CMPS 2200 – Fall 2017

Knapsack Problem

Carola Wenk

Knapsack Problem

- Given a knapsack with weight capacity $W > 0$, and given n items of positive integer weights $w_1, \ldots, w_n$ and positive integer values $v_1, \ldots, v_n$.
(So, item i has value v_i and weight w_i.)

- **0-1 Knapsack Problem:** Compute a subset of items that maximize the total value (sum), and they all fit into the knapsack (total weight at most W).

- **Fractional Knapsack Problem:** Same as before but we are allowed to take fractions of items (→ gold dust).

Greedy Knapsack

- Greedy Strategy:
  - Compute $\frac{v_i}{w_i}$ for each i
  - Greedily take as much as possible of the item with the highest value/weight. Then repeat/recurse.

$\Rightarrow$ Sort items by value/weight

$\Rightarrow$ $O(n \log n)$ runtime

Knapsack Example

item	1	2	3	
value	12	15	4	W=4
weight	2	3	1	
value/weight	6	5	4	

- **Greedy fractional:** Take item 1 and 2/3 of item 2
$\Rightarrow$ weight=4, value=$12+2/3\cdot 15 = 12+10 = 22$

- **Greedy 0-1:** Take item 1 and then item 3
$\Rightarrow$ weight = $1+2=3$, value=$12+4=16$

greedy 0-1
$\neq$ optimal 0-1

- **Optimal 0-1:** Take items 2 and 3, value =19

Optimal Substructure

- Let $s_1, \dots, s_n$ be an optimal solution, where $s_i =$ amount of item i that is taken; $0 \leq s_i \leq 1$
- Suppose we remove one item. → $n - 1$ items left
- Is the remaining "solution" still an optimal solution for $n - 1$ items?
- Yes; cut-and-paste.

Correctness Proof for Greedy

- Suppose items $1, \ldots, n$ are numbered in decreasing order by value/weight.
- Greedy solution G: Takes all elements $1, \ldots, j, \ldots, i^*$-1 and a fraction of i^*.
- Assume optimal solution S is different from G. Assume S takes only a fraction $\frac{1}{a}$ of item j, for $j \leq i^*$-1.
- Create new solution S' from S by taking $w_j - 1/a$ weight away from items $> j$, and add $w_j - 1/a$ of item j back in. Hence, all of item j is taken.

$\Rightarrow$ New solution S' has the same weight but increased value. This contradicts the assumption that S was optimal.
$\Rightarrow$ S=G. □

General Solution: DP

- $D[i,w]$ = max value possible for taking a subset of items $1, \ldots, i$ with knapsack constraint w.

- $D[0,w] = D[i,0] = 0$ for all $0 \le i \le n$ and $0 \le w \le W$
  $D[i,w] = -\infty$ for $w < 0$
  $$D[i,w] = \max(\underbrace{D[i-1,w]}_{\text{don't take item } i}, \underbrace{v_i + D[i-1, w-w_i]}_{\text{take item } i})$$

- Compute $D[n,W]$ by filling an $n \times W$ DP-table.
  $\Rightarrow$ Two nested for-loops, runtime and space $\Theta(nW)$

- Trace back from $D[n,W]$ by redoing computation or following arrows. $\Rightarrow$ $\Theta(n+W)$ runtime

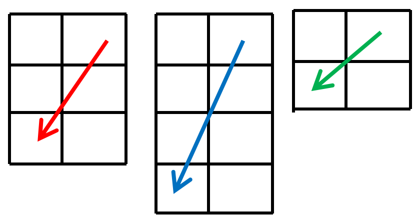
DP Example

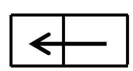
Solution:
Take items 3 and 2

W=4

item	1	2	3
value	12	15	4
weight	2	3	1
value/weight	6	5	4

w ↑				
W= 4	0	12	15	19
3	0	12	15	16
2	0	12	12	12
1	0	0	0	4
0	0	← 0	0	0
	0	1	2	3 (n) → i

Take item i: 

Don't take item i: 

$$D[i,w] = \max(\underbrace{D[i-1,w]}_{\text{don't take item } i}, \underbrace{v_i + D[i-1,w-w_i]}_{\text{take item } i})$$