Dynamic Programming

* We enumerate/generate all the possible decision sequences (but it consumes more time and space)
* So Dynamic programming considerably reduces the number of enumerations to those few instances which may eventually lead to an optimal solution.
* Based on principle of optimality

Difference between Dynamic programming and Greedy Method

* In Greedy method only one decision sequence is ever generated
* In dynamic programming many decision sequences are ever generated.

Multistage Graph Problem (Forward Approach)

Initial Condition

Cost(k-1, j) = c(j,t) if <j,t> belongs to E

Cost(k-1,j) = infinity if<j,t> does not belong to E

Formulae (Forward approach)

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

Where ‘l’ belongs to vi+1(vertex in next stage)

<j,l> belong to E

Solution:

Initial (when k=4) cost(4,9) = 4

Cost(4,10) =2

Cost(4,11)=5

For k=3 (stage 3)

1. Cost(3,6) = min{ c(6,9) +cost(3+1,9) , c(6,10)+cost(3+1,10) }

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

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

Cost(3,6)=7 where l=10

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

= min { 4+ 4, 3+2 }

Cost(3,7) = 5 where l=10

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

= min {5+2 , 6+5 }

Cost(3,8) = 7 where l=10

For k=2(stage 2)

1. Cost(2,2)= 7 where l=7
2. Cost(2,3)= 9 where l=6

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

=min(2+7, 7+5) = 9

1. Cost(2,4)= 18 where l= 8
2. Cost(2,5)= 15 where l= 8

For k=1 (stage 1)

1. Cost(1,1)= min{9+cost(2,2) , 7+cost(2,3), 3+cost(2,4), 2+cost(2,5)} = 16

where l= 2 or 3

Minimum cost path s =1->v2->v3->v4-> t

Where v2=d(1,1) => v2= 2 or 3

V3=d(2,d(1,1)) => v3=d(2,2)=7 or d(2,3)=6

V4 =d(3,d(2,d(1,1))) => v4 = d(3,7)=10 or d(3,6)=10

Shortest path from s to t

1->2->7->10->12 mini cost=16

1->3->6->10->12 mini cost=16

Find ‘d’ values {that l values becoz of which cost became minimum}

d(1,1) = 2 or 3

d(2,2) =7

d(2,3)=6

d(2,4)=8

d(2,5)=8

d(3,6)=10

d(3,7)=10

d(3,8)=10