

→ Equation:

$$f_n(c) = \max \{ \phi(m_n) f_{n-1}(c - c_n m_n) \}$$
$$1 \le m_n \le u_n$$

↳ Recurrence Relation / Equation for the problem.

→ $f_0(x) = 1, \quad x = -ve \Rightarrow f_0(x) = -\infty$

Eg.

Device	Cost	r_i
D_1	30	0.9
D_2	15	0.8
D_3	20	0.5

$C = 105$

→ 3 devices connected in serial order, reliability of total system $= 0.9 \times 0.8 \times 0.5 = 0.36$

$$u_1 = \left\lfloor \frac{(C + C_1 - \sum_{j=1}^{n} c_j)}{c_1} \right\rfloor = \left\lfloor \frac{105 + 30 - (30+15+20)}{30} \right\rfloor$$

$$= \left\lfloor \frac{135 - 65}{30} \right\rfloor = \left\lfloor \frac{70}{30} \right\rfloor = \left\lfloor \frac{7}{3} \right\rfloor = \lfloor 2.33 \rfloor = 2$$

$$u_2 = \left\lfloor \frac{(C + C_2 - \sum_{j=1}^{n} c_j)}{c_2} \right\rfloor = \left\lfloor \frac{105 + 15 - (30+15+20)}{15} \right\rfloor$$

$$= \left\lfloor \frac{120 - 65}{15} \right\rfloor = \left\lfloor \frac{55}{15} \right\rfloor = \left\lfloor \frac{11}{3} \right\rfloor = \lfloor 3.667 \rfloor = 3$$

$$u_3 = \left\lfloor \frac{(C + C_2 - \sum_{j=1}^{n} c_j)}{c_2} \right\rfloor = \left\lfloor \frac{105 + 20 - (30+15+20)}{20} \right\rfloor$$

$$= \left\lfloor \frac{125 - 65}{20} \right\rfloor = \left\lfloor \frac{60}{20} \right\rfloor = \left\lfloor \frac{6}{2} \right\rfloor = \lfloor 3.00 \rfloor = 3$$

Consider maximum 2 devices for stage 1

3	2
3	3

Stg 3 ⇒ 2 devices

$\phi_i(m_i)$ ⇒ m_i no. of devices for stage i

$\phi_1(1) = 1 - (1 - r_i)^{m_i} = 1 - (1-0.9)^1 = 0.9$

$\phi_1(2) = 1 - (1 - r_i)^{m_i} = 1 - (1-0.9)^2 = 0.99$

$\phi_2(1) = 1 - (1 - r_i)^{m_i} = 1 - (1-0.8)^1 = 0.8$

$\phi_2(2) = 1 - (1 - r_i)^{m_i} = 1 - (1-0.8)^2 = 0.96$

$\phi_2(3) = 1 - (1 - r_i)^{m_i} = 1 - (1-0.8)^3 = 0.992$

$\phi_3(1) = 1 - (1 - r_i)^{m_i} = 1 - (1-0.5)^0 = 0.5$

$\phi_3(2) = 1 - (1 - r_i)^{m_i} = 1 - (1-0.5)^2 = 0.75$

$\phi_3(3) = 1 - (1 - r_i)^{m_i} = 1 - (1-0.5)^3 = 0.875$

$f_n(c)$ ⇒ n = no. of stages c = max cost

↳ $= \max_{(1 \le m_n \le u_n)} \{\phi_n(m_n) f_{n-1}(c - c_n m_n)\}$

$f_3(105) = \max \{\phi_3(m_1) f_2(c - c_3 m_1), \phi_3(m_2) f_2(c - c_3 m_2), \phi_3(m_3) f_2(c - c_3 m_3)\}$

$= \max \{\phi_3(1) f_2(105 - 20 \times 1), \phi_3(2) f_2(105 - 20 \times 2), \phi_3(3) f_2(105 - 20 \times 3)\}$

$= \max \{(0.5 \times f_2(85)), (0.75 \times f_2(65)), (0.875 \times f_2(45))\}$

0.8925 0.864 0.772

$m_1 = 2$

→

$f_2(85) = \max \{\phi_2(m_3)\, f_3(c - c_2 m_3), \phi_2(m_2) f_3(c - c_2 m_2), \phi_2(m_3) f_3(c - c_2 m_3)\}$

$= \max \{\phi_2(1)\, f_3(85 - 15\times1), \phi_2(2) f_3(85 - 15\times2), \phi_2(3) f_3(85 - 15\times3)\}$

$= \max \{(0.8 \times f_3(70)), (0.96 \times f_3(55)), (0.992 \times f_3(40))\}$

$=$ max

$=$

$f_2(65) = \max \{\phi_2(m_3) f_3(c - c_2 m_3), \phi_2(m_2) f_3(c - c_2 m_2), \phi_2(m_3)(c - c_2 m_3)\}$

$= \max \{\phi_2(5) f_3(65 - 15\times1), \phi_2(2) f_3(65 - 15\times2), \phi_2(3) f_3(65 - 15\times3)\}$

$= \max \{(0.8 \times f_1(50)), (0.96 \times f_1(35)), (0.992 \times f_1(20))\}$

$=$ max $\{$

$=$

$f_2(45) = \max \{\phi_2(m_3) f_3(c - c_2 m_3), \phi_2(m_2) f_3(c - c_2 m_2), \phi_2(m_3)(c - c_2 m_3)\}$

$= \max \{\phi_2(1) f_1(45 - 15\times1), \phi_2(2) f_1(45 - 15\times2), \phi_2(3)(45 - 15\times3)\}$

$= \max \{(0.8 \times f_2(30)), (0.96 \times f_1(15)), (0.992 \times f_1(0))\}$

$=$ max

$= 1$

$f_1(70) = \max \{\phi_1(m_1) f_0(c - c_1 m_1), \phi_2(m_2) f_0(c - c_2 m_2)\}$

$= \max \{(0.9 \times f_0$

$= 0.99$

05-03-'18

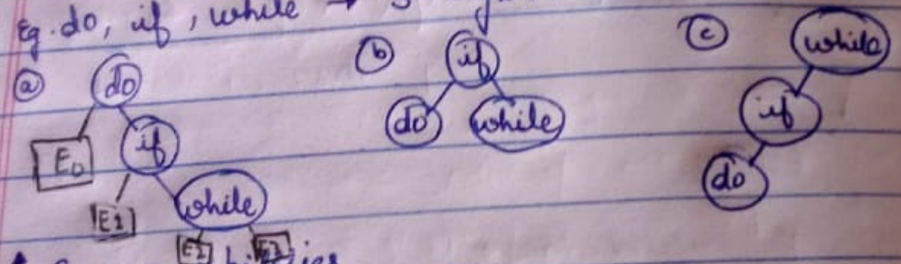
Optimal Binary Search Tree (OBST)

→ Problem Statement: For given set of identifiers we construct BST, that BST takes minimum number of search operations to find any ~~element~~ identifier.

→ $a_1, a_2, a_3, \ldots\ldots; a_n$ → Set of identifiers

$a_1 < a_2 < a_3 < \ldots\ldots < a_n$

→ Eg. do, if, while → 3 keywords



↑ Some possibilities

To search 'int' → unsuccessful search

$E_0 < do$

$do < E_1 < if$

$E_2 < while < E_3$

} Set of unsuccessful searches (external nodes for (a)

BST contains 'n' internal nodes, (n+1) external nodes

Each identifier has some probability

Total Cost of BST = $\sum_{i=1}^{n} P_i * (d_i + 1)$

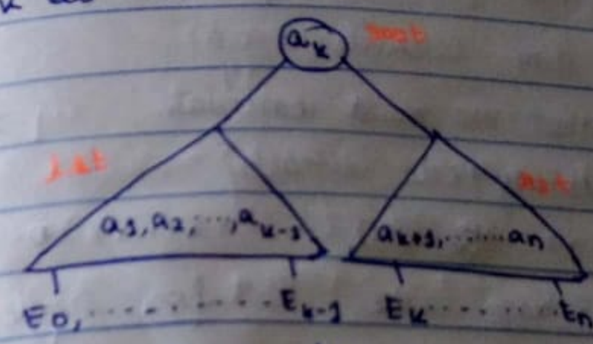
P_i = Probability of identifier, d_i = depth of identifier

Probability = $p_i + q_i = 1$

(Succesful search + unsuccesful search)

→ $a_1, a_2, a_3, \ldots\ldots, a_{k-1}, a_k, a_{k+1}, \ldots\ldots a_n$

Select a_k as root node



Construct BST, ~~for~~ cost of (0,n) = cost of lst + cost rst + P_k

Cost (0, n) = cost (lst) + cost (rst) + P_k + $w^{(lst)}_{(0,k-1)}$ + $q w^{(rst)}_{(k+1,n)}$

Choose one element as root in lst, then make lst, rst of it

Recurrence Equation:

$$c(0,n) = \text{Cost}(lst) + \text{Cost}(rst) + P_k + w(0, k-1) + w(k+1, n)$$

$$\Rightarrow c(0,n) = c(0, k-1) + c(k+1, n) + w(0, k-1) + w(k+1, n) + P_k$$

$$w(i,j) = q(i) + \sum_{l=i+1}^{j} (P(l) + q(l))$$

w(0,3) → k = 1, 2, 3

General Equation

$$c(i,j) = c(i, k-1) + c(k+1, j) + P_k + w(i, k-1) + w(k+1, j)$$

$w(i,j)$

$$c(i,j) = c(i, k-1) + c(k+1, j) + w(i,j)$$

$$c(i,j) = \min_{i < k \le j} \{c(i, k-1) + c(k, j)\} + w(i,j)$$

↳ Final equation to find BST

Every iteration, we must find out which 'k' value

We must calculate three terms everytime:
$c(i,j)$, $w(i,j)$, $r(i,j)$

Our aim : calculate $c(0,n)$

For that, we must calculate $i-j=1, i-j=2, i-j=3, \ldots, i-j=n$

0 = dummy node initially

$w(i,i) = q(i)$ $\quad c(i,i) = 0$ $\quad r(i,i) = 0$

↳ 3 base cases BST for single node

$\boxed{w(i,j) = p(j) + q(j) + w(i,j-1)}$

i node 'i', node 'j'

→ Eg. $n=4$

$(a_1, a_2, a_3, a_4) =$ (do, if, int, while)

$p(1:4) = (3,3,1,1)$ → probabilities of identifiers

$q(0:4) = (2,3,1,1,1)$ → probabilities of unsuccessful searches

↳ All probabilities are of the form 1/16

3/16, 1/16, 2/16. We multiply ×16 for our simplicity

Construct a table

	0	1	2	3	4
0	$w_{00}=2$ $c_{00}=0$ $r_{00}=0$				
1					
2					
3					
4					

→ identifiers

↓ difference b/w i and j (i-j)

	0	1	2	3	4
0	$w_{00}=2$ $c_{00}=0$ $r_{00}=0$	$w_{11}=3$ $c_{11}=0$ $r_{11}=0$	$w_{22}=1$ $c_{22}=0$ $r_{22}=0$	$w_{33}=1$ $c_{33}=0$ $r_{33}=0$	$w_{44}=1$ $c_{44}=0$ $r_{44}=0$
1	$w_{01}=8$ $c_{01}=8$ $r_{01}=1$	$w_{12}=7$ $c_{12}=7$ $r_{12}=2$	$w_{23}=3$ $c_{23}=3$ $r_{23}=3$	$w_{34}=3$ $c_{34}=3$ $r_{34}=4$	
2	$w_{02}=12$ $c_{02}=19$ $r_{02}=1$	$w_{13}=9$ $c_{13}=14$ $r_{13}=2$	$w_{24}=5$ $c_{24}=8$ $r_{24}=3$		
3	$w_{03}=14$ $c_{03}=25$ $r_{03}=02$	$w_{14}=11$ $c_{14}=19$ $r_{14}=4$			
4	$w_{04}=16$ $c_{04}=32$ $r_{04}=2$				

$w_{01} = p(1) + q(1) + w(0,0) = 3+3+2 = 8$

$c_{01} = \min\{c(0,0) + c(1,1)\} + w_{01} = \min\{0,0\} + 8 = 8$

[node 0, node 1 are available; 0 is for unsuccessful search; so consider k=1 as root node]

$r_{01} = 1$ [r = which k value gives minimum BST = 1]

$w_{12} = p(2) + q(2) + w(1,1) = 3+1+3 = 7$

$c_{12} = \min\{c(1,0) + c(2,2), c(1,1) + c(2,2)\} + w(1,2)$

$= \min\{0+0, 0+7\} + 7 = 0+7 = 7$ $\quad r_{12} = 2$

$w_{23} = p(3) + q(3) + w(2,2) = 1+1+1 = 3$

$c_{23} = \min\{c(2,2) + c(3,3)\}$ ~~$\{c(2,1) + c(3,3)\}$~~ $+ w(2,3)$

$= \min\{0+0\}$ ~~$0+5$~~ $+ 5 = 0+5 = 5$

$r_{23} = 3$

$c(i, k-1) + c(k, j)$

$w_{34} = P(4) + q(4) + w(3,3) = 1+1+1 = 3$

$c_{34} = \min\{c(3,3) + c(4,4)\} + w(3,4) = \min\{0,0\} + 3 = 3$

$r_{34} = 4$

$w_{02} = P(2) + q(2) + w(0,1) = 3+1+8 = 12$

$c_{02} = \min\{c(0,0) + c(1,2), c(0,1) + c(2,2)\} + w(0,2)$

$= \min\{0+7, 8+0\} + 12 = 7+12 = 19$

$r_{02} = 1$

$w_{13} = P(3) + q(3) + w(1,2) = 3+3+7 = 9$

$c_{13} = \min\{c(1,1) + c(2,3), c(1,2) + c(3,3)\} + w(1,3)$

$= \min\{0+5, 7+0\} + 9 = 5+9 = 14$

$r_{13} = 2$

$w_{24} = P(4) + q(4) + w(2,3) = 1+1+3 = 5$

$c_{24} = \min\{c(2,2) + c(3,4), c(2,3) + c(4,4)\} + w(2,4)$

$= \min\{0+3, 3+0\} + 5 = 3+5 = 8$

$r_{24} = 3$

$w_{03} = P(3) + q(3) + w(0,2) = 1+1+12 = 14$

$c_{03} = \min\{c(0,0) + c(1,3), c(0,1) + c(2,3), c(0,2) + c(3,3)\} + w(0,3)$

$= \min\{0+14, 8+5, 19+0\} + 14 = 13+14 = 27$

$r_{03} = 1$

$w_{14} = P(4) + q(4) + w(1,3) = 1+1+9 = 11$

$c_{14} = \min\{c(1,1) + c(2,4), c(1,2) + c(3,4), c(1,3) + c(4,4)\} + w(1,4)$

$= \min\{0+8, 7+3, 14+0\} + 11 = 8+11 = 19$

$r_{14} = 2$ or 4

$w_{04} = P(4) + q(4) + w(0,3) = 1+1+14 = 16$

$c_{04} = \min\{c(0,0) + c(1,4), c(0,1) + c(2,4), c(0,2) + c(3,4), c(0,3) + c(4,4)\} + w(0,4)$

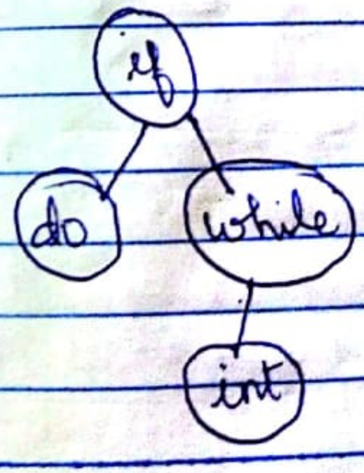
$= \min\{0+19, 8+8, 19+3, 25+0\} + 16 = 16+16 = 32$

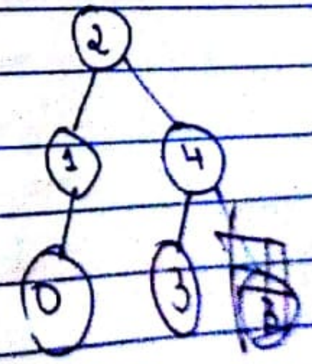
$r_{04} = 2$

$r_{04} = 2$

$0\,1$ $\quad$ $3\,4$

$r_{01} = 1$ $\quad$ $r_{34} = 4$





08-03-'18

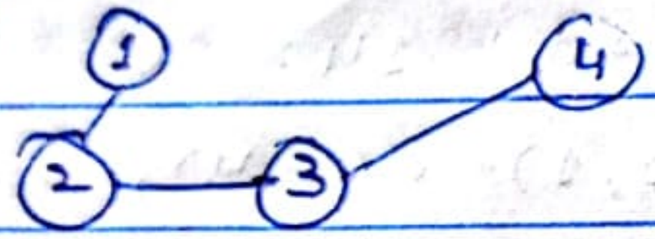
1. $(a_1, a_2, a_3, a_4) = (\text{count}, \text{float}, \text{if}, \text{while})$

$p(1:4) = (3/20, 1/5, 1/10, 1/20) = (3, 4, 2, 1)$

$q(0:4) = (1/5, 1/10, 1/5, 1/20, 1/20) = (4, 2, 4, 2, 2)$

Construct an OBST.

2.



Basic Traversal and Minimum Spanning Trees

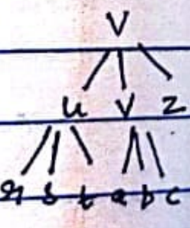
* Tree Traversal
  - → Pre Order (PLR)
  - → In Order (LPR)
  - → Post Order (LRP)

* Graph Traversal

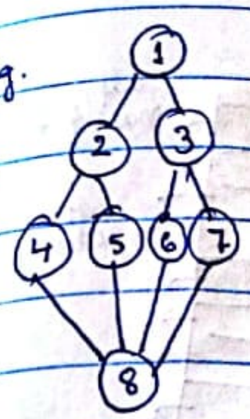
- Visiting a node
  - just visiting the node
- Exploring a node
  - visiting all the adjacent nodes of this node.

→ BFS (Breadth First Search)

- Implemented using a 'queue'.
- Add v to queue 'q'. 'v' is visited but not explored.
- Now explore the nodes in 'q'.
- We explore 'v' and add 'u', 'v', 'z' to 'q'.
- Remove 'v' from 'q' since it is explored.
- Repeat this process until queue has 0 elements.
- At this stage, BFS process is completed.



Eg.



Queue q

Visited:

1 2 3 4 5 ~~6~~ 7 8, 1 3 2 6 7 4 5 8

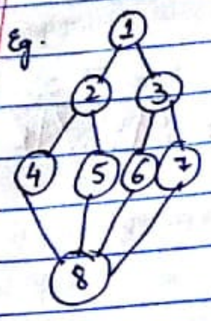
BFS Minimum Spanning Tree

- In BFS, we immediately explore current visited node.
- In DFS, we suspend exploring of current visited node and move to next node.

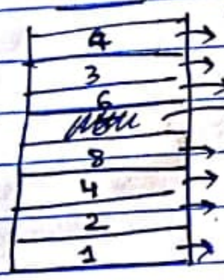
→ DFS (Depth First Search)

- Implemented using a 'stack'.
- Uses the phenomenon of backtracking.
- Whenever we are visiting a node, we are adding it to the stack.
- If all the adjacent nodes of the present node are ~~not~~ visited i.e., this node is explored ~~completed~~, go back to the node you visited before this and pop this node from the stack.
- Repeat this process until stack has 0 elements.
- At this stage, DFS process is completed
- Whenever we are suspending exploring a node, we must add it to stack.

Eg.



Stack:



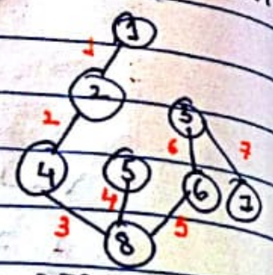
adj nodes of 7 are visited, don't add

adj nodes of 5 are visited, don't add

Visited:

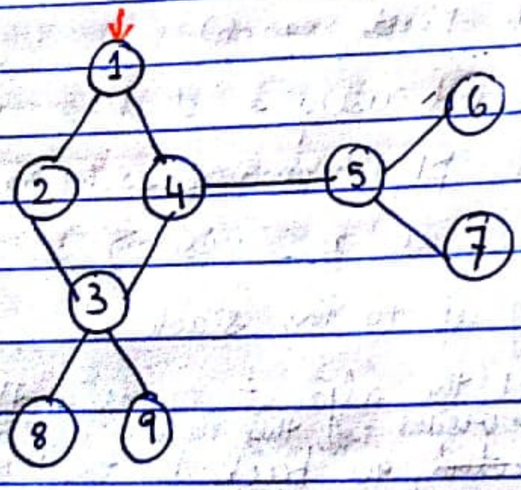
1 2 4 8 5 6 3 7

1 3 7 8 6 5 2 4



↑ DFS minimum spanning tree

Eg. Write all possible BFS and DFS orders for the following graph:



BFS

i) From (1)

1 2 4 3 5 8 9 6 7

1 4 2 3 5 8 9 6 7

1 4 2 5 3 6 7 8 9

1 2 4 3 5 9 8 6 7

1 2 4 3 5 9 8 7 6

1 2 4 3 5 8 9 7 6

1 4 2 3 5 9 8 6 7

1 4 2 3 5 9 8 7 6

1 4 2 3 5 8 9 7 6

DFS

i) From (1)

1 2 3 8 9 4 5 6 7

1 4 5 6 7 3 2 8 9

1 4 3 2 9 8 5 6 7

1 2 3 8 4 5 7 6 9

1 2 3 9 4 5 6 7 8

1 2 3 9 4 5 7 6 8

1 2 3 9 4 5 8 7 8

1 2 3 9 8 4 5 7 6

BFS, DFS

→ To maintain a graph using adjacency matrix, time complexity : $O(n^2)$

→ To maintain a graph using adjacency list / linked list time complexity : $O(n+e)$

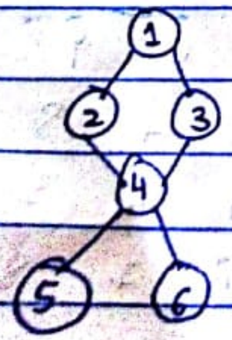
where n = no. of vertices, e = no. of edges

→ Articulation Point

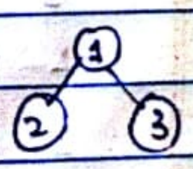
Given undirected graph 'G', we can delete vertex and all edges adjacent to vertex 'v', creates two or more non empty parts / components, then 'v' is an articulation point

A vertex 'v' in a connected graph 'G' is an articulation point if and only if the deletion of vertex 'v' together with all edges incident to 'v' disconnects the graph into two or more non empty components.

Eg.



=> Let 4 be articulation point



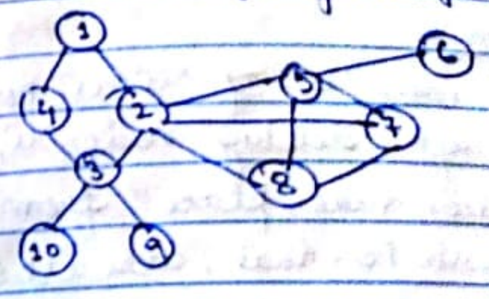
=> 3 non empty components

Biconnected Components and DFS

If given graph 'G' is biconnected component, then it has no articulation points.

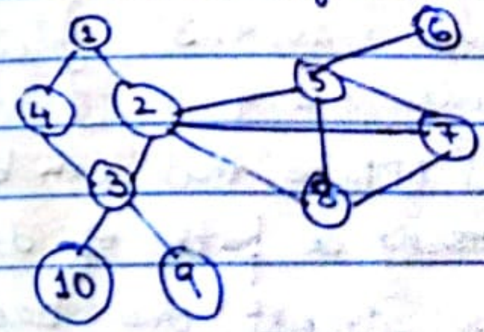
How to identify articulation points?

i) Construct DFS for given graph 'G'

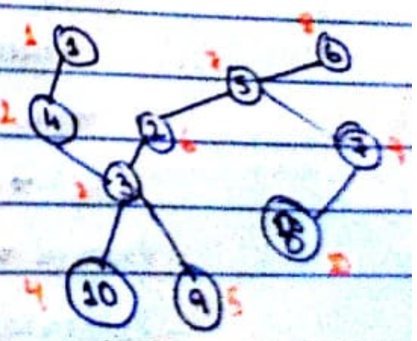


=> 1, 4, 3, 10, 9, 2, 5, 6, 7, 8

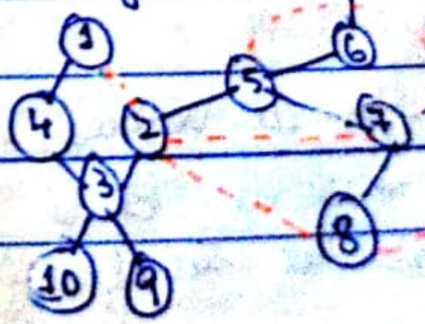
ii) Assign depth first number (DFN no) to each node
- sequence number of visited node



=>



iii) Construct Back edges in DFS graph (dotted lines) for the edges missing in original graph



iv) We must check for articulation points using 3 observations

Observation 1:
'v' is root node. If 'v' is an articulation point, it contains atleast two edges (two solid lines), no back edges).

Observation 2:
Leaf nodes are not articulation points.

Observation 3:
Consider any node 'u'. If 'u' is not articulation point, even after deleting node 'u', the communication takes place from descendants to antecedants. For that, atleast one back edge is required.

Eg. ③

Descendants => 10, 9, 3
Antecedants => 1, 4
} in DFS

L(u) → Least (Lowest) DFN no that can be reached from u using a path of descendants followed by atmost one back edge.

$L(u) = \min\{DFN(u), \min\{L(w) / w \text{ is child of } u\}, \min\{DFN(w) / (u,w) \text{ is back edge}\}\}$

After finding L(u) for all vertices, verify
$L(w) \geq DFN(u)$ → w is child of u'
If the above condition is satisfied 'u' is articulation pt

$L(u) = \min\{DFN(u), \min\{L(w) / w \text{ is child of } u\}, \min\{DFN(w) / (u,w) \text{ is back edge}\}\}$

$L(1) = \min\{DFN(1), \min\{L(4)\}, \min\{DFN(2)\}\} = \min\{1, \min\{1\}, \min\{1\}\} = 1$

$L(4) = \min\{DFN(4), \min\{L(3)\}\} = \min\{2, \min\{1\}\} = 1$

$L(3) = \min\{DFN(3), \min\{L(10), L(9), L(2)\}\} = \min\{3, \min\{4, 5, 1\}\} = 1$

$L(10) = \min\{DFN(10)\} = \min\{4\} = 4$

$L(9) = \min\{DFN(9)\} = \min\{5\} = 5$

$L(2) = \min\{DFN(2), \min\{L(5)\}, \min\{DFN(7), DFN(8), DFN(2)\}\} = 1$

$L(5) = \min\{DFN(5), \min\{L(7), L(6)\}, \min\{DFN(8)\}\}$
$= \min\{7, \min\{6, 8\}, \min\{10\}\} = 6$

$L(7) = \min\{DFN(7), \min\{L(8), L(5)\}, \min\{DFN(2)\}\}$
$= \min\{9, \min\{6\}, \min\{6\}\} = 6$

$L(8) = \min\{DFN(8), \min\{L(7)\}, \min\{DFN(2), DFN(5)\}\}$
$= \min\{10, 6, 7\} = 6$

$L(6) = \min\{DFN(6)\} = \min\{8\} = 8$

$L(w) \geq DFN(u)$

- $L(4) \geq DFN(1) \Rightarrow 1 \geq 1\ \checkmark \Rightarrow V1$
  But 1 is root node. it contains only 1 edge, not articulation
- $L(3) \geq DFN(4) \Rightarrow 1 \geq 2\ \times \Rightarrow V4$
  Not articulation point
- $L(10) \geq DFN(3)$, $L(9) \geq DFN(3)$, $L(2) \geq DFN(3) \Rightarrow V4$
  $4 \geq 3\ \checkmark$, $5 \geq 3\ \checkmark$, $1 \geq 3\ \times \Rightarrow V4$
  Articulation point

08-03-'18

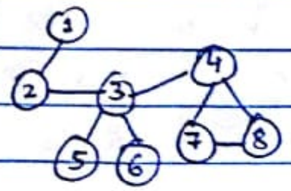
1. (a_0, a_2, a_3, a_4) : (count, float, if, while)

$P(1:4) = (1/20, 1/5, 1/10, 1/20)$

$q(0:4) : (1/5, 1/20, 1/15, 1/20, 1/20)$

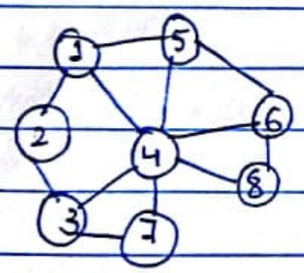
Construct OBST

2.



Find the articulation points

3.



Check if the given graph is biconnected or not.

→ Formulae:

1.) $c(i,j) = \min \{c(i,k-1) + c(k,j)\} + w(i,j)$

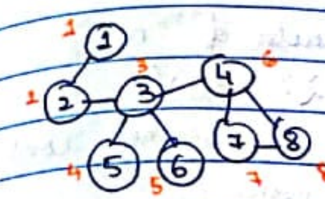
$w(i,j) = P(j) + q(j) + w(i,j-1)$

$w(i,i) = q(i)$

$c(i,i) = 0$

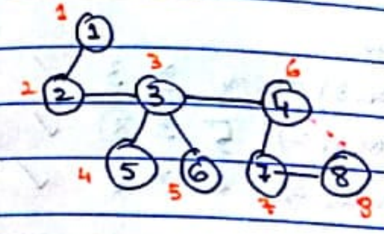
$r(i,i) = 0$

2), 3) $L(u) = \min \{DFN(u), \min \{L(w) / w \text{ is child of } u\}, \min \{DFN(w) / u,w \text{ is back edge}\}\}$



DFS: 1, 2, 3, 5, 6, 4, 7, 8

⇒



$L(u) = \min \{DFN(u), \min \{L(w) / w \text{ is child of } u\}, \min \{DFN(w) / (u,w) \text{ is back edge}\}\}$

$L(1) = \min \{DFN(1), \min \{L(2)\}\} = \min \{1, \min \{2\}\} = \min \{1,2\} = 1$

$L(2) = \min \{DFN(2), \min \{L(3)\}\} = \min \{2, \min \{3\}\} = \min \{2,3\} = 2$

$L(3) = \min \{DFN(3), \min \{L(5), L(6), L(4)\}\}$

$= \min \{3, \min \{4,5,6\}\} = \min \{3,4\} = 3$

$L(4) = \min \{DFN(4), \min \{L(7)\}, \min \{DFN(8)\}\}$

$= \min \{6, \min \{7\}, \min \{8\}\} = \min \{6,7,8\} = 6$

$L(5) = \min \{DFN(5)\} = \min \{4\} = 4$

$L(6) = \min \{DFN(6)\} = \min \{5\} = 5$

$L(7) = \min \{DFN(7), \min \{L(8)\}\} = \min \{7, \min \{6\}\} = \min \{7,6\} = 6$

$L(8) = \min \{DFN(8), \min \{DFN(4)\}\} = \min \{8, \min \{6\}\}$

$= \min \{8,6\} = 6$

$L(w) \geq DFN(n)$ // w is child of n

- $V_1 \Rightarrow L(2) \geq DFN(1) \Rightarrow 2 \geq 1 \Rightarrow \checkmark$

But 1 is root node and it does not contain 2 edges

∴ 1 is not articulation point

- $V_2 \Rightarrow L(3) \geq DFN(2) \Rightarrow 3 \geq 2 \Rightarrow \checkmark$

Articulation Point

- $V_3 \Rightarrow L(5) \geq DFN(3) \Rightarrow 4 \geq 3 \checkmark$

  $L(6) \geq DFN(3) \Rightarrow 5 \geq 3 \checkmark$

  $L(4) \geq DFN(3) \Rightarrow 6 \geq 3 \checkmark$

Articulation Point

- $V_4 \Rightarrow L(7) \geq DFN(4) \Rightarrow 6 \geq 6 \checkmark$

Articulation Point

- $V_5 \Rightarrow$ Leaf node

Not Articulation Point

- $V_6 \Rightarrow$ Leaf node

Not Articulation Point

- $V_8 \Rightarrow$ Leaf node

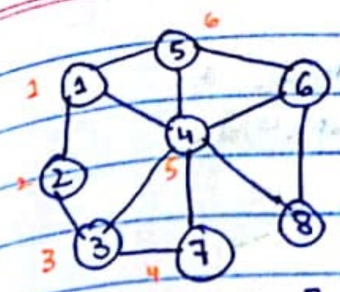
Not Articulation Point

- $V_7 \Rightarrow (L8) \geq DFN(7) \Rightarrow 6 \geq 7$ X

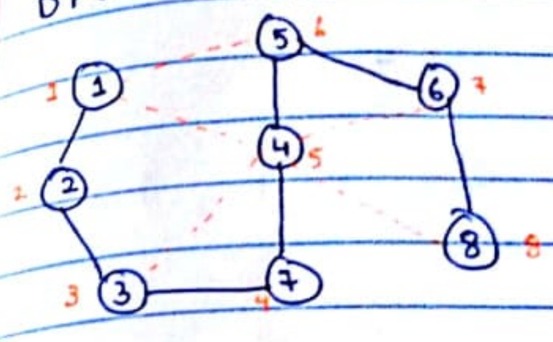
Not Articulation Point

Articulation Points = (2), (3), (4)

A. 3)



DFS: 1, 2, 3, 7, 4, 5, 6, 8



$L(u) = \min\{DFN(u), \min\{L(w) \mid w \text{ is child of } u\}, \min\{DFN(w), \mid (u,w) \text{ is back edge}\}\}$

$L(1) = \min\{DFN(1), \min\{L(2)\}, \min\{DFN(5), DFN(4)\}\}$

$= \min\{1, \min\{2\}, \min\{6, 5\}\} = \min\{1, 2, 5\} = 1$

$L(2) = \min\{DFN(2), \min\{L(3)\}\} = \min\{2, \min\{3\}\} = \min\{2, 3\} = 2$

$L(3) = \min\{DFN(3), \min\{L(7)\}, \min\{DFN(4)\}\}$

$= \min\{3, \min\{5\}, \min\{5\}\} = \min\{3, 5, 5\} = 3$

$L(4) = \min\{DFN(4), \min\{L(5)\}, \min\{DFN(1), DFN(3), DFN(6), DFN(8)\}\}$

$= \min\{5, \min\{1\}, \min\{1, 3, 7, 8\}\} = \min\{5, 1, 1\} = 1$

$L(5) = \min\{DFN(5)\}, \min\{L(6)\}, \min\{DFN(1)\}\}$

$= \min\{6, \min\{5\}, \min\{1\}\} = \min\{6, 5, 1\} = 1$

$L(6) = \min\{DFN(6), \min\{L(8)\}, \min\{DFN(4)\}\}$

$= \min\{7, \min\{5\}, \min\{5\}\} = \min\{7, 5, 5\} = 5$

$L(7) = \min\{DFN(7), \min\{L(4)\}\} = \min\{7, \min\{5\}\} = \min\{7, 5\} = 5$

$L(8) = \min\{DFN(8), \min\{DFN(4)\}\} = \min\{8, \min\{5\}\} = \min\{8, 5\} = 5$