Unit-2

Greedy Method

General method:

The Greedy method is a straight forward and powerful technique to design algorithms. It is popular to obtain optimal solution, most of these problems have 'n' inputs and require us to obtain a subset that satisfies some constraints. Any subset that satisfies these constraints is called a feasible solution

A feasible solution which maximises and minimises the objective function. So when the required solution is obtained it is called as optimal solution. The Greedy method divides an algorithm that works in stages, considering one input at a time, at each stage a decision is made regarding whether a particular input is in an optimal solution. This is done by some selection procedure. The feasible solutions are added to the partially constructed optimal solution. If the input is infeasible then that will not be added to the partial solution. The selection process itself is based on some optimisation measure. This measure may be the objective function. This kind of selection process is called selection paradigm

Control abstraction:

```
Algorithm Greedy(a,n)
{
  Solution := 0;
  for i:=1 to n do
  {
    x := select(a);
    if (feasible(solution,x)) then
```

Scanned with CamScanner

}
return solution;
}

Applications of Greedy method:

1) knapsack problem
2) Job Scheduling with deadlines
3) Minimum Cost Spanning tree
4) Optimal storage on tapes
5) Single source shortest path problem

Advantages:

→ It is easy to write the algorithm
→ It is easy to write the code
→ It produces efficient results

Disadvantage:

It doesn't give guarantee for the optimal solution

Knapsack Problem:-

In this method 'n' objects are given and a knapsack (or) empty bag are given. object 'i' has a weight w_i and the knapsack has the capacity 'm'. Now the knapsack problem is that we should place the objects into the bag without exceeding the bag capacity

If a fraction x_i, $0 \le x_i \le 1$, of object 'i' is placed into the knapsack then a profit of '$p_i x_i$' is earned.

The problem can be stated as Maximise $\sum_{1 \le i \le n} p_i x_i$ such that $\sum_{1 \le i \le n} w_i x_i$ such that $1 \le i \le n$. The profits and weights are +ve numbers

51/107

Scanned with CamScanner

Example: Consider the following instance of knapsack problem $n = 3$, $m = 20$

$(P_1, P_2, P_3) = (25, 24, 15)$

$(W_1, W_2, W_3) = (18, 15, 10)$

Case(i): Maximum profit

Case(ii): Minimum weight

Case(iii): Maximum profit per unit weight

Case(iv): Maximum weight

Sol:

Case(i): Maximum Profit

Here we need to place an item in the bag whose profit is Maximum

$X_1 = 1$, $X_2 = 2/15$, $X_3 = 0$

$\Rightarrow \sum P_i X_i = P_1 X_1 + P_2 X_2 + P_3 X_3$

$= 25(1) + 24(2/15) + 15(0)$

$= 28.2$

Case(ii): Minimum weight:

We place an item in the bag whose weight is minimum

$X_1 = 0$, $X_2 = 10/15$, $X_3 = 1$

$\sum P_i X_i \Rightarrow P_1 X_1 + P_2 X_2 + P_3 X_3$

$= 25(0) + 24(10/20) + 15(1)$

$= 0 + 12 + 15$

$= 31$

Case(iii): Maximum profit per unit weight

We place an item in the bag whose P/W ratio is Maximum

$\frac{P_1}{W_1} = \frac{25}{18} = 1.4$ $\frac{P_2}{W_2} = \frac{24}{15} = 1.6$ $\frac{P_3}{W_3} = \frac{15}{10} = 1.5$

$X_1 = 0$, $X_2 = 1$, $X_3 = 5/10 = 1/2$

52/107

Scanned with CamScanner

$\sum P_i X_i \Rightarrow P_1X_1 + P_2X_2 + P_3X_3$

$= 25(0) + 24(1) + 15(1/2)$

$= 24 + 7.5$

$= 31.5$

Case(iv): Maximum weight

$X_1 = 1;\ X_2 = 2/15;\ X_3 = 0$

$\sum W_i X_i = W_1X_1 + W_2X_2 + W_3X_3$

$= 18(1) + 15(2/15) + 10(0)$

$= 18 + 2$

$= 20$

Algorithm:

Algorithm Greedyknapsack(m,n)

{

for i:=1 to n do

x[i]:=0.0;

U:=m;

for i:=1 to n do

{

if (w[i]>U) then

break;

x[i]:=1.0;

U:=U-w[i];

}

if (i≤n) then

x[i]:=U/w[i];

}

Intra

① Ex n=7

m=15

$\{P_1, P_2 \cdots P_7\} = \{10,5,15,7,6,18,3\}$

$\{w_1, w_2 \cdots w_7\} = \{2,3,5,7,1,4,1\}$

53/107

Scanned with CamScanner

Task scheduling with deadlines:

Scanned with CamScanner

Job Scheduling with deadlines:

Given a set of 'n' jobs. Job 'i' is associated with an integer deadline $d_i \geq 0$ and a profit $P_i > 0$

For any job 'i' the profit earned if and only if the job is completed by its deadline. To complete a job one has to process the job on a machine for 1 unit of time. Only one machine is available for processing the jobs.

A feasible solution for this problem is a subset 'j' of jobs such that each job in this subset can be completed by its deadline. The value of a feasible solution 'j' is sum of the profits of the job 'j', And optimal solution with maximum value.

Ex:

n	P_i	d_i
1	70	2
2	12	1
3	18	2
4	35	1

(1,3)
(2,1)
(2,3)
(3,1)
(4,1)
(4,3)

n	P_i
(1,3)	88
(2,1)	82
(2,3)	30
(3,1)	88
(4,1)	105
(4,3)	53

{4,1} is the optimal solution. which produce maximum profit

Example

J_1	J_2	J_3	J_4	J_5	J_6
5	3	3	2	4	2
200	180	190	300	120	100

optimal → 990 units

J_2, J_4, J_3, J_5, J_1

54/107

Scanned with CamScanner

Find the optimal solution of the knapsack instances

$n=7, M=15$

$(P_1, P_2, P_3, P_4, P_5, P_6, P_7) = (10, 5, 15, 7, 6, 18, 3)$

$(W_1, W_2, W_3, W_4, W_5, W_6, W_7) = (2, 3, 5, 7, 1, 4, 1)$

Case(i): Maximum profit

$X_1 = 1, X_2 = 0, X_3 = 1, X_4 = 4/7, X_5 = 0, X_6 = 1, X_7 = 0$

$\sum P_i X_i = 10(1) + 5(0) + 15(1) + 7(4/7) + 6(0) + 18(1) + 3(0)$

$= 10 + 15 + 4 + 18$

$= 47$

Case(ii): Minimum weight

$X_1 = 1, X_2 = 1, X_3 = 4/5, X_4 = 0, X_5 = 1, X_6 = 1, X_7 = 1$

$\sum P_i X_i = 10(1) + 5(1) + 15(4/5) + 7(0) + 6(1) + 18(1) + 3(1)$

$= 10 + 5 + 12 + 6 + 18 + 3$

$= 54$

Case(iii): Maximum profit per unit weight:

$\frac{P_1}{W_1} = 5, \frac{P_2}{W_2} = 1.6, \frac{P_3}{W_3} = 3, \frac{P_4}{W_4} = 1, \frac{P_5}{W_5} = 6, \frac{P_6}{W_6} = 4.5, \frac{P_7}{W_7} = 3$

$X_1 = 1, X_2 = 1/3, X_3 = 1, X_4 = 0, X_5 = 1, X_6 = 1, X_7 = 1$

$\sum P_i X_i = 10(1) + 5(0.5) + 15(1) + 7(0) + 6(1) + 18(1) + 3(1)$

$= 10 + 2.5 + 15 + 0 + 6 + 18 + 3$

$= 54.5$

Case(iv): Maximum weight: $X_1 = 0, X_2 = 0, X_3 = 1, X_4 = 1, X_5 = 0, X_6 = 1, X_7 = 0$

$\sum P_i X_i = 10(0) + 5(0) + 15(1) + 7(1) + 6(0) + 18(1) + 3(0)$

$= 15 + 7 + 18$

$= 40$

Ex: Job Sequencing $n=4$

i) $(P_1, P_2, P_3, P_4) = (100, 10, 15, 27)$

$(D_1, D_2, D_3, D_4) = (2, 1, 2, 1)$

ii) $(P_1, P_2, P_3, P_4, P_5) = (20, 15, 10, 5, 1)$

$(D_1, D_2, D_3, D_4, D_5) = (2, 2, 1, 3, 3)$

55/107

Scanned with CamScanner

Algorithm:

```
Algorithm GreedyJob(d, j, n)
{
  J := {1}
  for i := 2 to n do
  {
    if (All jobs in J ∪ {i} can be completed by their deadlines)
    then
      J = J ∪ {i};
  }
}
```

```
Algorithm JS(d, j, n)
{
  d[0] := J[0] := 0;
     J[1] := 1;
     k := 1;
  for i := 2 to n do
  {
     r := k;
  while((d[J[r]] > d[i]) and (d[J[r]] ≠ r)) do
     r := r - 1;
  if ((d[J[r]] ≤ d[i]) and (d[i] > r)) then
  {
  for q := k to (r+1) step -1 do
     J[q+1] := J[q];
     J[r+1] := i;
     k := k+1;
  }
  return k;
  }
}
```

$S_1 - \frac{2}{=}$

$S_2 - \frac{}{=}\ {}^4C_1$

$S_3 - \frac{2}{=}\ \frac{4\times3\times2\times1}{1\times2}$ (1)

$S_4 - \frac{}{=}\ {}^4C_1 + {}^4C_2 + {}^4C_3 + {}^4C_4$

$= 4 + 6 + 4 + 1$

$= 15$

S.NO	Feasible sol	Procession Seq	value	Remarks

$= {}^nC_r = \frac{n!}{(n-r)!\,r!}$

$= \frac{5!}{3!\times2!} + \frac{4!}{2!\times2!}$

$= \frac{5\times4\times3!}{3!\times2!} + \frac{4\times3\times2!}{2!\times2!}$

$= 10 + 6$

$= 16$

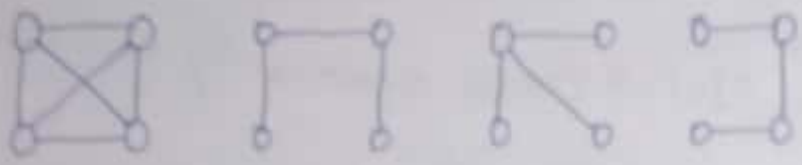
$nP_r = \frac{n!}{(n-r)!}$

Scanned with CamScanner

Minimum cost spanning tree:

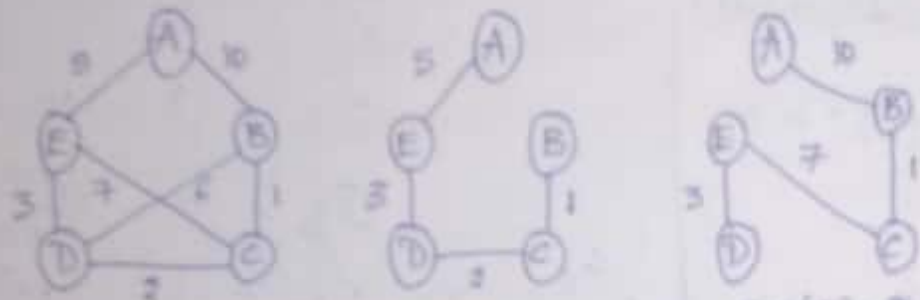
Let G=(V,E) be an undirected connected graph. A subgraph T=(V,E') of 'G' is a spanning tree if and only if 'T' is a tree.

Eg:



Minimum cost spanning trees of weighted connected graphs 'G' is a spanning tree with minimum(or) smallest weights.

Eg:



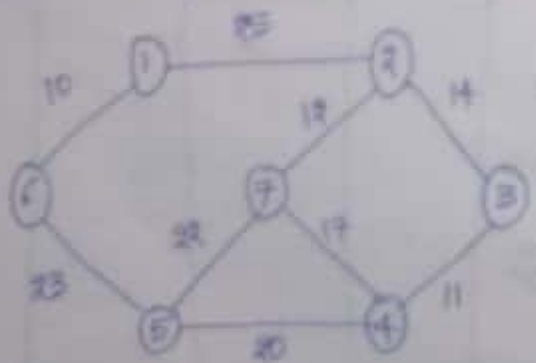
In the above example graph 'T1' have the minimum weight. Therefore 'T1' is considered as the minimum cost spanning tree.

Minimum cost spanning tree can be obtained by using the following two algorithms

i) Prim's algorithm
ii) Kruskal's algorithm

i) Prim's algorithm:

Consider the graph given below



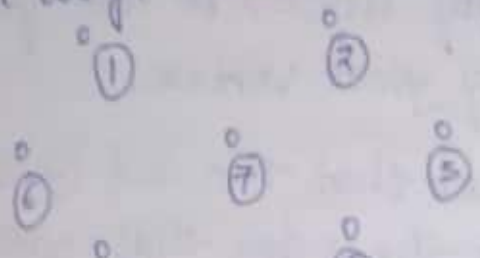
Now consider all the vertices as '0' then select one vertex as the first vertex, then select adjacent vertices of first vertex

57/107

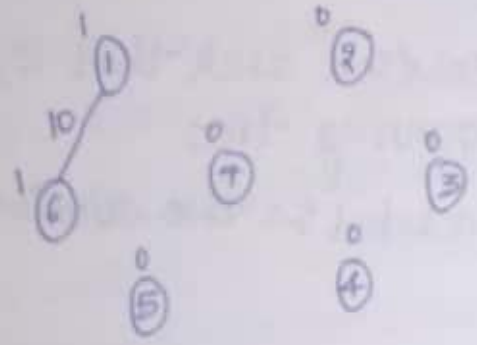
Scanned with CamScanner

among the adjacent vertices, select one vertex with minimum edge and the algorithm proceeds by selecting adjacent edges with minimum weight, care should be taken for not forming circuit

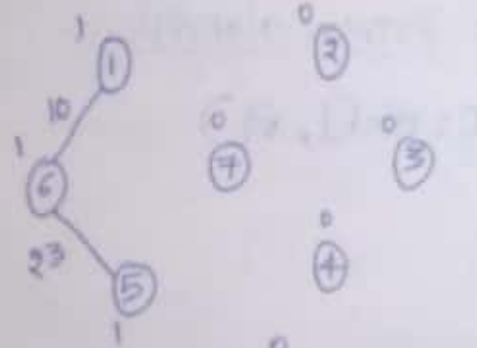
step 1:



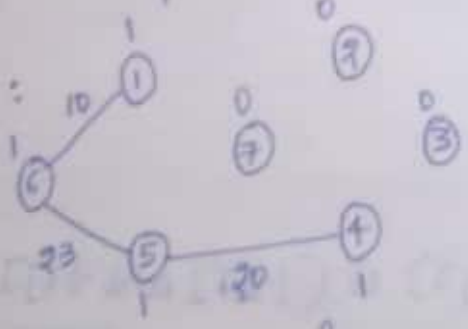
step 2:



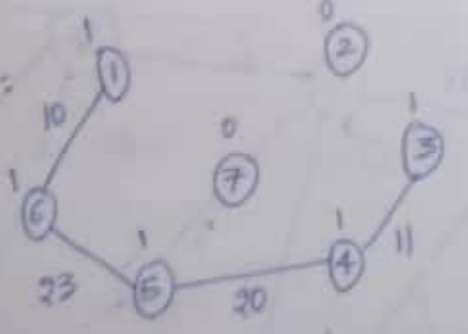
step 3:



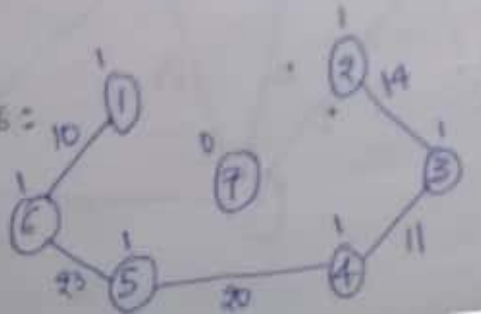
step 4:



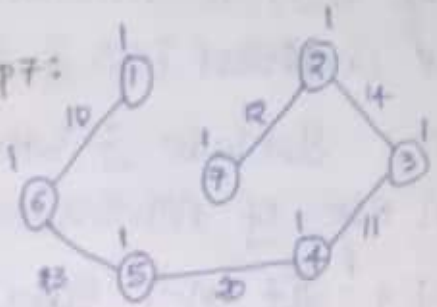
step 5:



step 6:



step 7:



Total weight = 10 + 23 + 20 + 11 + 14 + 12

Total weight = 90

58/107

Scanned with CamScanner

Scanned with CamScanner

Algorithm:

Step1: Let 's' be the solution for minimum spanning tree which is initially empty G(V,E) be the graph. Include any vertex i, say first vertex.

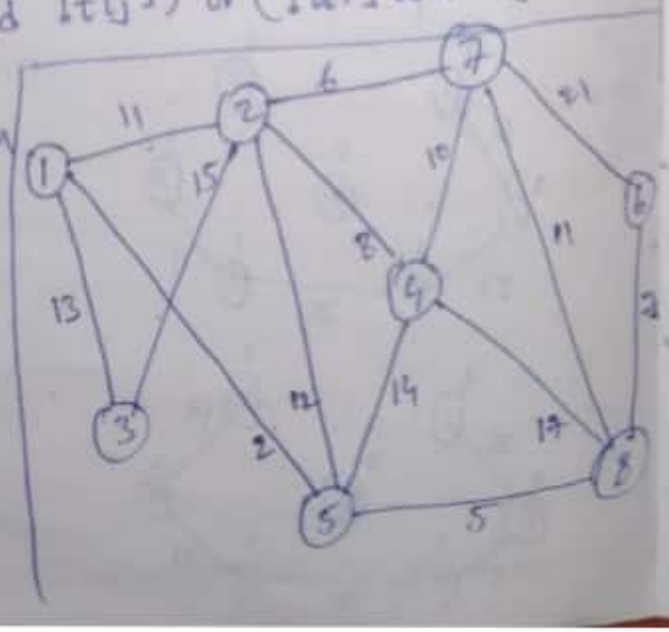
Step2: Select a vertex 'j' such that the cost of the edge (i,j) is minimum along the edges that satisfy the condition $i \in s$ & $j \notin s$

Step 3: Include 'j' into the solution such that (i,j) is an edge of minimum cost spanning tree

Step 4: If 'n-1' edges are included in the solution then stop, else goto step 2

Detailed Implementation of Prim's algorithm:-

```
Algorithm prim's (G[0:n-1,0:n-1], n)
for i←0 to n-1 do
    t[i]←0;   t[0]←1;
mindist←0.0;
for i←0 to n-1 do
  {
    for j←1 to n-1 do
  {
if (G[i,j] and ((!t[i] and t[j]) or (!t[i] and t[j])) then
  {
if (G[i,j]← mindist) then
    {
    mindist← G[i,j];
      V1← i; V2← j;
    }
  }
}
write (V1, V2, mindist)
  t[V1]← t[V2]←1;
```



Scanned with CamScanner

Scanned with CamScanner

total ← total + mindist;

}

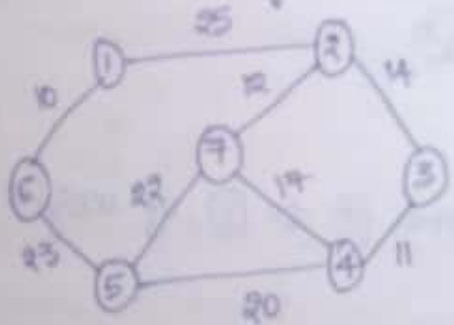
write("Total path set cost", total);

Kruskal's Algorithm :-

kruskal's Algorithm is the another algorithm of obtaining minimum spanning tree. This algorithm was discovered by a II year graduate Joseph kruskals. In this algorithm always the minimum cost edge has to be selected but it is not necessary that the selected edge is adjacent.

First we will select all the vertices then an edge with optimum weight is selected from heap even though it is not adjacent to previously selected edge. Care should be taken for not forming circuit

Eg:

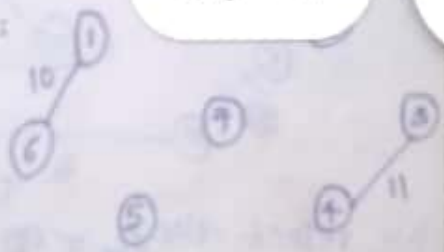


step 1:

$T_W = 0$

step 2:

step 3:

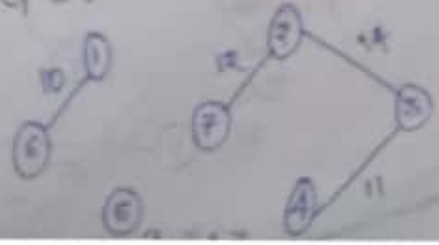


$T_W = 21$

step 4:

$T_W = 33$

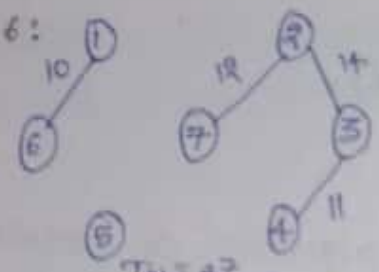
step 5:



Scanned with CamScanner

Scanned with CamScanner

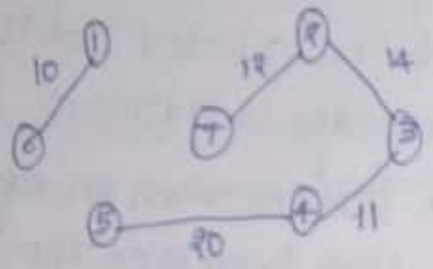
Step 6:



$T_W = 47$

We reject the edge weight '17' i.e from (4) to (7) as it forms a loop

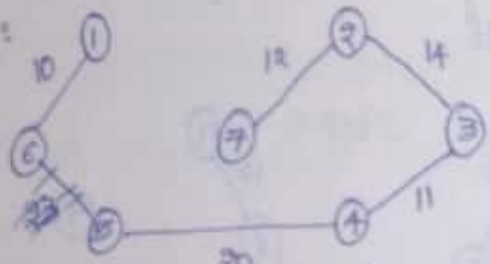
step 7:



$T_W = 67$

67
23
90

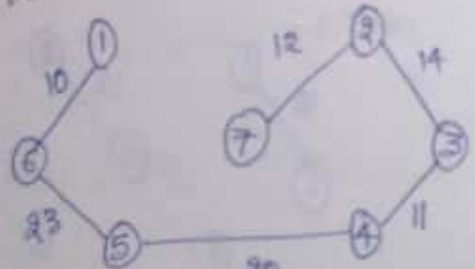
step 8:



$T_W = 67$

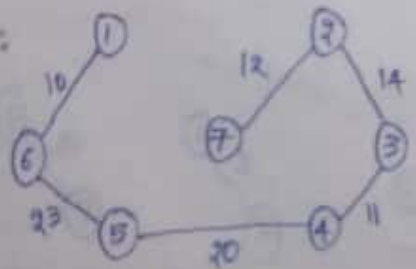
We reject the $T_W = 90$ edge from (5) to (7) of weight '23' since it forms a loop

step 9:



$T_W = 90$

step 10:



$T_W = 90$

Reject the edge from node (1) to node (3) since it forms a loop

Scanned with CamScanner

Scanned with CamScanner

Algorithm:

Step 1: Arrange all the edges in the increasing order of cost

Step 2: Include the edge with minimum cost in to the solution set.

Step 3: Add the next edge with minimum cost and delete edge from edge set until 'n-1' edges are added (or) the edge set become empty

Step 4: If the inclusion of any edge results in a cycle, reject it and move to the next edge

Step 5: Goto step3

Detailed Implementation of kruskal's Algorithm:

```
Algorithm kruskals(E,cost,n,t)
{
Construct a heap out of the edge costs using heapify;
for i← 1 to n do
    P[i]:= -1;
i:=0 ; mincost:=0.0;
while((i< n-1) and (heap not empty)) do
{
Delete a minimum cost edge (u,v) from 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];
```

62/107

Scanned with CamScanner

Scanned with CamScanner

```
        union[j,k];
      }
    }
  }
  if (i != n-1) then
  write ("No spanning tree");
  else
  return mincost;
  }
```

Optimal storage on Tapes:

There are n- programs that are to be stored on a computer tape of length 'l' associated with each program 'i' is a length l_i, $1 \le i \le n$. All programs can be stored on the tape if & only if the sum of the length of the program is almost 'l'. We assume that whenever program is to be retrieved from this tape, the tape is initially positioned at the end. Hence if the programs are stored in the order $I = I_1, I_2, \ldots I_n$. The time t_j needed to retrieve the program I_j and is proportional to $\sum_{1 \le k \le j} l_{i_k}$. If all the programs are retrieved equally often, then the expected/mean retrieval mean is $\frac{1}{n} \sum t_j$

63/107

In this problem we require to find a permutation for 'n' programs so that when they are stored on a tape, Mean Retrieval Time is minimized.

Minimising the MRT is equivalent to minimising the

$$d(I) = \sum_{1 \le j \le n} \sum_{1 \le k \le j} l_{i_k}$$

Eg: Let $n = 3$, $(l_1, l_2, l_3) = (5, 10, 3)$

$n! = 3! = 6$

Scanned with CamScanner

Minimising the MRT is equivalent to

$$d(I) = \sum_{1 \le j \le n} \sum_{1 \le k \le j} l_{I_k}$$

Eg: Let $n=3$, $(l_1, l_2, l_3) = (5, 10, 3)$

Soln: $n! = 3! = 6$

Refer.

Scanned with CamScanner

There are $n! = 3! = 3 \times 2 \times 1 = 6$ possible orderings. These orderings and their respective 'd' values are

Orderings I	d(I)
1, 2, 3	5+5+10+5+10+3 = 38
1, 3, 2	5+5+3+5+3+10 = 31
2, 1, 3	10+10+5+10+5+3 = 43
2, 3, 1	10+10+3+10+3+5 = 41
3, 1, 2	3+3+5+3+5+10 = 29
3, 2, 1	3+3+10+3+10+5 = 34

The optimal ordering is (3,1,2)

Algorithm:

```
Algorithm store(n, m)
{
j := 0;
for i := 1 to n do
{
write("append program", i, "to permutation for tape", j);
j := (j+1) mod m;
}
}
```

64/107

Single Source Shortest Path Problem:-

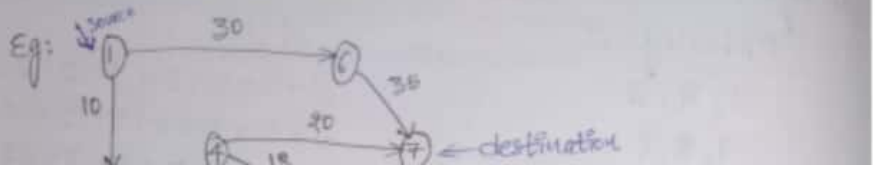
Many times graph is used to represent the distances b/n two cities. In single source shortest path problem the shortest distance from a single vertex called source is obtained.

Let G(V,E) be a graph, then in single source shortest path from vertex V_0 to all remaining vertices is determined. The vertex V_0 is called source and the

Scanned with CamScanner

last vertex is called destination. Assume that all the distances are +ve

Eg:



Scanned with CamScanner

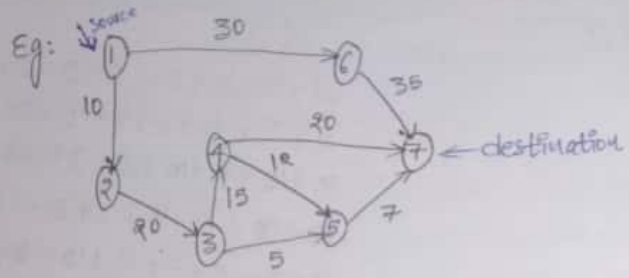
two cities. In single source shortest path
shortest distance from a single vertex called source is
obtained.

Let G(V,E) be a graph, then in single source shortest
path from vertex V_0 to all remaining vertices is
determined. The vertex V_0 is called source and the

Scanned with CamScanner

last vertex is called destination. Assume that all
the distances are +ve

Eg:



Start from source vertex, say 1st vertex s[1]=1. Now
select next vertex from vertex '1'

{1,2} = 10 {1,3} = ∞ {1,4} = ∞ {1,5} = ∞ {1,6} = 30

{1,7} = ∞

Hence set s[2]=1

{1,2} produces shortest path. Hence set s[2]=1.
From vertex '2' select the next vertex.

{1,2,3} = 30, {1,2,4} = ∞ {1,2,5} = ∞

{1,2,6} = ∞ {1,2,7} = ∞

65/107

{1,2,3} produces shortest path. Hence set s[3]=1
From vertex '3' select the next vertex

{1,2,3,4} = 45 {1,2,3,5} = 35 {1,2,3,6} = ∞

{1,2,3,7} = ∞

{1,2,3,5} produces shortest path. Hence set s[5]=1
From vertex 5 select the next vertex

{1,2,3,5,4} = ∞ {1,2,3,5,6} = ∞ {1,2,3,5,7} = 42

{1,2,3,5,7} produces, shortest path. Hence set s[7]=1
From vertex '7', select the next vertex

{1,2,3,5,7,4} = ∞ {1,2,3,5,7,6} = ∞

Scanned with CamScanner

Algorithm:

Algorithm shortest path (N, cost, dist, n)

Scanned with CamScanner

{1, 2, 3, 5} produces shortest path. Hence set s[5] = 1
From vertex 5 select the next vertex
{1, 2, 3, 5, 4} = ∞ {1, 2, 3, 5, 6} = ∞ {1, 2, 3, 5, 7} = 45
{1, 2, 3, 5, 7} produces shortest path. Hence set s[7] = 1
From vertex 7, select the next vertex
{1, 2, 3, 5, 7, 4} = ∞ {1, 2, 3, 5, 7, 6} = ∞

Scanned with CamScanner

Algorithm:

```
Algorithm shortestpath (N, cost, dist, n)
{
for i := 1 to n do
  { s[i] := false;
dist[i] := cost [v,i];
}
s[v] = true, dist[v] = 0.0;
for num := 2 to n do
                    // select u with minimum in dist
{
 s[u] := true;
for (each w adjacent to u with s[w] := false)
{
if (dist[w] > dist[u] + cost [u,w]) then
  dist[w] := dist[u] + cost[u,w];
   }
  }
}
```

66/107

Dynamic programming:-

General method:

Dynamic programming is applied to optimization problems. This technique is invented by us mathematician Richard Bellman in 1955. In the word dynamic programming, the word programming stands for planning. It is a technique for solving problems with overlapping subprograms. In this method, each subprogram is solved only once, the result of each subproblem is recorded in a table form, which can obtain a solution to the original problem.
For each given problem we may get a number of solutions

Scanned with CamScanner

We seek for optimal solution, such an optimal solution becomes the solution to the given problem

Scanned with CamScanner

Richard Bellman in 1955. In
programming, the word programming stands for planning.
It is a technique for solving problems with overlapping
subprograms. In this method, each subprogram is solved
only once, the result of each subproblem is recorded
in a table form, which can obtain a solution to
the original problem.
For each given problem we may get a number of solutions

Scanned with CamScanner

We seek for optimal solution, such an optimal solut
becomes the solution to the given problem.

Steps of dynamic Programming:-

Dynamic Programming design involves 4 steps

Step 1: Characterize the structure of optimal solu
that means develop a mathematical solution that ca
express any solution and subsolution for the given
problem.

Step 2:
Recursively define the value of an optimal solution

Step 3:
By using Bottom-up technique, compute the value of
optimal solution, for that you have to develop a
recurrance solut relation that relates a solution
to its subsolutions using mathematical solution of step 1

Step 4:
Compute an optimal solution from computed solution

67/107

Principle of Optimality:-

The DP Algorithm obtained the solution using principle
of optimality.

The principle of optimality states that "In an optim
solution sequences of decisions, (or) choices each sub
sequence must also be optimal"

When it is not possible to apply the principle of optima
it is almost impossible to obtain the solution using
dynamic programming approach

Applications of Dynamic Programming:-

1) Matrix chain multiplication
2) 0/1 knapsack problem

Scanned with CamScanner

3) Optimal Stear binary search trees

Scanned with CamScanner

When it is not possible to apply the principle of optimality it is almost impossible to obtain the solution using dynamic programming approach

Applications of Dynamic Programming :-

1) Matrix chain multiplication
2) 0/1 knapsack problem

Scanned with CamScanner

3) optimal ~~star~~ binary search trees
4) All-pairs shortest path problem
5) Travelling sales person problem
6) Reliability design

Differences b/n divide and conquer method and dynamic Programming :-

Divide and Conquer	Dynamic programming
1) The problem is divided into subproblems. These subproblems are solved independently finally all the solutions of subproblems are collected together to get the solution to the given problem.	1) In dynamic programming, many decision sequences are generated & all the overlapping sub-instances are considered.
2) In this method, duplications in subsolutions are neglected.	2) In dynamic computing, duplications are completely avoided.
3) Divide and conquer method is less efficient because of rework on solutions.	3) DP is efficient than the divide & conquer strategy.
4) It uses top down approach of problem solving	4) DP uses bottom up approach of problem solving
5) In this method, splits its input at specific deterministic points usually in the middle	5) In this method, splits its input at every possible split points rather than at a particular point after trying all the split points it determines which split points is optimal

Greedy method	Dynamic programming
1) Greedy method is used for obtaining the optimum solution	1) DP is also used for obtaining the optimum solution

Scanned with CamScanner

2) In Greedy method, a set of feasible solutions are obtained and picks up the optimum solution from feasible solutions	2) There is no special set of feasible solutions in this method
3) In greedy method, the	3) In this method, Consider all possible sequences in order

Scanned with CamScanner

Greedy method	Dynamic programming
1) Greedy method is used for obtaining the optimum solution	1) DP is also used for obtaining the optimum solution

Scanned with CamScanner

2) In Greedy Method, a set of feasible solutions are obtained and picks up the optimum solution from feasible solutions	2) There is no special set of feasible solutions in this Method.
3) In greedy method, the optimum selection is without revising previously generated solution	3) In this method, Consider all possible sequences in order to obtain the optimum solution
4) In Greedy method, there is no such guarantee of getting optimal solution.	4) It is guaranteed that the DP will generate optimal solution using principle of optimality.

Principle of optimality: The principle of optimality states that an optimal sequence of decisions has the property that whatever the initial state and decision are, the remaining decisions must constitute an optimal decision sequence with regard to the state resulting from the first decision.

69/107

Scanned with CamScanner

0/1 knapsack Problem:

In this problem we can't place fractional weights in the knapsack. Here either the item should be put into the bag completely or no item is to be put into the bag. Fraction of item can't be placed in the bag. Consider the weights $w_1, w_2, \ldots w_n$ and fraction of weights to be put into the bag should be $x_1, x_2, x_3, \ldots x_n$, $x_i = 0/1$. The dynamic programming solution for 0/1 knapsack problem is

$$\max [F_{n-1}(M), F_{n-1}(M-W_n) + P_n]$$

Scanned with CamScanner

sequence with regard to the state resulting from the first decision.

Scanned with CamScanner

0/1 knapsack Problem:

In this problem we can't place fractional weights in the knapsack. Here either the item should be put into the bag completely or no item is to be put into the bag. Fraction of item can't be placed in the bag. Consider the weights $w_1, w_2, \ldots w_n$ and fraction of weights to be put into the bag should be $x_1, x_2, x_3, \ldots x_n$, $x_i = 0/1$. The dynamic programming solution for 0/1 knapsack problem is

$$F_n(m) = \max [F_{n-1}(M), F_{n-1}(M-W_n) + P_n]$$

when $x_n = 1$, then size of the bag is reduced by W_n which is the weight of n^{th} item. As we are placing the n^{th} item we should the profit for the last item. Similarly we find for $F_{n-1}(M)$ and so on upto $F_1(M)$

If we try to remove the i^{th} item, then the profit of the i^{th} item reduced from the total profit and weight of i^{th} item from the total weight which belongs to profit and weight of i^{th} item

$S_i^0 = \{0,0\}$ // Initial condition

$S_1^i = S^{i-1} + (P_i, W_i)$ // addition operation

$S^i = S^{i-1} + S_1^{i-1}$ // merging operation

Eg: $n=3$, $(w_1, w_2, w_3) = (2,3,4)$

$(P_1, P_2, P_3) = (1,2,5)$

Sol: $S^0 = \{0,0\}$

$S_1^1 = S^{1-1} + (P_1, W_1)$

$= S^0 + (1,2)$

$= (0,0) + (1,2)$

$= (1,2)$

$S^1 = S^{1-1} + S_1^1$

$= S^0 + (1,2)$

$= (0,0) + (1,2)$

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

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

$S_1^i = S^i + (P_{i+1}, W_{i+1})$

$S^{i+1} = S^i + S_1^i$

Scanned with CamScanner

$i=2$:

$S_1^2 = S^{2-1} + (P_2, W_2)$

$= S^1 + (2,3)$

Scanned with CamScanner

$S_1^i = S^{i-1} + (P_i, W_i)$

$S^i = S^{i-1} + S_1^{i-1}$ // merging operation

Eg: $n=3$, $(w_1, w_2, w_3) = (2,3,4)$

$(P_1, P_2, P_3) = (1,2,5)$

Sol: $S^0 = \{0,0\}$

$S_1^1 = S^{1-1} + (P_1, W_1)$

$= S^0 + (1,2)$

$= (0,0) + (1,2)$

$= (1,2)$

$S^1 = S^{1-1} + S_1^1$

$= S^0 + (1,2)$

$= (0,0) + (1,2)$

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

$S^0 = \{0,0\}$

$S_1^i = S^i + (P_{i+1}, W_{i+1})$

$S^{i+1} = S^i + S_1^i$

Scanned with CamScanner

$i=2$:

$S_1^2 = S^{2-1} + (P_2, W_2)$

$= S^1 + (2,3)$

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

$= \{(2,3), (3,5)\}$

$S^2 = S^{2-1} + S_1^2$

$= S^1 + S_1^2$

$= \{(0,0), (1,2)\} + \{(2,3), (3,5)\}$

$= \{(0,0), (1,2), (2,3), (3,5)\}$

$i=3$:

$S_1^3 = S^{3-1} + (P_3, W_3)$

$= S^2 + (5,4)$

$= \{(0,0), (1,2), (2,3), (3,5)\} + (5,4)$

$= \{(5,4), (6,6), (7,7), (8,9)\}$

$S^3 = S^{3-1} + S_1^3$

$= S^2 + \{(5,4), (6,6), (7,7), (8,9)\}$

$= \{(0,0), (1,2), (2,3), (3,5)\} + \{(5,4), (6,6), (7,7), (8,9)\}$

$= \{(0,0), (1,2), (2,3), (3,5), (5,4), (6,6), (7,7), (8,9)\}$

71/107

Purging Rule/Dominance Rule:

If one of S^{i-1} and S_1^i has a pair (P_j, W_j) and the other has a pair (P_k, W_k) and $P_j \leq P_k$ and $W_j \geq W_k$ then the pair (P_j, W_j) is discarded

After Applying the purging rule

$S^3 = \{(0,0), (1,2), (2,3), (5,4), (6,6), (7,7), (8,9)\}$

Scanned with CamScanner

Now we will check the following condition in order to find the solution

If $(P_i, W_i) \in S^n$ & $(P_i, W_i) \notin S^{n-1}$ then

Scanned with CamScanner

Purging Rule/Dominance Rule:

If one of S^{i-1} and S_1^i has a pair (P_j, W_j) and the other has a pair (P_k, W_k) and $P_j \leq P_k$ and $W_j \geq W_k$ then the pair (P_j, W_j) is discarded

After Applying the purging rule

$S^3 = \{(0,0), (1,2), (2,3), (5,4), (6,6), (7,7), (8,9)\}$

Scanned with CamScanner

Now we will check the following condition in order to find the solution

If $(P_i, W_i) \in S^n$ & $(P_i, W_i) \notin S^{n-1}$ then

$X_n = 1$

Otherwise $X_n = 0$

If $(6,6) \in S^3$ &
$(6,6) \notin S^2$ → true

$X_3 = 1$

$(6,6) - (5,4) = (1,2) \in S^2$
$(1,2) \notin S^1$ → false

$X_2 = 0$

$(1,2) - (0,0) = (1,2) \in S^1$
$(1,2) \notin S^0$ → true

$X_1 = 1$

$\therefore X_1 = 1, X_2 = 0, X_3 = 1$

$P_i X_i = P_1 X_1 + P_2 X_2 + P_3 X_3$
$= 1*1 + 2*0 + 5*1$
$= 1 + 0 + 5$
$= 6$

Algorithm:

Algorithm DKP (P, W, n, M)

$S^0 \leftarrow \{(0,0)\}$

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

$S_1^i \leftarrow \{(P_i, W_i) / (P-P_i), (W-W_i) \in S^{i-1}$ & $W_i \leq M\}$

$S^i \leftarrow$ MERGE_PURGE (S^{i-1}, S_1^i)

repeat

$(P_x, W_x) \leftarrow$ last tuple in S^{n-1}

$(P_{...}, W_{...}) \leftarrow (P_i + P_n, W_i + W_n)$

Scanned with CamScanner

where W_i is the largest
W in any tuple in S^{n-1} such that
$W + W_n \leq M$

If $P_x > P_y$ then

Scanned with CamScanner

for i:=1 to n-1 do

$S_i^i \leftarrow \{(P_i, w_i) / (P-P_i), (w-w_i) \in S^{i-1} \text{ & } w_i \leq m\}$

$S^i \leftarrow$ MERGE_PURGE (S^{i-1}, S_i^i)

repeat

$(P_x, w_x) \leftarrow$ last tuple in S^{n-1}

$(P_{x+1}, w_{x+1}) \leftarrow (P_i + P_x, w_i + w_x)$

Scanned with CamScanner

Where w_i is the largest w in any tuple in S^{n-1} such that

$w + w_x \leq m$

If $P_x > P_y$ then

$x_n \leftarrow 0$;

else

$x_n \leftarrow 1$

endif

end 0/1 KP

All pairs shortest path problem:

When a weighted graph represented by its weight Matrix 'w' then the objective is to find the distance b/n every pair of node.

Step1: We will decompose the given problem into subproblems Let $A^k_{(i,j)}$ be the length of shortest path from node 'i' to node 'j' such that the label for intermediary node will be $\leq k$. We will compute A^k for $k=1,2,\ldots n$ for 'n' nodes

step 2: Any subpath of shortest path is a shortest path b/n the 'n' nodes. Divide the paths from i^{th} node to j^{th} node for every intermediary node, say 'k' then there arises 2 cases

Case(i): Path going from 'i' to 'j' through 'k'

Case(ii): Path which is not going through 'k'

Select only the shortest path from these two cases

Step 3: The shortest path can be computed using bottom-up Computation method, here follows recursive method

Initially

$A^0 = w[i,j]$

Next Computations

Scanned with CamScanner

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

Eg:

Calculate all pairs shortest path for the following graph

Scanned with CamScanner

arises & cases

Step 3: The shortest path can be computed using bottom-up computation method, here follows recursive method

Initially

$A^0 = W[i,j]$

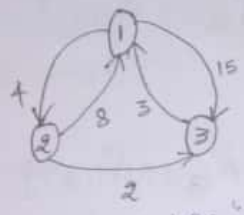
Next Computations

Scanned with CamScanner

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

Eg:

Calculate all pairs shortest path for the following graph



Sol: Initially the weight matrix 'W' is

$$A^0 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{ccc} 1 & 2 & 3 \end{array} \\ \begin{bmatrix} 0 & 4 & 15 \\ 8 & 0 & 2 \\ 3 & \infty & 0 \end{bmatrix} \end{array}$$

To generate the next matrices the equation is

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

Now $k=1$

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

$= \min\{0, 0\}$

$= 0$

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

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

$= 4$

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

$= \min\{15, 0+15\}$

$= 15$

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

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

$= 8$

$A^1_{(2,2)} = \min\{A^0_{(2,2)}, A^0_{(2,1)} + A^0_{(1,2)}\} = \min\{0, 8+4\}$

$= 0$

74/107

Scanned with CamScanner

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

$= \min\{2, 3+15\}$

Scanned with CamScanner

ere

$A^1_{(1,2)} = \min\{A^0_{(1,2)}, A^0_{(2,1)} + A^0_{(1,2)}\}$
$= \min\{4, 8+4\}$
$= 4$

$A^1_{(1,3)} = \min\{A^0_{(1,3)}, A^0_{(2,1)} + A^0_{(1,3)}\}$
$= \min\{15, 8+15\}$
$= 15$

ttom-up

$A^1_{(2,1)} = \min\{A^0_{(2,1)}, A^0_{(2,1)} + A^0_{(1,1)}\}$
$= \min\{8, 8+0\}$
$= 8$

$A^1_{(2,2)} = \min\{A^0_{(2,2)}, A^0_{(2,1)} + A^0_{(1,2)}\} = \min\{0, 8+4\}$
$= 0$

Scanned with CamScanner

$A^1_{(2,3)} = \min\{A^0_{(2,3)}, A^0_{(3,1)} + A^0_{(1,3)}\}$
$= \min\{7, 3+15\}$
$= 7$

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

$A^1_{(3,2)} = \min\{A^0_{(3,2)}, A^0_{(3,1)} + A^0_{(1,2)}\}$
$= \min\{6, 3+4\}$
$= 7$

$A^1_{(3,3)} = \min\{A^0_{(3,3)}, A^0_{(3,1)} + A^0_{(1,3)}\}$
$= \min\{0, 3+15\}$
$= 0$

$$A^1 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{ccc} 1 & 2 & 3 \end{array} \\ \begin{bmatrix} 0 & 4 & 15 \\ 8 & 0 & 7 \\ 3 & 7 & 0 \end{bmatrix} \end{array}$$

k=2:

$A^2_{(1,1)} = \min\{A^1_{(1,1)}, A^1_{(1,2)} + A^1_{(2,1)}\}$
$= \min\{0, 4+8\}$
$= 0$

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

$A^2_{(1,3)} = \min\{A^1_{(1,3)}, A^1_{(1,2)} + A^1_{(2,3)}\}$
$= \min\{15, 4+7\}$
$= 11$

$A^2_{(2,1)} = \min\{A^1_{(2,1)}, A^1_{(2,2)} + A^1_{(2,1)}\}$
$= \min\{8, 0+8\}$
$= 8$

$A^2_{(2,2)} = 0$

75/107

Scanned with CamScanner

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

Scanned with CamScanner

$A^2_{(1,1)} = \min\{A^1_{(1,1)}, A^1_{(1,2)} + A^1_{(2,1)}\}$
$= \min\{0, 4+8\}$
$= 0$

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

$A^2_{(1,3)} = \min\{A^1_{(1,3)}, A^1_{(1,2)} + A^1_{(2,3)}\}$
$= \min\{15, 4+2\}$
$= 6$

$A^2_{(2,1)} = \min\{A^1_{(2,1)}, A^1_{(2,2)} + A^1_{(2,1)}\}$
$= \min\{8, 0+8\}$
$= 8$

$A^2_{(2,2)} = 0$

Scanned with CamScanner

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

$A^2_{(3,1)} = \min\{A^1_{(3,1)}, A^1_{(3,2)} + A^1_{(2,1)}\}$
$= \min\{3, 7+8\}$
$= 3$

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

$$A^2 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{ccc} 1 & 2 & 3 \end{array} \\ \begin{bmatrix} 0 & 4 & 6 \\ 8 & 0 & 2 \\ 3 & 7 & 0 \end{bmatrix} \end{array}$$

K=3:

$A^3_{(2,1)} = \min\{A^2_{(2,1)}, A^2_{(2,3)} + A_{(3,1)}\}$
$= \min\{8, 2+3\}$
$= 5$

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

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

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

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

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

76/107

Scanned with CamScanner

Scanned with CamScanner

$= \min\{4, 6+7\}$

$= 4$

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

$= \min\{6, 6+0\}$

$= 6$

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

$= \min\{2, 2+0\}$

$= 2$

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

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

$= 3$

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

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

Scanned with CamScanner

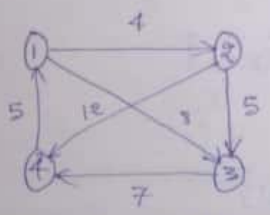
$$A^3 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{ccc} 1 & 2 & 3 \end{array} \\ \begin{bmatrix} 0 & 4 & 6 \\ 5 & 0 & 2 \\ 3 & 7 & 0 \end{bmatrix} \end{array}$$

A^3 gives shortest distance between any pair of vertices

Algorithm:

```
Algorithm Allpairs (cost, A, 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]);
}
```

2)



Initially the weight matrix is

$$A^0 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 4 & 3 & \infty \\ \infty & 0 & 5 & 12 \\ \infty & \infty & 0 & 7 \\ 5 & \infty & \infty & 0 \end{bmatrix} \end{array}$$

To generate the next matrices the equation is

$A^k(i,j) = \min$

77/107

Scanned with CamScanner

Multistage Graphs:

A graph $G = (V, E)$ is a directed graph in

Scanned with CamScanner

2	4	0	5	12
3	6	2	0	7
4	5	8	4	0

To generate the next matrices the equation is

$A^k(i,j) = \min$

Scanned with CamScanner

Multistage Graphs:

A multistage graph $G=(V,E)$ is a directed graph in which the vertices are partitioned into $k \geq 2$ disjoint sets V_i, $1 \leq i \leq k$. In addition, if $\langle u,v \rangle$ is an edge in E, then $u \in V_i$ and $v \in V_{i+1}$ for some i, $1 \leq i < k$.

Let the vertex 's' is the source and 't' the sink. Let $c(i,j)$ be the cost of edge $\langle i,j \rangle$. 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'. Each set V_i defines a stage in the graph. Because of the constraints on E, every path from 's' to 't' starts in stage 1, goes to stage 2, then to stage 3, then to stage 4, and so on, and eventually terminates in stage k.

A dynamic programming formulation for k-stage graph problem is obtained by first noticing that every s-to-t path is the result of a sequence of k-2 decisions. The i^{th} decision involves determining which vertex in V_{i+1}, $1 \leq i \leq k-2$, is to be on the path. Let $c(i,j)$ be on the cost of the path from source to destination

To solve the multistage graph we follow 2 approaches

1) forward approach
2) Backward approach

By using forward approach we obtain

$$\text{Cost}(i,j) = \min_{l \in V_{i+1}, \langle j,l \rangle \in E} \{c(j,l) + \text{cost}(i+1,l)\}$$

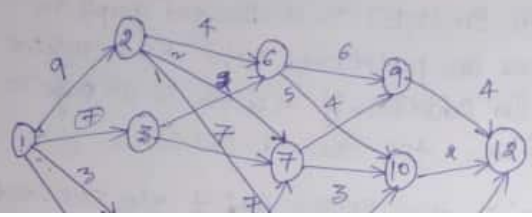
By using backward approach we obtain

$$\text{Bcost}(i,j) = \min_{l \in V_{i-1}} \{\text{Bcost}(i-1,l) + c(l,j)\}$$

78/107

Scanned with CamScanner

Example:

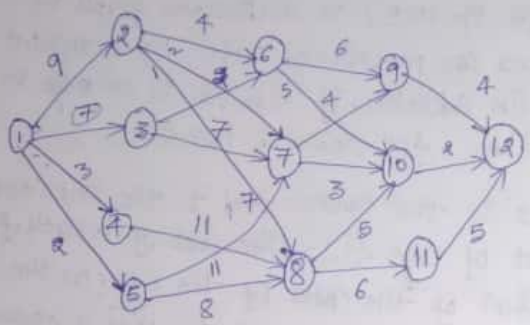


Scanned with CamScanner

$l \in V_{i-1}$

Scanned with CamScanner

Example:



Using Forward approach:-

We use the following equation to find the minimum cost from s to t

$$Cost(i,j) = \min_{l \in V_{i+1}, <j,l> \in E} \{c(j,l) + cost(i+1,l)\}$$

Node 1: $i=1, j=1$

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

$= \min\{9 + cost(2,2), 7 + cost(2,3), 3 + cost(2,4), 2 + cost(2,5)\}$

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

$= \min\{4 + cost(3,6), 2 + cost(3,7), 1 + cost(3,8)\}$

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

$= \min\{6 + cost(4,9), 5 + cost(4,10)\}$

$Cost(4,9) = \min\{c(9,12) + cost(5,12)\} = \min\{4+0\} = 4$

$Cost(4,10) = \min\{c(10,12) + cost(5,12)\} = \min\{2+0\} = 2$

Scanned

$Cost(3,6) = \min\{6+4, 5+2\} = 7$

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

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

$= \min\{4+4, 3+2\} = 5$

$Cost(4,7) = \min\{c(7,9) + cost(5,9), c(7,10) + cost(5,10)\}$

$= \min\{4 + cost(5,9)\}$

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

$= \min\{5 + cost(4,10), 6 + cost(4,11)\}$

$= \min\{5+2, 6 + cost(4,11)\}$

$Cost(4,11) = \min\{c(11,12) + cost(5,12)\}$

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

Scanned with CamScanner

$Cost(3,6) = \min\{c(6,9) + Cost(4,9), c(6,10) + Cost(4,10)\}$
$= \min\{6 + Cost(4,9), 5 + Cost(4,10)\}$

$Cost(4,9) = \min\{c(9,12) + Cost(5,12)\} = \min\{4+0\} = 4$

$Cost(4,10) = \min\{c(10,12) + Cost(5,12)\} = \min\{2+0\} = 2$

Scanned with CamScanner

$Cost(3,6) = \min\{6+4, 5+2\} = 7$

$Cost(3,7) = \min\{c(7,9) + Cost(4,9), c(7,10) + Cost(4,10)\}$
$= \min\{4 + Cost(4,9), 5 + Cost(4,10)\}$
$= \min\{4+4, 3+2\} = 5$

$Cost(4,7) = \min\{c(7,9) + Cost(5,9), c(7,10) + Cost(5,10)\}$
$= \min\{4 + Cost(5,9)\}$

$Cost(3,8) = \min\{c(8,10) + Cost(4,10), c(8,11) + Cost(4,11)\}$
$= \min\{5 + Cost(4,10), 6 + Cost(4,11)\}$
$= \min\{5+2, 6 + Cost(4,11)\}$

$Cost(4,11) = \min\{c(11,12) + Cost(5,12)\}$
$= \min\{5+0\}$
$= 5$

$Cost(3,8) = \min\{7, 6+5\}$
$= 7$

$Cost(2,2) = \min\{4+7, 2+5, 1+7\}$
$= \min\{11, 7, 8\}$
$= 7$

$Cost(1,1) = \min\{9+7, 2 + Cost(2,3), 3 + Cost(2,4), 2 + Cost(2,5)\}$

the path ① — ⑧ — ⑦ — ⑩ — ⑫

(or)

① — ⑤ — ⑥ — ⑩ — ⑫

The minimum cost path is 16

Using Backward approach:

We use the following equation to find the minimum cost from s to t

$Cost(i,j) = \min\{BCost(i-1,l) + c(l,j)\}$
$l \in V_{i-1}$
$\langle l,j \rangle \in E$

Node 9 : $i=5, j=12$

$Cost(5,12) = \min\{BCost(4,9) + c(9,12), BCost(4,10) + c(10,12), BCost(4,11) + c(11,12)\}$

Scanned with CamScanner

$= \min\{BCost(4,9) + 4, BCost(4,10) + 2, BCost(4,11) + 5\}$

$BCost(4,9) = \min\{BCost(3,6) + c(6,9), BCost(3,7) + c(7,9)\}$
$= \min\{BCost(3,6) + 6, BCost(3,7) + 4\}$

$BCost(3,6) = \min\{BCost(2,2) + c(2,6), BCost(2,3) + c(3,6)\}$
$= \min\{BCost(2,2) + 4, BCost(2,3) + 2\}$

$BCost(2,2) = \min\{BCost(1,1) + c(1,2)\} = \min\{0+9\} = 9$

Scanned with CamScanner

Cost (i,j) = min {BCost (i-1, l) ...}
$l \in V_{i-1}$ i starts from 2 to ...
$<l,j> \in E$

Node 1 : i=5, j=12
Cost (5,12) = min {BCost (4,9) + c(9,12), BCost(4,10) + c(10,12), BCost (4,11) + c(11,12)}

Scanned with CamScanner

= min {BCost(4,9) + 4, BCost(4,10) + 2, BCost(4,11) + 5}

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

BCost (3,6) = min {BCost (2,2) + c(2,6), BCost(2,3) + c(3,6)}
= min {BCost (2,2) + 4, BCost (2,3) + 2}

BCost (2,2) = min {BCost (1,1) + c(1,2)} = min{0+9} = 9

BCost (2,3) = min {BCost (1,1) + c(1,3)} = min {0+7} = 7

BCost (3,6) = min {9+4, 7+2} = min {13,9} = 9

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

BCost (2,5) = min {BCost(1,1) + c(1,5)} = 2

BCost (3,7) = min {9+2, 7+7, 2+11} = min {11,14,13} = 11

BCost (4,9) = min {9+6, 11+4} = min {15,15} = 15

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

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

BCost (2,4) = min {BCost (1,1) + c(1,4)} = 3

BCost (3,8) = min {9+1, 3+11, 2+8} = min {10,14,10} = 10

BCost (4,10) = min {9+5, 11+3, 10+5} = min {14,14,15} = 14

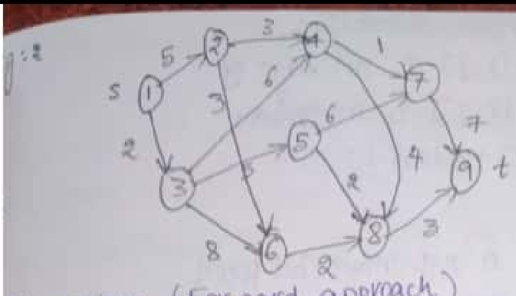
BCost (4,11) = min {BCost (3,8) + c(8,11)}

BCost (5,12) = min {15+4, 14+2, 16+5} = min {19,16,21} = 16

81/107

Scanned with CamScanner

g:2



Algorithm:- (Forward approach)

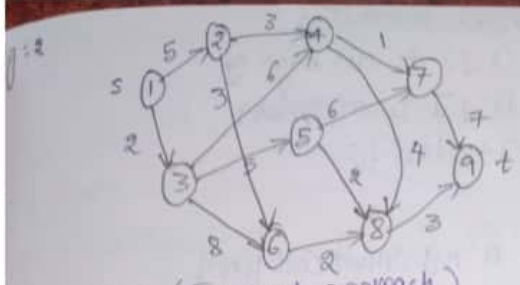
Algorithm Fgraph (G,k,n,p)

// The is a input is a k-stage graph G=(V,E) with n

Scanned with CamScanner

Bcost (5,12) = min {5+4, 14+2, 16+5} = min {9, 16, 21}

Scanned with CamScanner



Algorithm:- (Forward approach)

```
Algorithm Fgraph (G, k, n, p)
// G is a input is a k-stage graph G=(V,E) with n
   vertices
// indexed in order or 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];
     d[j] := r;
  }
  p[1] := 1; p[k] := n;  // Find a minimum cost path
  for j := 2 to k-1 do p[j] := d[p[j-1]];
}
```

Algorithm for backward approach:-

```
Algorithm Bgraph (G, k, n, p)
// same function as Fgraph
{
  Bcost[1] := 0.0;
  for j := 2 to n do
```

82/107

Scanned with CamScanner

```
  {                    // 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]];
```

Scanned with CamScanner

}

Algorithm for backward approach:-

Algorithm Bgraph (G, k, n, p)

// same function as Fgraph

{

Bcost[1] := 0.0;

for j := 2 to n do

Scanned with CamScanner

{ //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]];

}

Complexity Analysis:

The complexity analysis of the algorithm is fairly straightforward. Here, if G has |E| edges, then the time for the first for loop is θ V E

Traveling Sales Person Problem:-

In this problem the sales person should start at a starting point and travel all the places and come back to the starting point. In this problem we need to minimise the traveling cost

Eg: Suppose we have to route a postal van to pick up the mails from the mail boxes located at 'n' different sites. n+1 vertex graph may be used to represent the situation, one vertex represents the post office from which the postal van starts and to which it must return. The route taken by the postal van should have minimum length (or) cost. To solve this problem the following equations are used.

1) $g(i, \phi) = C_{i1}, 1 \le i \le n$

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

83/107

Scanned with CamScanner

min is considered as the intermediate Node.
j∈s

$g(j, s-\{j\})$ means 'j' is already traversed so next we have to traverse $s-\{j\}$ with 'j' as starting point.

Eg: 1) Consider the following graph to solve travelling sales person problem

	1	2	3	4
	[0	10	15	20

Scanned with CamScanner

minimise the traveling cost.

Eg: Suppose we have to route a postal van to pick up

situation, one vertex represents the post office from

which the postal van starts and to which it must return.

The route taken by the postal van should have minimum

length (or) cost. To solve this problem the following equations

are used.

1) $g(i,\phi) = C_{i1}, 1 \le i \le n$

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

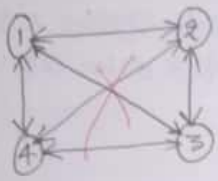
Scanned with CamScanner

min is considered as the intermediate Node.

$j \in s$

$\{g(j, s-\{j\})\}$ means 'j' is already traversed so next we have to traverse $s-\{j\}$ with 'j' as starting point.

Eg: 1) Consider the following graph to solve travelling sales person problem



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

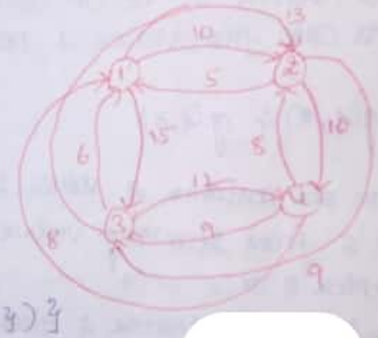
Sol: $g(i,\phi) = C_{i1}, 1 \le i \le n$

$g(1,\phi) = C_{11} = 0$

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

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

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



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

84/107

$|s| = 1$

set contains only 1 element and here we make the assumption that the sales person starts at vertex 1 from that vertex, he can move to vertex '2' if, he visits vertex '2' from that vertex he can visit next either vertex '3' or vertex '4' since $g(2,3)$ & $g(3,4)$

$g(2,3) = \min_{j \in s} \{C_{23} + g(3,\phi)\} = \min_{j \in s} \{9+6\} = 15$

$g(2,4) = \min_{j \in s} \{C_{24} + g(4,\phi)\} = \min_{j \in s} \{10+8\} = 18$

$g(3,2) = \min_{j \in s} \{C_{32} + g(2,\phi)\} = \min_{j \in s} \{13+5\} = 18$

Scanned with CamScanner

Scanned with CamScanner

$g(2,4) = \min_{j \in S} \{C_{24} + g(4,\phi)\} = \min_{j \in S} \{10+8\} = 18$

$g(3,2) = \min_{j \in S} \{C_{32} + g(2,\phi)\} = \min_{j \in S} \{13+5\} = 18$

Scanned with CamScanner

$g(3,4) = \min_{j \in S} \{C_{34} + g(4,\phi)\} = \min_{j \in S} \{12+8\} = 20$

$g(4,2) = \min_{j \in S} \{C_{42} + g(2,\phi)\} = \min_{j \in S} \{8+5\} = 13$

$g(4,3) = \min_{j \in S} \{C_{43} + g(3,\phi)\} = \min_{j \in S} \{9+6\} = 15$

$|S| = 2$

From vertex '1' next he can visit vertex '3' instead of vertex '2' in this case from vertex '3' he can visit either vertex '2' or '4'

$\therefore g(3,2)$ & $g(3,4)$

From vertex '1' he can visit vertex '4' instead of '3'. In this case from vertex '4' next he can visit either '2' or 4

$\therefore g(4,2)$ & $g(4,3)$

Here set contains 2 values. So we can place two vertices in 'S' from starting vertex '1'. Next he can visit either vertex '2' or '3' or '4'

If he visits vertex '2' then from that vertex he can visit either vertex '3' or vertex '4'. So $g(2,\{3,4\})$

If he visits vertex '3' from that vertex he can visit either '2' or '4'. So $g(3,\{2,4\})$

If he visits vertex '4' from that vertex he can visit either vertex '2' (or) '3'. So $g(4,\{2,3\})$.

$g(2,\{3,4\}) = \min_{j \in S} \{C_{23} + g(3,4), C_{24} + g(4,3)\}$

$= \min_{j \in S} \{9+20, 10+15\} = 25$

$g(3,\{2,4\}) = \min_{j \in S} \{C_{32} + g(2,4), C_{34} + g(4,2)\}$

Scanned with

$= \min_{j \in S} \{13+18, 12+13\} = 25$

$g(4,\{2,3\}) = \min_{j \in S} \{C_{42} + g(2,3), C_{43} + g(3,2)\}$

$= \min_{j \in S} \{8+15, 9+18\}$

$= 23$

$|S| = 2$

Now the set contains 2 elements after starting from vertex 1, next he can visit either vertex '2' or '3' or '4'. So determine

Scanned with CamScanner

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

$$= \min_{j \in \{3,4\}} \{9+20, 10+15\} = 25$$

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

Scanned with CamScanner

$$= \min_{j \in \{2,4\}} \{13+18, 12+13\} = 25$$

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

$$= \min_{j \in \{2,3\}} \{8+15, 9+18\}$$

$$= 23$$

$|s| = 3$

Now the set contains 3 elements after starting from vertex 1, next he can visit either vertex '2' or '3' or '4'. so determine

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

$$= \min_{j \in \{2,3,4\}} \{10+25, 15+25, 20+23\}$$

$$= \min_{j \in \{2,3,4\}} \{35, 40, 43\}$$

$$= 35$$

$1 \rightarrow 2 \rightarrow 4 \rightarrow 3 \rightarrow 1 = 35$

Optimal binary search trees:-

Let us assume that the given set of identifiers is $\{a_1, \ldots a_n\}$ with $a_1 < a_2 < \ldots < a_n$. Let $p(i)$ be the probability with which we search for a_i. Let $q(i)$ be the probability that the identifier x being searched for is such that $a_i < x < a_{i+1}$, $0 \le i \le n$ (assume $a_0 = -\infty$ and $a_{n+1} = +\infty$). We have to arrange the identifiers in a binary search tree in a way that minimizes the expected total access time.

In a binary search tree, the number of comparisons

86/107

Scanned with CamScanner

needed to access an element at depth d is $d+1$, so if a_i is placed at depth d_i, then we want to minimize $\sum_{i=1}^{n} p_i (1+d_i)$.

Let $p(i)$ be the probability with which we shall be searching for a_i. Let $q(i)$ be the probability of an un-successful search. Every internal node represents a point where a un-successful search may terminate.

The expected cost contribution for the internal node for

Scanned with CamScanner

and $a_{n+1} = +\infty$). We have to arrange the ... in a binary search tree in a way that minimizes the expected total access time.

In a binary search tree, the number of comparisons

Scanned with CamScanner

needed to access an element at depth 'd' is d+1. If 'a_i' is placed at depth d_i, then we want to minimize $\sum_{i=1}^{n} p_i(1+d_i)$

Let P(i) be the probability with which we shall be searching for 'a_i'. Let Q(i) be the probability of an un-successful search. Every internal node represents a point where a un-successful search may terminate

The expected cost contribution for the internal node for 'a_i' is:

$$P(i) * \text{level}(a_i)$$

Unsuccessful search terminate with I=0. Hence the cost contribution for this node is:

$$Q(i) * \text{level}((E_i) - 1)$$

The expected cost of binary search tree is:

$$\sum_{i=1}^{n} P(i) * \text{level}(a_i) + \sum_{i=0}^{n} Q(i) * (\text{level}((E_i) - 1)$$

Given a fixed set of identifiers, we wish to create a binary search tree organization. We may expect different binary search trees for the same identifier set to have different performance characteristics.

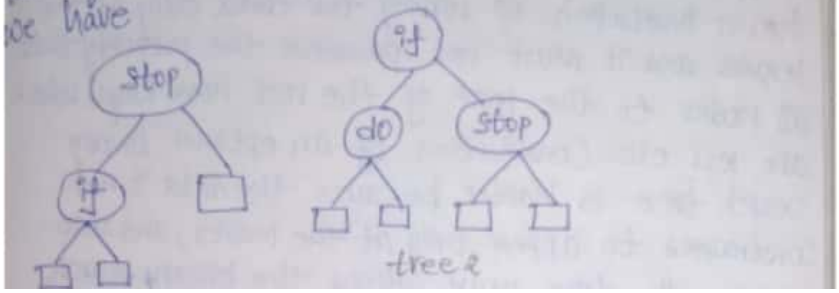
The computation of each of these C(i,j)'s requires us to find the minimum of 'm' quantities. Hence each such C(i,j) can be computed in time O(m). The total time for all C(i,j)'s with j-i=m is therefore $O(nm - m^2)$

The total time to evaluate all the C(i,j)'s and r(i,j)'s is therefore:

$$\sum (nm - m^2) = O(n^3)$$

Scanned with Camscanner

Eg-1): The possible binary search tree for the identifier set $(a_1, a_2, a_3) = (\text{do}, \text{if}, \text{stop})$ are as follows. Given the equal probabilities $P(i) = Q(i) = 1/7$ for all i, we have



tree 2

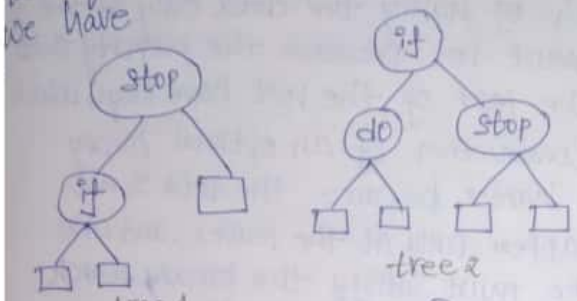
Scanned with CamScanner

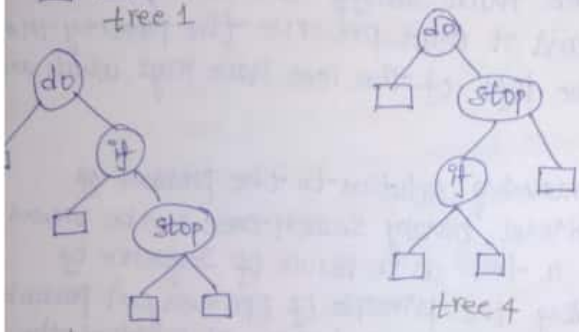
is therefore:

$$\sum_{1 \le m \le n} (nm - m^2) = O(n^3)$$

Scanned with CamScanner

Eg-1): The possible binary search tree for the identifier set $(a_1, a_2, a_3) = (do, if, stop)$ are as follows. Given the equal probabilities $P(i) = Q(i) = 1/7$ for all i, we have





$$\text{Cost (tree\#1)} = \left(\frac{1}{7}\times1 + \frac{1}{7}\times2 + \frac{1}{7}\times3\right) + \left(\frac{1}{7}\times1 + \frac{1}{7}\times2 + \frac{1}{7}\times3 + \frac{1}{7}\times3\right)$$

$$= \frac{1+2+3}{7} + \frac{1+2+3+3}{7} = \frac{6+9}{7} = \frac{15}{9}$$

$$\text{Cost (tree\#2)} = \left(\frac{1}{7}\times1 + \frac{1}{7}\times2 + \frac{1}{7}\times2\right) + \left(\frac{1}{7}\times2 + \frac{1}{7}\times2 + \frac{1}{7}\times2 + \frac{1}{7}\times2\right)$$

$$= \frac{1+2+2}{7} + \frac{2+2+2+2}{7} = \frac{5+8}{7} = \frac{13}{7}$$

$$\text{Cost (tree\#3)} = \left(\frac{1}{7}\times1 + \frac{1}{7}\times2 + \frac{1}{7}\times3\right) + \left(\frac{1}{7}\times1 + \frac{1}{7}\times2 + \frac{1}{7}\times3 + \frac{1}{7}\times3\right)$$

$$= \frac{1+2+3}{7} + \frac{1+2+3+3}{7} = \frac{6+9}{7} = \frac{15}{7}$$

$$\text{Cost (tree\#4)} = \left(\frac{1}{7}\times1 + \frac{1}{7}\times2 + \frac{1}{7}\times3\right) + \left(\frac{1}{7}\times1 + \frac{1}{7}\times2 + \frac{1}{7}\times3 + \frac{1}{7}\times3\right)$$

Scanned with C

$= \frac{1+2+3}{7} + \frac{1+2+3+3}{7} = \frac{15}{7}$

Huffman Coding tree solved by a greedy algorithm has a limitation of having the data only at the leaves and it must not preserve the property that all nodes to the left of the root have keys; values are less etc Construction of an optimal binary search tree is harder, because the data is not constrained to appear only at the leaves, and also because the tree must satisfy the binary search tree property and it must preserve the property that

Scanned with CamScanner

Cost (tree #3) = $\left(\frac{1}{7}\times1+\frac{1}{7}\times2+\frac{1}{7}\times3\right)+\left(\frac{1}{7}\times1+\frac{1}{7}\times2+\frac{1}{7}\times3+\frac{1}{7}\times3\right)$

$= \frac{1+2+3}{7} + \frac{1+2+3+3}{7} = \frac{6+9}{7} = \frac{15}{7}$

Cost (tree #4) = $\left(\frac{1}{7}\times1+\frac{1}{7}\times2+\frac{1}{7}\times3\right)+\left(\frac{1}{7}\times1+\frac{1}{7}\times2+\frac{1}{7}\times3+\frac{1}{7}\times3\right)$

Scanned with CamScanner

$= \frac{1+2+3}{7} + \frac{1+2+3+3}{7} = \frac{15}{7}$

Huffman Coding tree solved by a greedy algorithm has a limitation of having the data only at the leaves and it must not preserve the property that all nodes to the left of the root have keys, which are less etc. Construction of an optimal binary search tree is harder, because the data is not constrained to appear only at the leaves, and also because the tree must satisfy the binary search tree property and it must preserve the property that all nodes to the left of the root have keys, which are less.

A dynamic programming solution to the problem of obtaining an optimal binary search tree can be viewed by constructing a tree as a result of sequence of decisions by holding the principle of optimality. A possible approach to this is to make a decision as to which of the a_i's be arraigned to the root node at 'T'. If we choose a_k then it is clear that the central internal nodes for $a_1, a_2 \cdots a_{k-1}$ as well as the external nodes for the classes $E_0, E_1, \cdots E_{k-1}$ will lie in the left subtree, L, of the root. The remaining nodes will be in the right subtree, R. The structure of an optimal binary search tree is:

Cost (L) = $\sum_{i=1}^{k} P(i) * \text{level}(a_i) + \sum_{i=0}^{k} Q(i) *$

89/107

Scanned with CamScanner

Cost (R) = $\sum_{i=k}^{n} P(i) * \text{level}(a_i) + \sum_{i=k}^{n} Q(i) * (\text{level}(E_i) - 1)$

the $C(i,j)$ can be computed as

$$C(i,j) = \min_{i<k\le j} \{C(i,k-1) + C(k,j) + P(k) + W(i,k-1) + W(k,j)\}$$

$$= \min_{i<k\le j} \{C(i,k-1) + C(k,j)\} + W(i,j) \longrightarrow (1)$$

Scanned with CamScanner

$$\text{Cost}(T) = \sum_{i=1}^{k} P(i) * \text{level}(a_i) + \sum_{i=0}^{k} Q(i) * (\text{level}(E_i) - 1)$$

Scanned with CamScanner

$$\text{Cost}(T) = \sum_{i=1}^{n} P(i) * \text{level}(a_i) + \sum_{i=0}^{n} Q(i) * (\text{level}(E_i) - 1)$$

The $C(i,j)$ can be computed as

$$C(i,j) = \min_{i<k\le j} \{C(i,k-1) + C(k,j)\} + P(k) + W(i,k-1) + W(k,j)\}$$

$$= \min_{i<k\le j} \{C(i,k-1) + C(k,j)\} + W(i,j) \longrightarrow (1)$$

where $W(i,j) = P(j) + Q(j) + W(i,j-1) \longrightarrow (2)$

Initially $C(i,i) = 0$ and $W(i,i) = Q(i)$ for $0 \le i < n$

Equation (1) may be solved for $C(0,n)$ by first computing all $C(i,j)$ such that $j-i=1$. Next, we can compute all $C(i,j)$ such that $j-i=2$, then all $C(i,j)$ with $j-i=3$, and so on $C(i,j)$ is the cost of the optimal binary search tree T_{ij} during computation we record the root $R(i,j)$ of each tree T_{ij}. Then an optimal binary search tree may be constructed from these $R(i,j)$. $R(i,j)$ is the value of 'k' that minimizes equation (1)

We solve the problem by knowing $W(i,i+1)$, $C(i,i+1)$ and $R(i,i+1)$, $0 \le i < $ knowing $W(i,i+2)$, $C(i,i+2)$ and $R(i,i+2)$, $0 \le i < 3$ and repeating until $W(0,n)$, $C(0,n)$ and $R(0,n)$ are obtained.

The results are tabulated to recover the actual tree

Example 1: Let $n=4$ and $(a_1, a_2, a_3, a_4) = (\text{do, if, int, while})$. Let $P(1:4) = (3,3,1,1)$ and $Q(0:4) = (2,3,1,1,1)$

Solution:

90/107

Scanned with CamScanner

Table for recording $W(i,j)$, $C(i,j)$ and $R(i,j)$

This computation is carried out row-wise from row 0 to row 4.

Initially, $W(i,i) = Q(i)$ and $C(i,i) = 0$ and $R(i,i) = 0$, $0 \le i < 4$

Solving for $C(0,n)$:

First, computing all $C(i,j)$ such that $j-i=1; j=i+1$ and $a \le i < 4$; $i=0,1,2,3$; $i < k \le j$. start with $i=0$; so $j=1$;

Scanned with CamScanner

$R(i,i+1)$; $0 \le i < 4$

$C(0,n)$ and $R(0,n)$ are obtained.

The results are tabulated to recover the actual tree

Example 1: Let $n=4$ and $(a_1, a_2, a_3, a_4) = (do, if, int, while)$. Let $P(1:4) = (3,3,1,1)$ and $Q(0:4) = (2,3,1,1,1)$

Solution:

Scanned with CamScanner

Table for recording $W(i,j)$, $C(i,j)$ and $R(i,j)$

This computation is carried out row-wise from row 0 to row 4.

Initially, $W(i,i) = Q(i)$ and $C(i,i) = 0$ and $R(i,i) = 0$,

$0 \le i < 4$

Solving for $C(0,n)$:

First, Computing all $C(i,j)$ such that $j-i=1$; $j=i+1$ and as $0 \le i < 4$; $i = 0,1,2,3$; $i < k \le j$. Start with $i=0$; so $j=1$; as $i < k \le j$, so the possible value for $k=1$

$W(0,1) = P(1) + Q(1) + W(0,0) = 3+3+2 = 8$

$C(0,1) = W(0,1) + \min\{C(0,0) + C(1,1)\} = 8$

$R(0,1) = 1$ (value of 'k' that is minimum in the above equation)

Next with $i=1$; so $j=2$; as $i < k \le j$, so the possible value for $k=2$

$W(1,2) = P(2) + Q(2) + W(1,1) = 3+1+3 = 7$

$C(1,2) = W(1,2) + \min\{C(1,1) + C(2,2)\} = 7$

$R(1,2) = 2$

Next with $i=2$; so $j=3$; as $i < k \le j$, so the possible value for $k=3$

$W(2,3) = P(3) + Q(3) + W(2,2) = 1+1+1 = 3$

$C(2,3) = W(2,3) + \min\{C(2,2) + C(3,3)\} = 3 + [0+0] = 3$

Next with $i=3$; so $j=4$; as $i < k \le j$, so the possible value for $k=4$

$W(3,4) = P(4) + Q(4) + W(3,3) = 1+1+1 = 3$

$C(3,4) = W(3,4) + \min\{C(3,3) + C(4,4)\} = [0+0]$

$R(3,4) = 4$

Second, Computing all $C(i,j)$ such that $j-i=2$; $j=i+2$ and as $0 \le i < 3$; $i = 0,1,2$; $i < k \le j$. Start with $i=0$; so $j=2$; as $i < k \le j$, so the possible values for $k=1$ and 2

Scanned with CamScanner

$W(0,2) = P(2) + Q(2) + W(0,1) = 3+1+8 = 12$

$C(0,2) = W(0,2) + \min\{C(0,0) + C(1,2), C(0,1) + C(2,2)\}$

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

$= 19$

Scanned with CamScanner

$W(3,4) = P(4) + Q(4) + W(3,3) = 1+1+1 = 3$

$C(3,4) = W(3,4) + \min\{[C(3,3) + C(4,4)]\} = 3 + [(0+0)] = 3$

$R(3,4) = 4$

Second, computing all $C(i,j)$ such that $j-i=2$; $j=i+2$ and as $0 \le i < 3$; $i = 0,1,2$; $i < k \le j$. Start with $i=0$; so $j=2$; as $i < k \le j$, so the possible values for $k = 1$ and 2

Scanned with CamScanner

$W(0,2) = P(2) + Q(2) + W(0,1) = 5+1+8 = 12$

$C(0,2) = W(0,2) + \min\{[C(0,0) + C(1,2), C(0,1) + C(2,2)]\}$

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

$= 19$

$R(0,2) = 1$

Next, with $i=1$; so $j=3$; as $i < k \le j$, so the possible value for $k = 2$ and 3

$W(1,3) = P(3) + Q(3) + W(1,2) = 1+1+7 = 9$

$C(1,3) = W(1,3) + \min\{[C(1,1) + C(2,3)], [C(1,2) + C(3,3)]\}$

$= W(1,3) + \min\{(0+3), (7+0)\}$

$= 9+3$

$= 12$

$R(1,3) = 2$

Next, with $i=2$; so $j=4$; as $i < k \le j$, so the possible values for $k = 3$ and 4

$W(2,4) = P(4) + Q(4) + Q(2,3) = 1+1+3 = 5$

$C(2,4) = W(2,4) + \min\{[C(2,2) + C(3,4)], [C(2,3) + C(4,4)]\}$

$= 5 + \min\{(0+3), (3+0)\} = 5+3$

$= 8$

$R(2,4) = 3$

Third, Computing all $C(i,j)$ such that $j-i=3$; $j=i+3$ and as $0 \le i < 2$; $i = 0,1$; $i < k \le j$. Start with $i=0$; so $j=3$; as $i < k \le j$; so the possible values for $k = 1, 2$ and 3

$W(0,3) = P(3) + Q(3) + W(0,2) = 1+1+12 = 14$

$C(0,3) = W(0,3) + \min\{[C(0,0) + C(1,3)], [C(0,1) + C(2,3)]$

$, [C(0,2) + C(3,3)]\}$

$= 14 + \min\{(0+12), (8+3), (19+0)\} = 14+11 = 25$

$R(0,3) = 2$

92/107

Scanned with CamScanner

Start with $i=1$; so $j=4$; as $i < k \le j$, so the possible values for $k = 2, 3$ and 4

$W(1,4) = P(4) + Q(4) + W(1,3) = 1+1+9 = 11$

$C(1,4) = W(1,4) + \min\{[C(1,1) + C(2,4)], [C(1,2) + C(3,4)]$

$[C(1,3) + C(4,4)]\}$

$= 11 + \min\{(0+8), (7+3), (12+0)\} = 11+8 = 19$

Scanned with CamScanner

$W(0,3) = P(3) + Q(3) + W(0,2) = 1+1+12 = 14$

$C(0,3) = W(0,3) + \min\{[C(0,0)+C(1,3)], [C(0,1)+C(2,3)], [C(0,2)+C(3,3)]\}$

$= 14 + \min\{(0+12), (8+3), (19+0)\} = 14+11 = 25$

$R(0,3) = 2$

Scanned with CamScanner

Start with $i=1$; so $j=4$; as $i<k\leq j$, so the possible values for $k = 2, 3$ and 4

$W(1,4) = P(4) + Q(4) + W(1,3) = 1+1+9 = 11$

$C(1,4) = W(1,4) + \min\{[C(1,1)+C(2,4)], [C(1,2)+C(3,4)], [C(1,3)+C(4,4)]\}$

$= 11 + \min\{(0+8), (7+3), (12+0)\} = 11+8 = 19$

$R(1,4) = 2$

Fourth, computing all $C(i,j)$ such that $j-i=4$; $j=i+4$ and as $0\leq i<1$, $i=0$; $i<k\leq j$

$W(0,4) = P(4) + Q(4) + W(0,3) = 1+1+14 = 16$

$C(0,4) = W(0,4) + \min\{[C(0,0)+C(1,4)], [C(0,1)+C(2,4)], [C(0,2)+C(3,4)], [C(0,3)+C(4,4)]\}$

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

$= 16+16$

$= 32$

$R(0,4) = 2$

Column / Row	0	1	2	3	4
0	2,0,0	3,0,0	1,0,0	1,0,0	1,0,0
1	8,8,1	7,7,2	3,3,3	3,3,4	
2	19,19,1	19,12,2	5,8,3		
3	14,25,2	11,19,2			
4	16,32,2				

93/107

Scanned with CamScanner

Algorithm:

Algorithm OBST(P,Q,n)

{

for i:=0 to n-1 do

{

$W[i,i] = Q(i)$, $R[i,i] = 0$, $C[i,i] = 0.0$

Scanned with CamScanner

2	12, 19, 1	19, 12, 2	5, 8, 3
3	14, 25, 2	11, 19, 2	
4	16, 32, 2		

Scanned with CamScanner

Algorithm:

Algorithm OBST (P, q, n)

{

for i := 0 to n-1 do

{

$w[i,i] = q(i)$, $r[i,i] = 0$, $c[i,i] = 0.0$

$w[i,i+1] :=$

From the table we see that $c(0,4) = 32$ is the minimum cost of a binary search tree for (a_1, a_2, a_3, a_4). The root of the tree T_{04} is a_2.

Hence the left subtree is T_{01} and right subtree is T_{24}. The root of T_{01} is a_1 and the root of T_{24} is a_3.

The left and right subtrees for T_{01} are T_{11} respectively

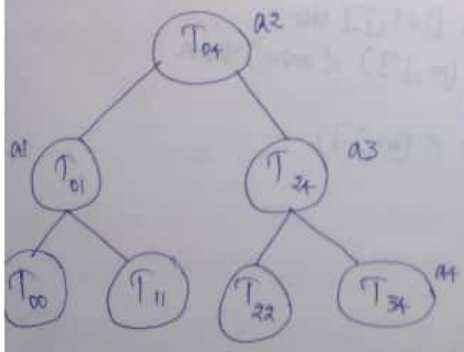
The root of T_{01} is a_1

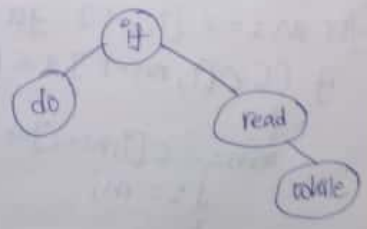
The left and right subtrees for T_{24} are T_{22} and T_{34} respectively.

The root of T_{24} is a_3

The root of T_{22} is null

The root of T_{34} is a_4





94/107

Scanned with CamScanner

Algorithm OBST (P, q, n)

{

for i := 0 to n-1 do

{

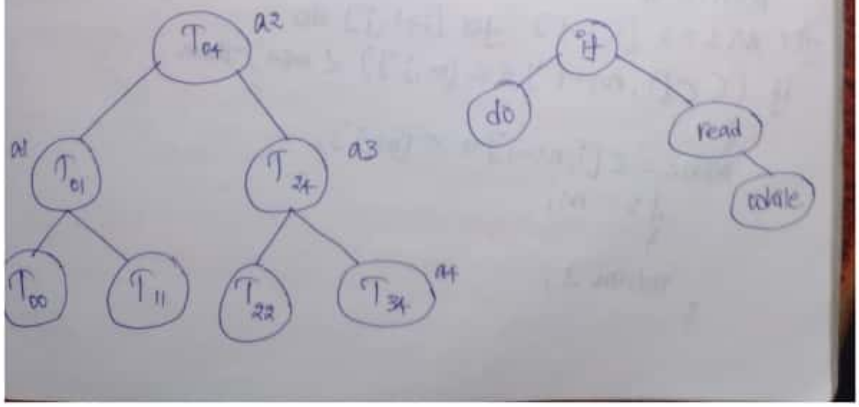
$w[i,i] = q(i)$, $r[i,i] = 0$, $c[i,i] = 0.0$;

$w[i,i+1] := q(i) + q[i+1] + p[i+1]$;

$r[i,i+1] = i+1$;

Scanned with CamScanner

the root of T_{34} is a_4



Scanned with CamScanner

```
Algorithm OBST (P, q, n)
{
for i:=0 to n-1 do
{
  w[i,i] = q(i), r[i,i]=0, c[i,j]=0.0;
  w[i,i+1]:= q[i] + q[i+1] + p[i+1];
  r[i,i+1]=i+1;
  c[i,i+1]:= q[i] + q[i+1] + p[i+1];
}
w[n,m]:= q[n]; r[n,n]:=0; c[n,n]:=0.0;
for m:=2 to n do
 for i:=0 to n-m do
   j:=i+m;
w[i,j]:= w[i,j-1] + p[j] + q[j];
  k= Find (c,r,i,j);
c[i,j]:= w[i,j] + c[i,k-1] + c[k,j];
     r[i,j]:=k;
}
write (c[0,n], w[0,n], r[0,n]);
}
Algorithm Find (c,r,i,j)
 {
  min:=∞
for m:=r[i,j-1] for [i+1,j] do
 if (c c[i,m-1] + c[m,j]) < min) then
   {
   min:= c[i,m-1] + c[m,j];
      d:=m;
   }
   return d;
 }
```

95/107

Scanned with CamScanner

Unit-III

Techniques for binary trees:

Scanned with CamScanner