Unit 4: Dynamic Programming

Dynamic programming

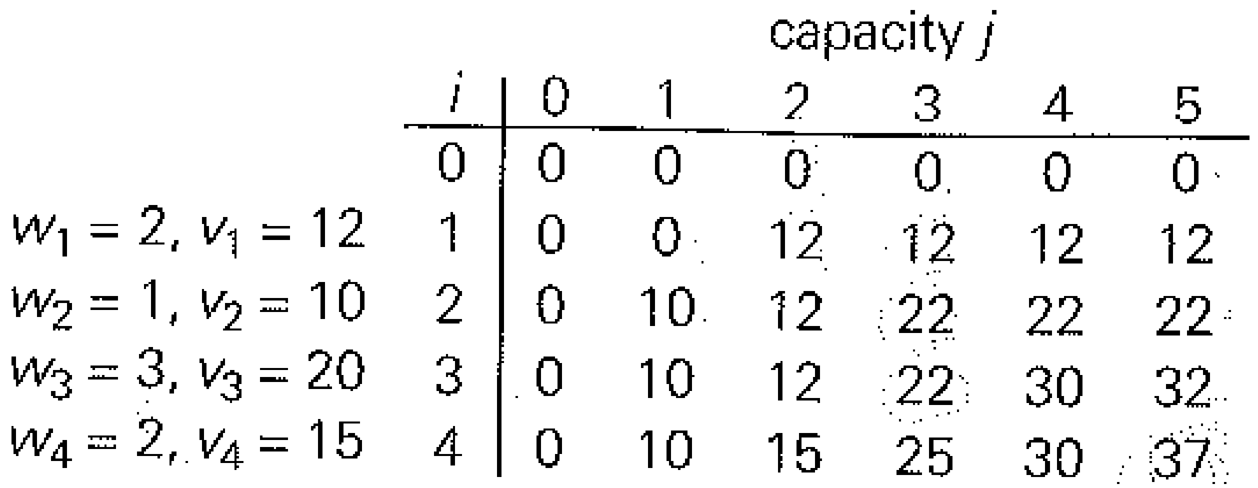
- **Memory function**
- **Puzzles**

Solve using memory function

item	weight	value
1	2	\$12
2	1	\$10
3	3	\$20
4	2	\$15

capacity $W = 5$

Bottom-up approach – solving the smaller subproblems first and then solving bigger subproblems, but each of them being solved only once



		capacity j					
	i	0	1	2	3	4	5
	0	0	0	0	0	0	0
$w_1 = 2, v_1 = 12$	1	0	0	12	12	12	12
$w_2 = 1, v_2 = 10$	2	0	10	12	22	22	22
$w_3 = 3, v_3 = 20$	3	0	10	12	22	30	32
$w_4 = 2, v_4 = 15$	4	0	10	15	25	30	37

optimal solution {item 1, item2, item 4)

With maximal value = Rs.37

Limitations:

some of these smaller subproblems are often not necessary for getting a solution to the problem given

Memory functions

- combines the strengths of the top-down and bottom-up approaches
- solves only subproblems which are necessary and does it only once

- **Top down approach + Table**
- Initially, all the table's entries are initialized with a special "null" symbol to indicate that they have not yet been calculated. Thereafter, whenever a new value needs to be calculated, the method checks the corresponding entry in the table first: if this entry is not "null," it is simply retrieved from the table; otherwise, it is computed by the recursive call whose result is then recorded in the table

```
ALGORITHM  MFKnapsack(i, j)
    //Implements the memory function method for the knapsack problem
    //Input: A nonnegative integer i indicating the number of the first
    //       items being considered and a nonnegative integer j indicating
    //       the knapsack's capacity
    //Output: The value of an optimal feasible subset of the first i items
    //Note: Uses as global variables input arrays Weights[1..n], Values[1..n],
    //and table V[0..n, 0..W] whose entries are initialized with −1's except for
    //row 0 and column 0 initialized with 0's
    if $V[i, j] < 0$
        if $j < Weights[i]$
            $value \leftarrow MFKnapsack(i-1, j)$
        else
            $value \leftarrow \max(MFKnapsack(i-1, j),$
                            $Values[i] + MFKnapsack(i-1, j - Weights[i]))$
        $V[i, j] \leftarrow value$
    return $V[i, j]$
```

		0	1	2	3	4	5
	0	0	0	0	0	0	0
W1=2 , v1=12	1	0	-1	-1	-1	-1	-1
W2=1 , v1=10	2	0	-1	-1	-1	-1	-1
W3=3 , v1=20	3	0	-1	-1	-1	-1	-1
W4=2 , v1=15	4	0	-1	-1	-1	-1	-1

	i	capacity j 0	1	2	3	4	5
	0	0	0	0	0	0	0
$w_1 = 2, v_1 = 12$	1	0	0	12	12	12	12
$w_2 = 1, v_2 = 10$	2	0	–	12	22	–	22
$w_3 = 3, v_3 = 20$	3	0	–	–	22	–	32
$w_4 = 2, v_4 = 15$	4	0	–	–	–	–	37

Input : arr[] = {5, 5, 10, 100, 10, 5}
Output : 110
Input : arr[] = {1, 2, 3}
Output : 4
Input : arr[] = {1, 20, 3}
Output : 20

Puzzle:

Given a sequence of n positive numbers A(1) … A(n), write an algorithm for finding a contiguous subsequence A(i)…A(j) for which the sum of elements in the subsequence is maximum.

NOTE: Two contiguous elements cannot be selected

Puzzle:

Given a sequence of n positive numbers A(1) … A(n), write an algorithm for finding a contiguous subsequence A(i)…A(j) for which the sum of elements in the subsequence is maximum.

NOTE: THREE contiguous elements cannot be selected

Input: val[ ] = {6, 7, 1, 3, 8, 2, 4}
Output: 19
Explanation: The thief will steal 6, 1, 8 and 4 from the house.

Input: val[ ] = {5, 3, 4, 11, 2}
Output: 16
Explanation: Thief will steal 5 and 11

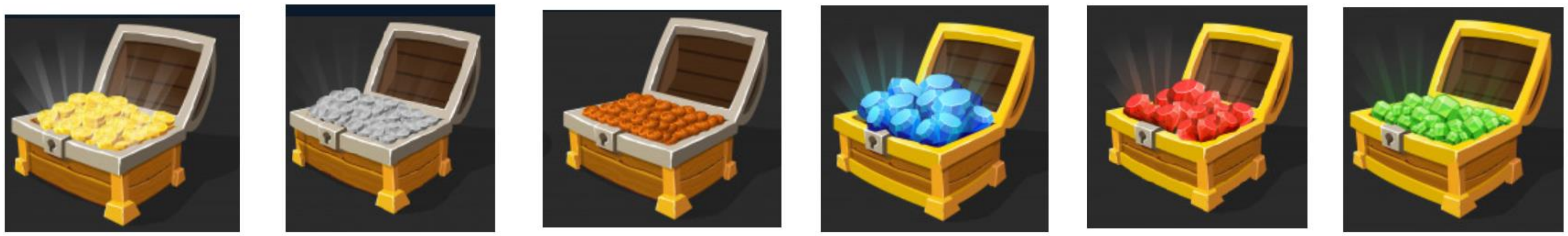
Puzzle:

There are n houses build in a line, each of which contains some value in it. A thief is going to steal the maximal value of these houses, but he can't steal in two adjacent houses because the owner of the stolen houses will tell his two neighbors left and right side.

What is the maximum stolen value?

Puzzle

5, 3, 7, 10 : The thief_1 collects maximum value as 15(10 + 5)
8, 15, 3, 7 : The thief_1 collects maximum value as 22(7 + 15)



Two thieves decide to share their loot. There are n treasures, where n is even. Each treasure contains collection of valuable gems worth values v1, v2,..., vn. They decide to arrange the treasures in a row and share by alternating turns. In each turn, a thief selects either the first or last treasure from the row, removes it from the row permanently, and now own the treasure.

Determine the maximum possible amount of money if thief_1 move first.
Note: Thief_2 is as clever as thief_1.

Let, $F(i, j)$ represents the maximum value the thief_1 can collect from ith coin to jth coin.

$F(i, j) = V_i$ If $j == i$

$F(i, j) = \max(V_i, V_j)$ If $j == i + 1$

otherwise

$F(i, j) = \text{Max}(V_i + \min(F(i+2, j), F(i+1, j-1)), V_j + \min(F(i+1, j-1), F(i, j-2)))$

Puzzle:

Code the following recurrence:

T(0) = T(1) = 2

T(n) = $\sum_{\boldsymbol{i=1}}^{\boldsymbol{n-1}} 2 \ * T(i) * T(i-1),$ for n>1

T(0) = T(1) = 2

T(n) = $\sum_{i=1}^{n-1} 2 * T(i) * T(i-1)$, for n>1

T(0) = T(1) = 2

T(2) = 2 * T(1) * T(0)

T(3) = 2 * T(1) * T(0) + 2 * T(2) *T(1)

T(4) = 2 * T(1) * T(0) + 2 * T(2) *T(1) + 2 * T(3) *T(2)