3-Dynamic programming Method:

Dynamic programming is a powerful algorithm design technique that is widely used to solve the optimization problems in which a set of choices must be made in order to arrive at an optimal solution.

It is used to the problems that its solution represents a sequence of decisions. The main difference between the greedy method (its decisions step by step and depends on local information) and the Dynamic programming (its decision step by step and depends on global information) is that in the first method, only one sequence of decisions is generate, while in the second can generate more sequences of decisions but the sequences that contains unoptimal subsequences will not generate.

3-1. Multistage Graph problem:

It is a directed graph in which nodes divided into \((k \geq 2)\) separated sets \(V_i\) where \((1 \leq i \leq k)\) \(k\) is the number of stages. If the edge \(<u, v>\) in \(V\)

\[
\begin{align*}
    u & \in V_i \\
    v & \in V_{i+1} \\
    1 \leq i \leq k
\end{align*}
\]

the two sets \(V_1\) and the \(V_k\) will be

\[|V_1| = |V_k| = 1\]

Suppose \(s\) is the start node in \(V_1\) and \(t\) is the ending node in \(V_k\), and suppose \(c[i,j]\) is the cost of the edge \(<i,j>\). The cost of the path from \(s\) to \(t\) is the sum of the edge costs along the path.
The multistage graph problem is finding the minimum cost path from s to t. Each set \( V_i \) represents stage in the graph. Each path from s to t start from stage 1 and ending in the stage \( k \). many problems can be formulated as multistage graph problem.

### Problem Formulation:

1. **Forward Approach:**

   Each path from s to t be as result for sequence of \( k-1 \) decision. Each decision \( i \) include determination any of nodes in the set \( V_{i+1} \) be on the path \((1 \leq i \leq k-2)\). Suppose \( P(i,j) \) is the minimum cost path from node \( j \) in the set \( V_i \) to the node \( t \), and \( \text{cost}(i,j) \) is the cost of the path from node \( j \) in the set \( V_i \) to the end.

   \[
   \text{Cost}(i,j) = \min \{ \text{cost}(j, r) + \text{cost}(i+1, r) \} \quad \text{...(1)}
   \]

   \[
   r \in V_{i+1}, \quad <j, r> \in E
   \]

   where

   \[
   \text{cost}(k-1,j) = \begin{cases} 
   \infty & \text{if } <j,t> \notin E \\
   \min \{ \text{cost}(j, r) \} & \text{if } <j,t> \in E 
   \end{cases} \quad \text{...(2)}
   \]

   we can find the minimum cost path from s to t as follow:

   \[
   \begin{align*}
   \text{cost}(3,6) &= \min \{ 6 + \text{cost}(4,9), 5 + \text{cost}(4,10) \} = 7 \\
   \text{cost}(3,7) &= \min \{ 4 + \text{cost}(4,9), 3 + \text{cost}(4,10) \} = 5 \\
   \text{cost}(3,8) &= \min \{ 5 + \text{cost}(4,10), 6 + \text{cost}(4,11) \} = 7 \\
   \text{cost}(2,2) &= \min \{ 4 + \text{cost}(3,6), 2 + \text{cost}(3,7), 1 + \text{cost}(3,8) \} = 7 \\
   \text{cost}(2,3) &= \min \{ 2 + \text{cost}(3,6), 7 + \text{cost}(3,7) \} = 7 \\
   \text{cost}(2,4) &= \min \{ 11 + \text{cost}(3,8) \} = 18 \\
   \text{cost}(2,5) &= \min \{ 11 + \text{cost}(3,7), 8 + \text{cost}(3,8) \} = 15 \\
   \text{cost}(1,1) &= \min \{ 9 + \text{cost}(2,2), 7 + \text{cost}(2,3), 3 + \text{cost}(2,4), 2 + \text{cost}(2,5) \} = 16 \\
   \end{align*}
   \]

   suppose \( D(i,j) = r \) where \( r \) minimize the value of \( \text{cost}(j,r) + \text{cost}(i+1,r) \)
The multistage graph algorithm (Forward Approach):

The algorithm is supposed that the nodes in the set V are sorted from 1 to n and the indices that given to nodes Vi+1 be larger than nodes in the Vi where s is given the index 1 and then nodes in the set V2 are given indices and so on until we arrive at t that is given the index n.

Procedure FGraph( G : Graph; k, n: integer; var P: PathList; var Pathcost: real);
Type Lcost = array [1..MaxN] of real;
var cost : Lcost;
D: array [1..MaxN] of integer;
r, j : integer;
begin
cost [n] := 0;
for j:= n-1 downto 1 do
begin
  let r be a vertex such that <j, r> ∈ E and c[j, r] + cost[r] is minimum;
  cost[j] := c[j, r] + cost[r];
  D[j] := r;
End;
P[1] := 1;
P[k] := n;
 For j:= 2 to k-1 do  P[j] := D[P[j-1]];
Pathcost := cost[1];
End;
2. Backward Approach:

\[\text{Cost}(i,j) = \min\{c[r,j] + \text{cost}(i-1,r)\} \quad \ldots \ldots \ldots \quad (1)\]

\[r \in V_{i-1}\]

\[<r,j> \in E\]

where

\[\text{cost}(k-1,j) = \begin{cases} 0 & \text{if } <1,j> \in E \\ \infty & \text{if } <1,j> \notin E \end{cases} \quad \ldots \ldots \ldots \quad (2)\]

we can find the minimum cost path from s to t as follow:

\[i=3\]

\[\text{cost}(3,6) = \min\{4 + \text{cost}(2,2), 2 + \text{cost}(2,3)\} = 9\]

\[\text{cost}(3,7) = \min\{2 + \text{cost}(2,2), 7 + \text{cost}(2,3), 11 + \text{cost}(2,5)\} = 11\]

\[\text{cost}(3,8) = \min\{1 + \text{cost}(2,2), 11 + \text{cost}(2,4), 8 + \text{cost}(2,5)\} = 10\]

\[\text{cost}(4,9) = \min\{6 + \text{cost}(3,6), 4 + \text{cost}(3,7)\} = 15\]

\[\text{cost}(4,10) = \min\{5 + \text{cost}(3,6), 3 + \text{cost}(3,7), 5 + \text{cost}(3,8)\} = 14\]

\[\text{cost}(4,11) = \min\{6 + \text{cost}(3,8)\} = 16\]

\[\text{cost}(5,12) = \min\{4 + \text{cost}(4,9), 2 + \text{cost}(4,10), 5 + \text{cost}(4,11)\} = 16\]

suppose \(D(i,j) = r\) where \(r\) minimize the value of \(c[r,j] + \text{cost}(i-1,r)\)

\[D(3,6) = 3 \quad D(4,9) = 6\]

\[D(3,7) = 2 \quad D(4,10) = 6\]

\[D(3,8) = 2 \quad D(4,11) = 8\]

\[D(5,12) = 10\]

\[1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 12\]

\[V_4 = D(5,12) = 10\]

\[V_3 = D(4,10) = 6\]

\[V_2 = D(3,6) = 3\]

The path = \(1 \rightarrow 3 \rightarrow 6 \rightarrow 10 \rightarrow 12\)
The multistage graph algorithm (Backward Approach):

Procedure BGraph( G : Graph; k, n: integer; var P: PathList; var Pathcost: real);
Type Lcost = array [1..MaxN] of real;
var cost : Lcost;
   D: array [1..MaxN] of integer;
   r, j : integer;
begin
  cost [1] := 0;
  for j:= 2 to n do
     begin
       let r be a vertex such that <r, j> ∈ E and c[r, j] + cost[r] is minimum;
       cost[j] := c[r, j] + cost[r];
       D[j] := r;
     End;
P[1] := 1;
P[k] := n;
  For j:= k-1 downto 2 do  P[j] := D[P[j + 1]];
  Pathcost := cost[n];
End;

Homework:

Find the path with the minimum cost from s to t in the following multistage graph using dynamic programming with

1. The Forward Approach.
2. The Backward Approach.