0/1 Knapsack

November 30, 2020

The Problem

- A knapsack with capacity W and n items with weights $w_1, w_2, , w_n$ and values $v_1, v_2, \ldots, v_n$ are given. The problem is to add items to the knapsack such that the total weight of the added items is $\leq W$ and the total value is maximized.
- In 0/1 Knapsack we cannot split an item. That is we must take item fully or none. Greedy algorithm not possible.

Optimal Substructure

- Let $T = \{s_1, s_2 \dots s_k\}$ be an optimal solution for the knapsack problem with capacity W.
- Then $T - \{s_k\}$ is an optimal solution for the subproblem $s_{k-1} = \{s_1, s_2 \dots s_{k-1}\}$ and knapsack with capacity $W - W_k$
- We use a matrix C to store the results of sub problems.
- The entry $C[i, w]$ denote the optimal solution for the items $1, 2 \dots i$ and weight $0 \leq w \leq W$

Populating the matrix

- In order to add the item k we need to consider three cases
  1. The Knapsack cannot accommodate the item k.
     In this case $c[k, w] = c[k-1, w]$
  2. The knapsack can accommodate the item, but it is not a good choice to add the item
     In this case $c[k, w] = c[k-1, w]$
  3. The knapsack can accommodate the item and it is a good choice to add the item.
     In this case $c[k, w] = v_k + c[k-1, w-w_k]$
- By Finding the max among the cases 2 and 3 we can decide whether the addition is a good choice

An Example

Item	Value	Weight
1	1	1
2	6	2
3	18	5
4	22	6
5	28	7

- Let the capacity of the Knapsack be 11
- We create a matrix with $W + 1 = 12$ columns and $n + 1 = 6$ rows
- The first row and column are initialized with zeros.

An Example

Item	Value	Weight
1	1	1
2	6	2
3	18	5
4	22	6
5	28	7

- To fill the entry $c[i, w]$ we use the following conditions.
  If $w_i > w$ then $c[i, w] = c[i-1, w]$
  else $c[i, j] =$
  $max(c[i-1, w], v_i + c[i-1, w-w_i])$

W + 1

n + 1

	0	1	2	3	4	5	6	7	8	9	10	11
ϕ	0	0	0	0	0	0	0	0	0	0	0	0
{ 1 }	0	1	1	1	1	1	1	1	1	1	1	1
{ 1, 2 }	0	1	6	7	7	7	7	7	7	7	7	7
{ 1, 2, 3 }	0	1	6	7	7	18	19	24	25	25	25	25
{ 1, 2, 3, 4 }	0	1	6	7	7	18	22	24	28	29	29	40
{ 1, 2, 3, 4, 5 }	0	1	6	7	7	18	22	28	29	34	35	40