Dynamic Programming

- Operations Research
  - Shortest Path Problems
  - Equipment Replacement
  - Resource Allocation
  - Traveling Salesman Problems
  - Cargo Loading Problems
- Engineering
  - Inventory Control
  - Temperature Control
- Economics
  - Auctions
- Computer Science
  - Scheduling, Capacitated Minimum Spanning Tree (CMST)
  - Quadratic Assignment Problem (QAP)

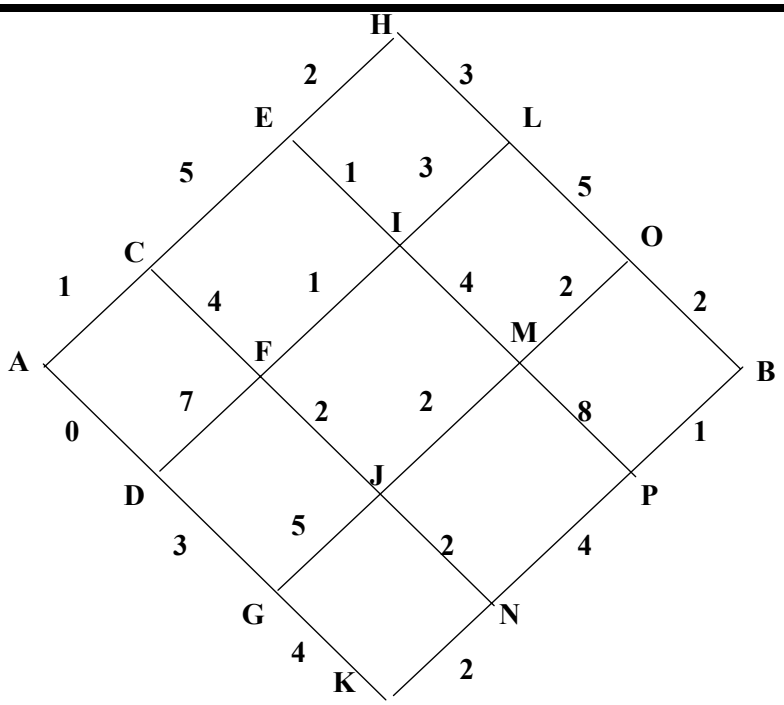
The Basic Problem

- Situations where decisions are made in stages.
- The outcome of each decision is not fully predictable but can be observed before the next decision is made.
- The objective is to minimize a certain cost – a mathematical expression of what is considered desirable outcome.

- One must balance the desire for low present cost with the possibility of high future costs being inevitable.

- Dynamic Programming (DP): at each stage one selects a decision that minimizes the sum of the current stage cost, and the best cost that can be expected from the future stages.

Generalized DP Problem



Get from point A to point B in shortest distance

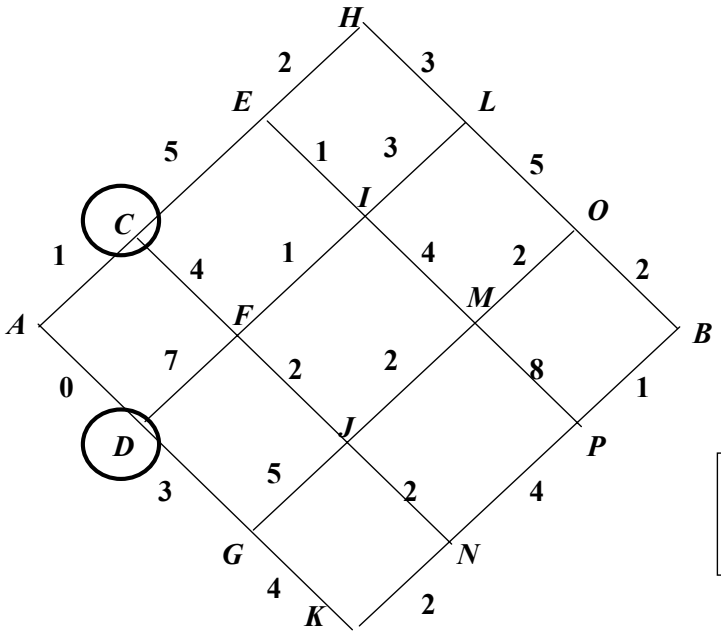
Brute Force Enumeration

- There are 20 distinct paths from A to B
- Five additions yield the sum of six numbers along a particular path.
- 100 additions would yield the 20 path sums to be compared.
- 19 comparisons to compare all of the solutions

Dynamic programming solves the problem with fewer additions and compares

Principle of Optimality

Could make the best decision at A if a person knew the optimal way to get to B from C and D



Likewise for every decision point in tree

DP Solution

Distance from A to C

Optimal from C to B

$$S_A = \min \begin{bmatrix} 1 + S_C \\ 0 + S_D \end{bmatrix}$$

DP Solution

$S_P = 1$

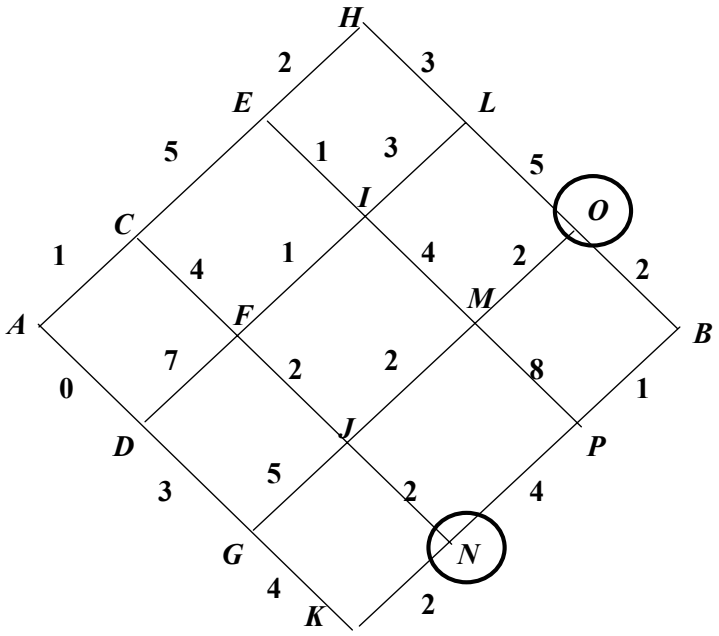
$S_O = 2$

$S_L = 5 + S_O = 7$

$S_H = 3 + S_L = 10$

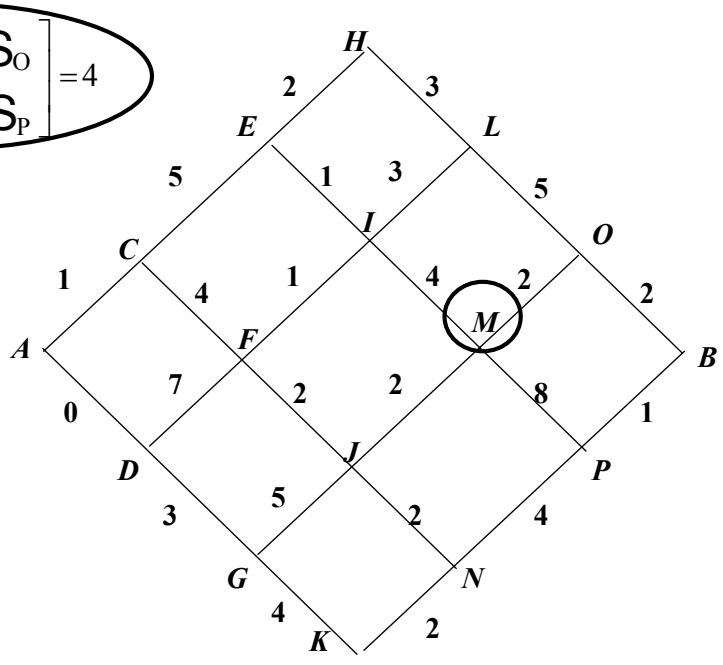
$S_K = 2 + S_N = 7$

$S_N = 4 + S_N = 5$



DP Solution

$$S_M = \min \begin{bmatrix} 2 + S_O \\ 8 + S_P \end{bmatrix} = 4$$



Full DP Solution

$S_P = 1$

$S_O = 2$

$S_L = 5 + S_O = 7$

$S_H = 3 + S_L = 10$

$S_K = 2 + S_N = 7$

$S_N = 4 + S_N = 5$

$$S_M = \min\begin{bmatrix} 2+S_O \\ 8+S_P \end{bmatrix} = 4$$

$$S_I = \min\begin{bmatrix} 3+S_L \\ 4+S_M \end{bmatrix} = 8$$

$$S_J = \min\begin{bmatrix} 2+S_M \\ 2+S_N \end{bmatrix} = 6$$

$$S_E = \min\begin{bmatrix} 2+S_H \\ 1+S_I \end{bmatrix} = 9$$

$$S_A = \min\begin{bmatrix} 1+S_C \\ 0+S_D \end{bmatrix} = 13$$

$$S_G = \min\begin{bmatrix} 5+S_K \\ 4+S_E \end{bmatrix} = 11$$

$$S_C = \min\begin{bmatrix} 5+S_E \\ 4+S_F \end{bmatrix} = 12$$

$$S_D = \min\begin{bmatrix} 7+S_F \\ 3+S_G \end{bmatrix} = 14$$

$$S_F = \min\begin{bmatrix} 1+S_I \\ 2+S_J \end{bmatrix} = 8$$

20 additions
9 compares

Best Path

- Let x represent any particular starting point, denoted by P_X the node after node x the optimal path from x to B,
- P table could be computed as the S table above.

The optimal next point for each initial point

$P_O = B$	$P_P = B$		
$P_L = O$	$P_M = O$	$P_N = P$	
$P_H = L$	$P_I = M$	$P_J = M$	$P_K = N$
$P_E = I$	$P_F = J$	$P_G = J$ or K	
$P_C = F$	$P_D = G$		
$P_A = C$			

Using this table, best path from A to B, P_A= C so we move to C, since P_C= F we continue to F, and P_F= J we move next to J, P_J= M SO move to M, P_M= O so move to O, and P_O= B

Therefore, The best path A-C-F-J-M-O-B

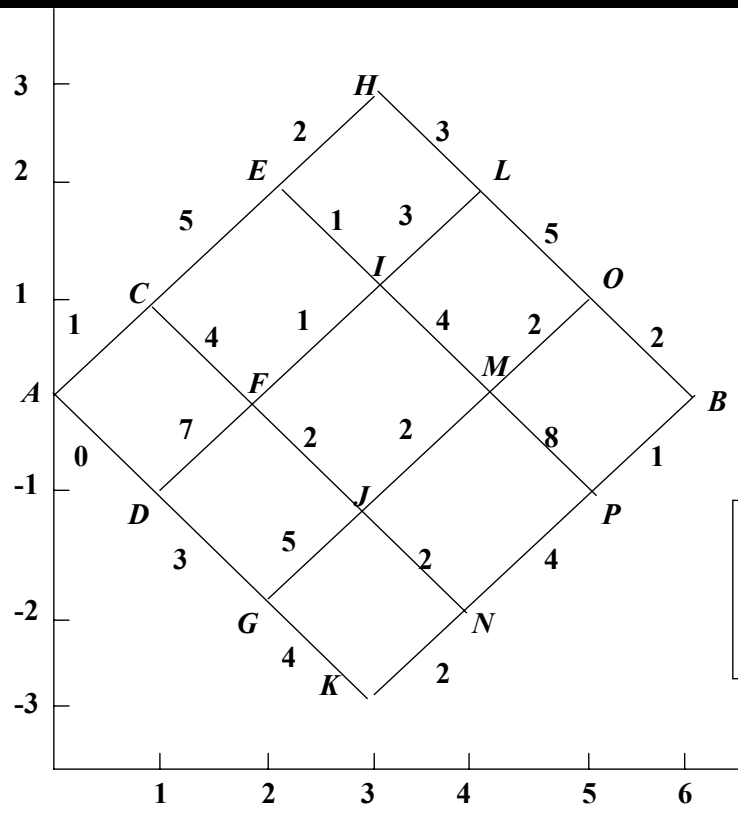
Terminology

- **Optimal Value Function**– rule that assigns values to each sub problem
  - S(x,y)
- **Optimal Policy Function** – rule that associate the best 1st decision with each sub problem
- **Recurrence Relation** – formula that relates various values of S.
- **Boundary Conditions** – value of S is assumed for certain obvious conditions

Steps in Creating a DP

1. Define the appropriate ***Optimal Value Function***
2. Write the appropriate ***Recurrence Relationship***
3. Note the appropriate ***Boundary Conditions***

Converting to Grid



This helps to understand the mathematical relationships

Example

- **Optimal Value Function**
  S(x,y) = value of minimum effort connecting node (x,y) with terminal node (6,0)
- **Recurrence Relationship**

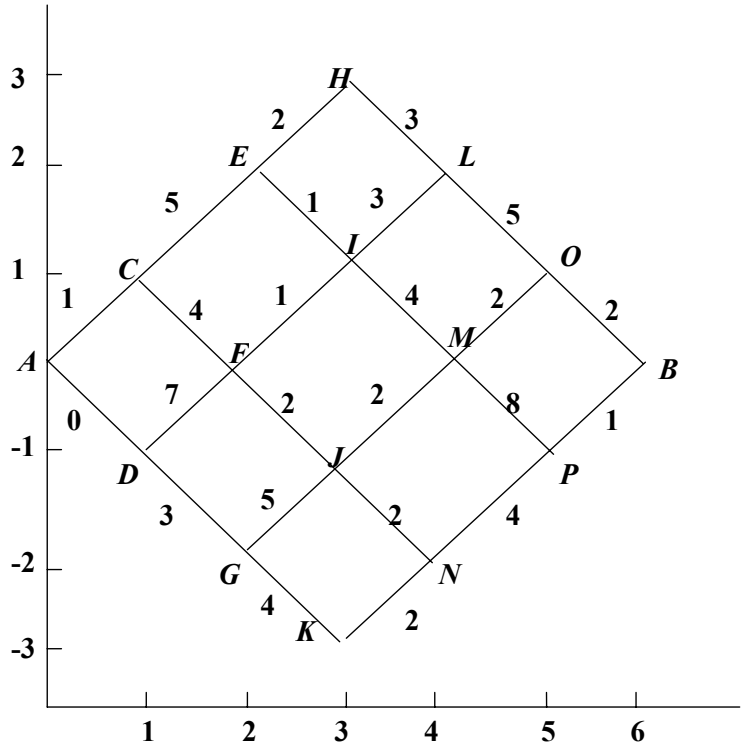
$$S(x,y) = \min \begin{bmatrix} a_u(x,y) + S(x+1, y+1) \\ a_d(x,y) + S(x+1, y-1) \end{bmatrix}$$

Where $a_u(x,y)$ denote the effort associated with the arc connecting the node (x,y) with the node (x+1, y+1), subscript **u** signifies that arc goes up.

- **Boundary Condition**
  S(6,0) = 0
  Since effort going from (6,0) to (6,0) is zero

Stage and State

- Suppose there was a penalty for turning in problem just described
- ***Stage*** variable describes where you are
- ***State*** variable describes current direction at each node



Solving

- How would you solve the turn problem?
  - Add a state variable, z, for direction
  - Now have 2 states at each node, where z=0 denotes diagonally upward, z=1 denotes diagonally downward.

$$S(x,y,0) = \min \begin{bmatrix} a_u(x,y) + S(x+1,y+1,0) \\ a_d(x,y) + 3 + S(x+1,y-1,1) \end{bmatrix}$$

moving upward direction incur S(x+1, y+1,0), but moving downward instead the cost is 3 plus S(x+1, y+1,1).

$$S(x,y,1) = \min \begin{bmatrix} a_u(x,y) + 3 + S(x+1,y+1,0) \\ a_d(x,y) + S(x+1,y-1,1) \end{bmatrix}$$

Knapsack Algorithm

Problem statement:

- Person going climbing and has a upper weight limit
- Wants to pack a number of food items, each with an integer weight.
- Each food item has a value (calories, taste)
- Wants to maximize value given available weight limit

Features of Knapsack Problem

- Strictly integer problem
- Only one constraint (in this case total length of the stem)
- Each item has:
  - cost – amount of weight capacity it takes
  - value –the value of the item taken

Example Knapsack Problem

- Weight capacity of a cargo vehicle (9)
- Want to maximize the value of the cargo
- Unlimited number of each of N items with
  - weight $\mathbf{w_i}$
  - value $\mathbf{v_i}$
- Limited to integer number of items xi

Data for Example Problem

Item Type i	Weight (w_i)	Value (v_i)	v_i / w_i
1	3	7	2.3333
2	6	16	2.6666
3	7	19	2.7142
4	5	15	3.0000

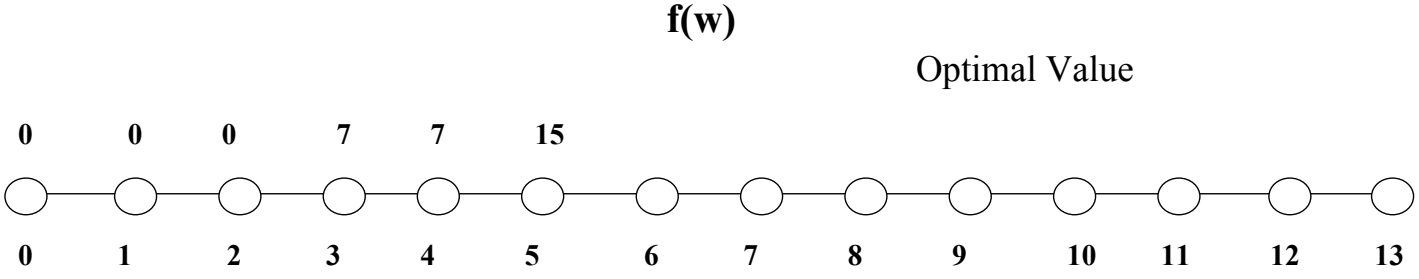
Highest v/w ratio
Highest non-integer solution is $x_4 = 9/5$

Knapsack Problem

Item Type i	Weight (w_i)	Value (v_i)	v_i / w_i
1	3	7	2.3333
2	6	16	2.6666
3	7	19	2.7142
4	5	15	3.0000

Note:
- Greatest common denominator of weight is 1
- Can always adjust weights to get integer if necessary

Graphic Representation



Define:

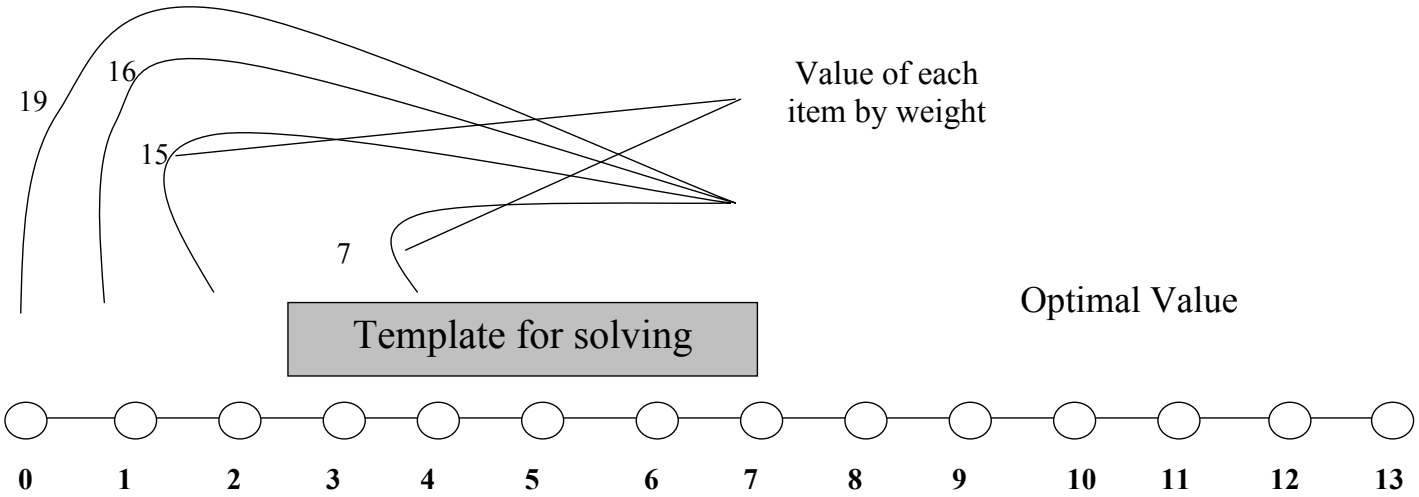
f(w) = maximum value attainable with weight capacity w

$f(w) = \max \{ v_i + f(w - w_i) \}$

i = 1,..,N (over all products)

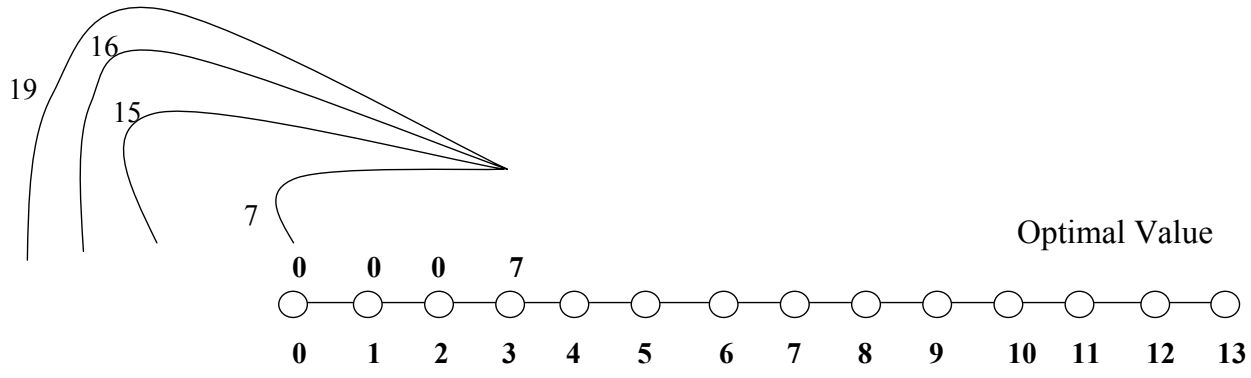
Recurrence relationship

Graphic Solution Procedure



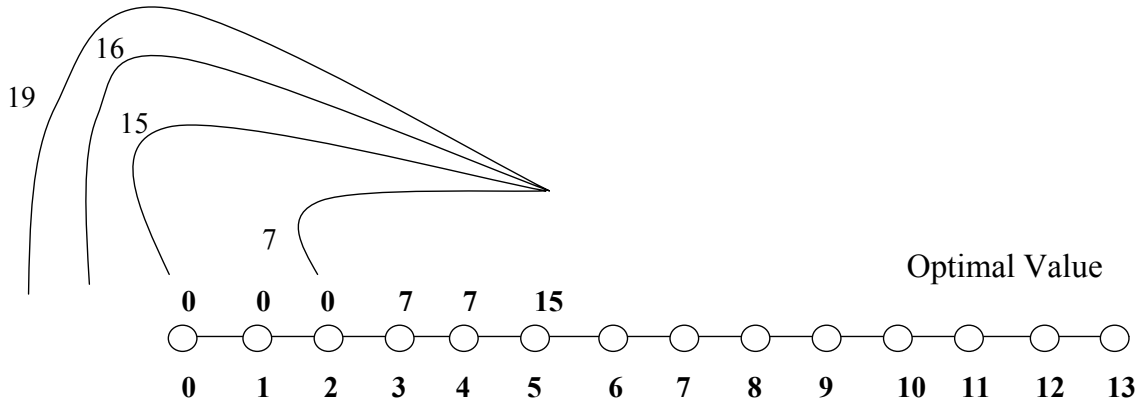
Available weight (w)

Optimal at 3 - f(3)

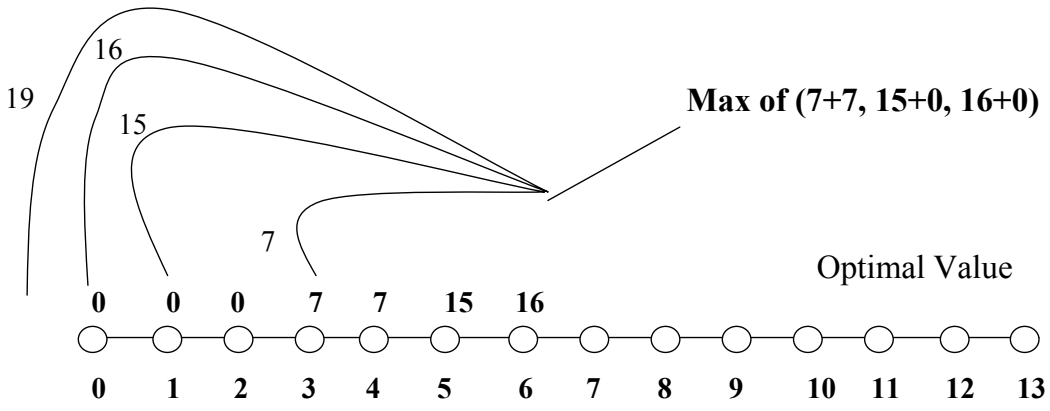


Available weight (w)

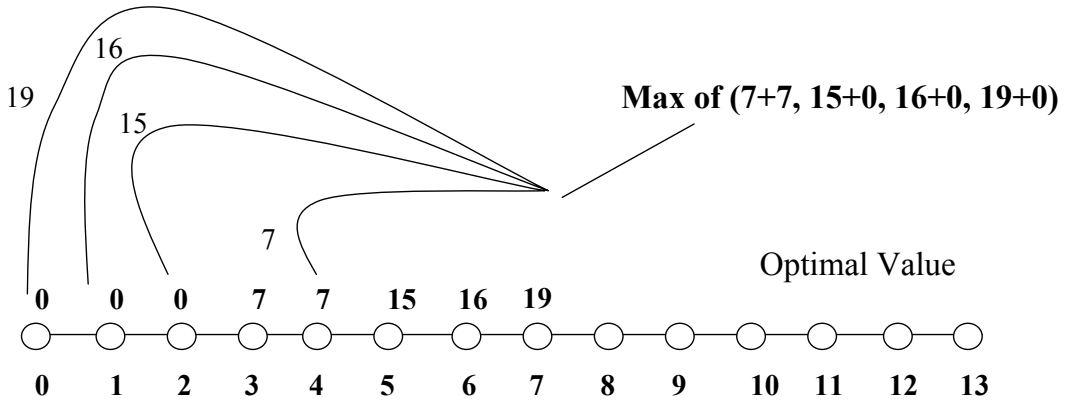
Optimal at 5 - f(5)



Optimal at 6 - f(6)

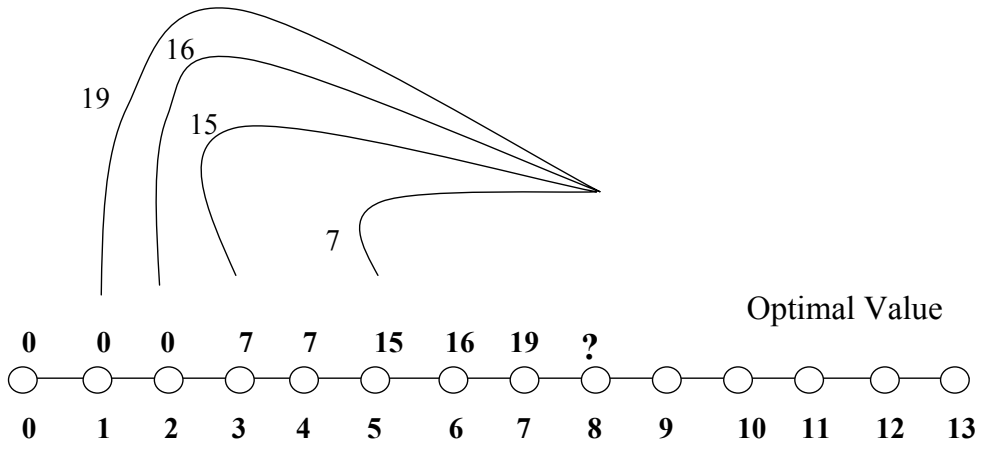


Optimal at 7 - f(7)



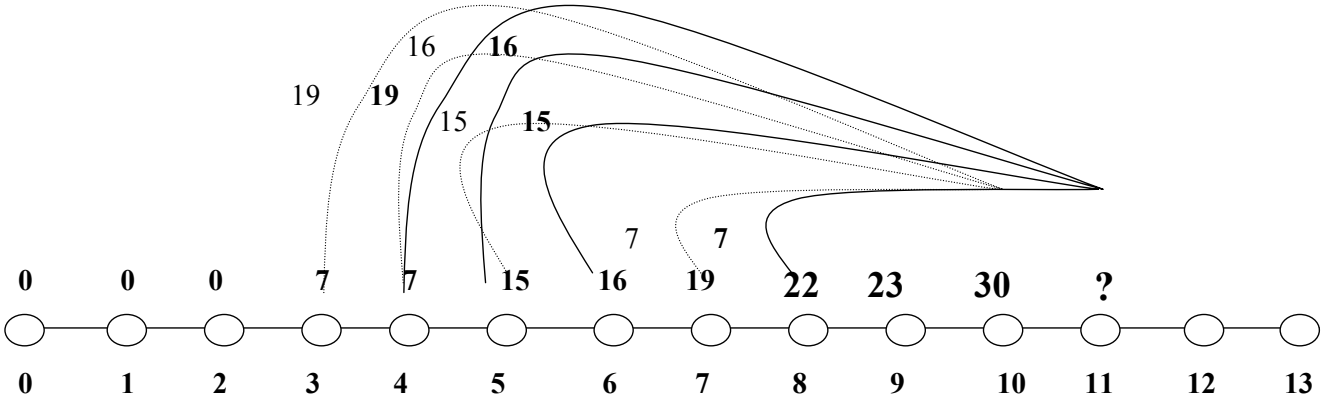
Available weight (w)

Optimal at 8 - f(8)



Available weight (w)

Optimal Packing Solution f(11)



Available weight (w)

Two Questions:

1. What information do you need to determine the optimal packing strategy?
2. The time to solve this problem is proportional to what expression?

Example Computer Code

```
For (j=1; j<= N; j++)
    {
        for (i=1; i <= M; i++)
            if ( i > = size[j] )
                {
                    cost[i] = cost[i -size[j]] + val[j];
                    best[i] = j;
                }
    }
```

Equipment Replacement Problems

Problem Statement

Crafty Cabinets is considering when to replace a CNC router that is currently 2 years old. A new machine costs $50,000. Operation costs increase every year due to maintenance, and the trade in value decreases every year, and at some point it has no trade in value at all.

How should Crafty decide when to replace the machine using DP as a solution mechanism?

Important Data

c(i) = cost of operating machine for one year at age i (start of the year)

p = price of new machine

t(i) = trade in value of new machine at age i (start of year)

s(i) = salvage value received for machine that has turned age i (at the end of year N)

y = current age of machine

Important Data

Operating Cost

c = {10,13,20,40,70,100,100}

Solution Period:

N = 5 yrs

New Machine Cost:

p = 50

Trade-in Value

t = {32,21,11,5,0,0}

Salvage Value

s = {25,17,8,0,0,0}

Solve this as a shortest path network.

Formulating the Problem

- Optimal Value Function
- Recurrence Relationship
- Boundary Condition

Optimal Value Function

S(x,k) = minimum cost of owning a machine from year k through N, starting year k with machine just turned age x.

Formulating the Problem

Recurrence Relationship

$$S(x,k) = \min \begin{bmatrix} \text{buy}: & p - t(x) + c(0) + S(1,k+1) \\ \text{keep}: & c(x) + S(x+1,k+1) \end{bmatrix}$$

Boundary Condition

S(x,N+1) = -s(x)

This sets the salvage value of the machine at the end of the period.

Solving the Problem

According to boundary condition

S(1,6) = -25, S(2,6) = -17, S(3,6) = -8, S(4,6) = S(5,6) = S(7,6) = 0

$$S(1,5) = \min \begin{bmatrix} 50 - 32 + 10 + S(1,6) \\ 13 + S(2,6) \end{bmatrix} = -4, \quad P(1,5) = \text{keep}$$

$$S(2,5) = \min \begin{bmatrix} 50 - 21 + 10 + S(1,6) \\ 20 + S(3,6) \end{bmatrix} = 12, \quad P(2,5) = \text{keep}$$

$$S(3,5) = \min \begin{bmatrix} 24 \\ 40 \end{bmatrix} = 24, \quad P(3,5) = \text{buy}$$

$$S(4,5) = \min \begin{bmatrix} 30 \\ 70 \end{bmatrix} = 30, \quad P(4,5) = \text{buy}$$

$$S(6,5) = \min \begin{bmatrix} 35 \\ 100 \end{bmatrix} = 35, \quad P(6,5) = \text{buy}$$

Solving the Problem

Now determine the S(x,4)

$$S(1,4) = \min \begin{bmatrix} 28 + S(1,5) \\ 13 + S(2,5) \end{bmatrix} = 24, \quad P(1,4) = \text{buy}$$

$$S(2,4) = \min \begin{bmatrix} 35 \\ 44 \end{bmatrix} = 35, \quad P(2,4) = \text{buy}$$

$$S(3,4) = \min \begin{bmatrix} 45 \\ 70 \end{bmatrix} = 45, \quad P(3,4) = \text{buy}$$

$$S(5,4) = \min \begin{bmatrix} 56 \\ 135 \end{bmatrix} = 56, \quad P(5,4) = \text{buy}$$

Solving the Problem

Next determine the S(x,3)

$$S(1,3) = \min \begin{bmatrix} 52 \\ 48 \end{bmatrix} = 48, \quad P(1,3) = \text{keep}$$

s

$$S(2,3) = \min \begin{bmatrix} 63 \\ 65 \end{bmatrix} = 63, \quad P(2,3) = \text{buy}$$

$$S(4,3) = \min \begin{bmatrix} 79 \\ 126 \end{bmatrix} = 79, \quad P(4,3) = \text{buy}$$

Next determine the S(x,2)

$$S(1,2) = \min \begin{bmatrix} 76 \\ 76 \end{bmatrix} = 76, \quad P(1,2) = \text{buy or keep}$$

$$S(3,2) = \min \begin{bmatrix} 97 \\ 119 \end{bmatrix} = 97, \quad P(3,2) = \text{buy}$$

Solving the Problem

Finally,

$$S(2,1) = \min \begin{bmatrix} 115 \\ 117 \end{bmatrix} = 115, \qquad P(2,1) = \text{buy}$$

The optimal sequence of decisions :

1. Buy at year 1 since P(2, 1)= **buy**;
2. Either buy or keep at the start of year 2 since P(1, 2) -either;
3. If we choose to **buy**, then keep during year 3 since P(1, 3) = **keep**;
4. Then **buy** at the start of year 4 since P(2, 4) =buy
5. And **keep** during year 5 since P(1, 5) =keep.

2. If we choose to **keep** during year 2, then **buy** at the start of year 3, **buy** at the start of year 4, and **keep** during year 5.

Therefore the two optimal policies are, *BBKBK* and *BKBBK*

Computer Code for DP

```
x = 1
For k = Year To 1 Step -1
    For x = 1 to k + 1
    If x <> k Then
        Buy = PurchPrice - TradeIn(x) + Cost(0) + Optimal(1, k + 1)
        Keep = Cost(x) + Optimal(x + 1, k + 1)
        If Buy < Keep Then
            Optimal(x, k) = Buy
            Decision(x, k) = "Buy"
        Else
            Optimal(x, k) = Keep
            Decision(x, k) = "Keep"
        End If
    End If
  Next
Next
```