KNAPSACK PROBLEM

PROBLEM & GOAL

- We are given **n** items {1, . . . , n}, and each has a given nonnegative weight $\mathbf{w_i}$ (for i = 1, . . . , n).
- We are also given a bound of Knapsack capacity W.
- We would like to select a subset S of the items so that $\sum_{i \in S} \mathbf{w_i} \leq \mathbf{W}$ and, subject to this restriction, $\sum_{i \in S} \mathbf{V_i}$ is as large as possible.
- We will call this the Knapsack Problem.

DESIGNING THE ALGORITHM

- Let us consider and optimal solution.
- **OPT(n, W) = max profit from items 1, …, n with weight limit W.**
- There can be 2 cases if we consider an nth item as follows
  - **Case 1: OPT** does not select item n i.e. n $\notin$ OPT
    - **OPT** selects **best** of **{ 1, 2, …, n-1 }** using weight limit **W**.
  - **Case 2: OPT** selects item n. (i.e. n $\in$ OPT)
    - New weight limit = $\mathbf{W - w_n}$
    - **OPT** selects best of **{ 1, 2, …, n–1 }** using **this new weight limit**.

DESIGNING THE ALGORITHM

- Recurrence Relation

$$OPT(n,W) = \begin{cases} 0 & \text{if } \ n = 0 \\ OPT(n-1,W) & \text{if } \ w_n > W \\ \max\{OPT(n-1,W), v_n + OPT(n-1,W-w_n)\} & \text{otherwise} \end{cases}$$

KNAPSACK ALGORITHM

//Purpose: To find the maximum value or profit from the given n items and their weights wi

//Input: A set of items 1,2,…….n, with, w1, … ,wN, and values v1,v2……,vn with knapsack capacity W

//Output: Max Profit M[n, W]

```
for w = 0 to W
   M[0, w] = 0
for i = 0 to n
   M[i, 0] = 0
for i = 1 to n                  // n items
   for w = 1 to W               // weights from 1 to max cap W
      if (wi > w)
         M[i, w] = M[i-1, w]
      else
         M[i, w] = max {M[i-1, w], vi + M[i-1, w-wi]}
   endfor
endfor
return M[n, W]
```

KNAPSACK EXAMPLE

ITEMS	WEIGHTS	VALUES	W = 5
1	2	20	
2	2	10	
3	3	30	

	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0					
2	0					
3	0					

KNAPSACK EXAMPLE

i	wi	w	wi > w	M [i, w]
1	2	1	2 > 1 **TRUE**	**M[i, w] = M[i-1, w] = M[0, 1] = 0**
		2	2 > 2 **FALSE**	**= Max {M[i-1, w], vi + M[i-1, w-wi] } = Max (M [0 , 2] , 20 + M [0 , 0]) = Max(0,20) = 20**

	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	20			
2	0					
3	0					

KNAPSACK EXAMPLE

i	**wi**	**w**	**wi > w**	**M [i, w]**
1	2	3	2 > 3 **FALSE**	**= Max {M[i-1, w], vi + M[i-1, w-wi] }** **= Max (M [0 , 3] , 20 + M [0 , 1])** **= Max (0 , 20)** **= 20**
		4	2 > 4 **FALSE**	**= Max {M[i-1, w], vi + M[i-1, w-wi] }** **= Max (M [0 , 4] , 20 + M [0 , 2])** **= Max (0 , 20)** **= 20**

	0	**1**	**2**	**3**	**4**	**5**
0	**0**	**0**	**0**	**0**	**0**	**0**
1	**0**	**0**	**20**	**20**	**20**	
2	**0**					
3	**0**					

KNAPSACK EXAMPLE

i	wi	w	wi > w	M [i, w]
1	2	5	2 > 5 **FALSE**	**= Max {M[i-1, w], vi + M[i-1, w-wi] }** **= Max (M [0 , 5] , 20 + M [0 , 3])** **= Max (0 , 20)** **= 20**

	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	20	20	20	20
2	0					
3	0					

KNAPSACK EXAMPLE

i	wi	w	wi > w	M [i, w]
2	2	1	2 > 1 **TRUE**	**M[i, w] = M[i-1, w]** **= M [1, 1]** **= 0**
		2	2 > 2 **FALSE**	**= Max {M[i-1, w], vi + M[i-1, w-wi] }** **= Max (M [1 , 2] , 10 + M [1 , 0])** **= Max (20 , 10)** **= 20**

	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	20	20	20	20
2	0	0	20			
3	0					

KNAPSACK EXAMPLE

i	wi	w	wi > w	M [i, w]
2	2	3	2 > 3 **FALSE**	**= Max {M[i-1, w], vi + M[i-1, w-wi] }** **= Max (M [1 , 3] , 10 + M [1 , 1])** **= Max (20 , 10)** **= 20**
		4	2 > 4 **FALSE**	**= Max {M[i-1, w], vi + M[i-1, w-wi] }** **= Max (M [1 , 4] , 10 + M [1 , 2])** **= Max (20 , 30)** **= 30**

	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	20	20	20	20
2	0	0	20	20	30	
3	0					

KNAPSACK EXAMPLE

i	wi	w	wi > w	M [i, w]
2	2	5	2 > 5 **FALSE**	**= Max {M[i-1, w], vi + M[i-1, w-wi] }** **= Max (M [1 , 5] , 10 + M [1 , 3])** **= Max (20 , 30)** **= 30**

	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	20	20	20	20
2	0	0	20	20	30	30
3	0					

KNAPSACK EXAMPLE

i	wi	w	wi > w	M [i, w]
3	3	1	3 > 1 **TRUE**	**M[i, w] = M[i-1, w]** **= M [2, 1]** **= 0**
		2	3 > 2 **TRUE**	**M[i, w] = M[i-1, w]** **= M [2 , 2]** **= 20**

	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	20	20	20	20
2	0	0	20	20	30	30
3	0	0	20			

KNAPSACK EXAMPLE

i	wi	w	wi > w	M [i, w]
3	3	3	3 > 3 **FALSE**	**= Max {M[i-1, w], vi + M[i-1, w-wi] }** **= Max (M [2 , 3] , 30 + M [2 , 0])** **= Max (20 , 30)** **= 30**
		4	3 > 4 **FALSE**	**= Max {M[i-1, w], vi + M[i-1, w-wi] }** **= Max (M [2 , 4] , 30 + M [2 , 1])** **= Max (30 , 30)** **= 30**

	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	20	20	20	20
2	0	0	20	20	30	30
3	0	0	20	30	30	

KNAPSACK EXAMPLE

i	wi	w	wi > w	M [i, w]
3	3	5	3 > 5 **FALSE**	**= Max {M[i-1, w], vi + M[i-1, w-wi] }** **= Max (M [2 , 5] , 30 + M [2 , 2])** **= Max (30 , 30 + 20)** **= Max (30 , 50)** **= 50**

	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	20	20	20	20
2	0	0	20	20	30	30
3	0	0	20	30	30	50

KNAPSACK EXAMPLE

ITEMS	WEIGHTS	VALUES	W = 5
1	2	20	
2	2	10	
3	3	30	

	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	20	20	20	20
2	0	0	20	20	30	30
3	0	0	20	30	30	50

SOLUTION

FIND ITEMS IN KNAPSACK

- In order to find the items that belong to the knapsack, we use the global array M[n,W] computed and find the items using the following algorithm.

Algorithm: Find_items_in_knapsack()

// Input: Global array M[n,W] giving the maximum value achievable for "n" set of items and knapsack capacity "W".

//Output: The items "i" that belong to the Knapsack.

```
i = n , k = W

while  i, k > 0
        if M[i, k] ≠ M[i-1, k] then
                mark the ith item as in the knapsack
                i = i-1
                k = k-wi
        else
                i = i-1
```

EXAMPLE
FIND ITEMS IN KNAPSACK

ITEMS	WEIGHTS	VALUES	
1	2	20	W = 5
2	2	10	
3	3	30	

	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	20	20	20	20
2	0	0	20	20	30	30
3	0	0	20	30	30	50

EXAMPLE
FIND ITEMS IN KNAPSACK

<table>
<tr><th>i</th><th>k</th><th>Global Array M[n, w]</th><th>Items in the Knapsack</th></tr>
<tr><td>3</td><td>5</td><td>
<table>
<tr><td></td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>20</td><td>20</td><td>20</td><td>20</td></tr>
<tr><td>2</td><td>0</td><td>0</td><td>20</td><td>20</td><td>30</td><td>30</td></tr>
<tr><td>3</td><td>0</td><td>0</td><td>20</td><td>30</td><td>30</td><td>50</td></tr>
</table>
</td><td>3</td></tr>
<tr><td></td><td></td><td>k = k – wi = 5 – 3 =2
i = i – 1 = 2</td><td></td></tr>
</table>

EXAMPLE
FIND ITEMS IN KNAPSACK

<table>
<tr><th>i</th><th>k</th><th>Global Array M[n, w]</th><th>Items in the Knapsack</th></tr>
<tr><td>2</td><td>2</td><td>
<table>
<tr><th></th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td><u>20</u></td><td>20</td><td>20</td><td>20</td></tr>
<tr><td>2</td><td>0</td><td>0</td><td><u>20</u></td><td>20</td><td>30</td><td>30</td></tr>
<tr><td>3</td><td>0</td><td>0</td><td>20</td><td>30</td><td>30</td><td>50</td></tr>
</table>
</td><td>3</td></tr>
<tr><td></td><td></td><td>i = i – 1 = 2 – 1
i = 1</td><td></td></tr>
</table>

EXAMPLE
FIND ITEMS IN KNAPSACK

<table>
<tr><th>i</th><th>k</th><th>Global Array M[n, w]</th><th>Items in the Knapsack</th></tr>
<tr><td>1</td><td>2</td><td>
<table>
<tr><th></th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>20</td><td>20</td><td>20</td><td>20</td></tr>
<tr><td>2</td><td>0</td><td>0</td><td>20</td><td>20</td><td>30</td><td>30</td></tr>
<tr><td>3</td><td>0</td><td>0</td><td>20</td><td>30</td><td>30</td><td>50</td></tr>
</table>
</td><td>3, 1</td></tr>
<tr><td></td><td></td><td>k = k – 2 = 0
i = i – 1 = 0</td><td></td></tr>
</table>

THANK YOU