Section 3.1: The Knapsack Problem

Knapsack

Input:

Knapsack with a capacity $B \in \mathbb{Z}^+$

Items $I = \{1, 2, \dots, n\}$

Item i has size $s_i \in \mathbb{Z}^+$ and value $v_i \in \mathbb{Z}^+$

Objective:

Find a set of items with total size $\leq B$ and largest possible total value

Greedy alg.:

Consider items in order of decreasing v/s-ratio

Does not have any constant approx. ratio:

Ex:

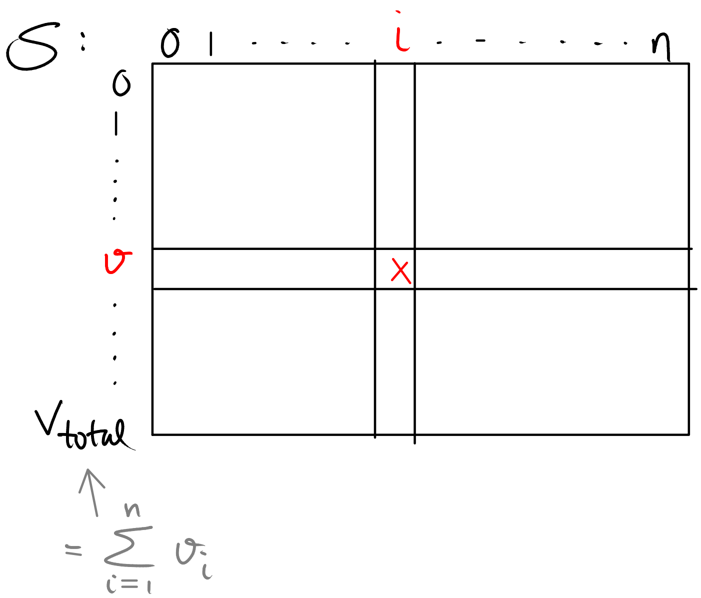
$$s_1 = v_1 = 1 \longrightarrow v_1/s_1 = 1$$

$$s_2 = B, \quad v_2 = B-1 \longrightarrow v_2/s_2 = 1 - \frac{1}{B}$$

$$\text{Greedy} = 1$$

$$\text{OPT} = B-1$$

Dynamic prg alg:



$S_{v,i}$: smallest possible total size of subset of items $1, \dots, i$ with total value v.

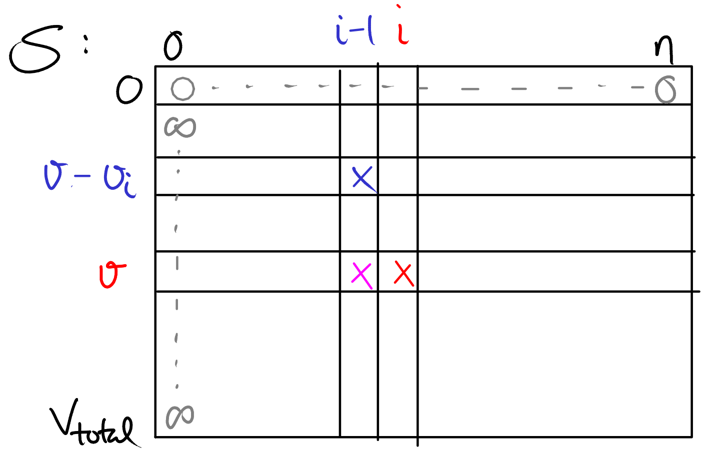
Ex: $B = 5$

	1	2	3
size	2	4	2
value	3	2	1

	0	1	2	3	
0	0	0	0	0	
1	∞	∞	∞	2	
2	∞	∞	4	4	
3	∞	2	2	2	
4	∞	∞	∞	4	$\leq B$
5	∞	∞	6	6	$> B$
6	∞	∞	∞	8	$> B$

largest possible total value $\rightarrow$ 4

What are the rules for filling the table?



$S_{v,i}$: smallest possible total size of subset of items $1, \dots, i$ with total value v.

If $i = 0$ and $1 \le v \le V_{total}$

$$S_{v,i} = \infty$$

Otherwise,

$$S_{v,i} = \begin{cases} 0, & \text{if } v = 0 \\ S_{v,i-1}, & \text{if } 0 < v < v_i \\ \min\{\underbrace{S_{v,i-1}}_{\substack{\text{best solution} \\ \text{\underline{without} item } i}}, \underbrace{S_{v-v_i,i-1} + s_i}_{\substack{\text{best solution} \\ \text{\underline{with} item } i}}\}, & \text{if } v \ge v_i \end{cases}$$

How do we determine which items to select to obtain the optimal value?

$i-1 \quad i$

$\nwarrow$ include item i in the solution

$i-1 \quad i$

$\leftarrow$ leave out item i

We don't have to store all columns, and we don't necessarily have to fill in all entries:

Alg 3.1:

$A[1] \leftarrow \{(0,0), (s_1, v_1)\}$

For $i \leftarrow 2$ to n

  $A[i] \leftarrow A[i-1]$

  For each $(S,V) \in A_{prev}$

    If $S + s_i \le B$

      $A[i] \leftarrow A[i] \cup \{(S+s_i, V+v_i)\}$

  Remove dominated pairs from $A[i]$

Return $\max\limits_{(S,V)\in A[n]} \{V\}$

Note that the alg. returns only the value of an optimal solution, not the corresponding set of items.

It is, however, possible to find the optimal solution set in asymptotically the same time and in $O(V_{total})$ space.

Analysis:

Running time: $O(n \cdot V_{total})$

Input size: $O(\log B + n(\log M + \log S))$, where

$M = \max\limits_{1 \le i \le n} \{v_i\}$ and $S = \max\limits_{1 \le i \le n} \{s_i\}$.

Poly. time?

Ex: $V_{total} = 2^n$

$\log B = \log M = \log S = n$

$\Rightarrow$ Running time $O(n \cdot 2^n)$

Input size $O(n^2)$

No.

But if the numeric part of the input (i.e., B, v_i, s_i) were written in unary, the input size would be $\Theta(B + V_{total} + S_{total})$, and the running time would be poly. in the input size.

Hence, the running time is pseudopolynomial.

Note: if V_{total} is poly. in n for all possible input instances, the dyn. prg. alg. is poly.

Leading to the following idea.

Idea for approximation algorithm:

Round values s.t. there are only a poly. number of (equidistant) values:

- Choose a value μ
- Round down each item value to the nearest multiple of μ
- Do dyn. prg. on the rounded values

How to choose μ?

- Approximation:

  When rounding, each item looses a value of less than μ. Hence, the value of any solution is changed by less than $n\mu$.

  Thus, if we want a precision of ε,
  $$\mu = \frac{\varepsilon M}{n}$$
  will do, since then $n\mu = \varepsilon M \leq \varepsilon \cdot OPT$.

  (We will add more detail to this argument in the proof of Thm 3.5.)

- Running time:
  $$n \cdot \frac{V_{total}}{\mu} \leq n \cdot \frac{nM}{\mu} = \frac{1}{\varepsilon} \cdot n^3$$

Since each rounded value is a multiple of μ, we might as well scale by a factor of $\frac{1}{\mu}$ s.t. the possible values will be $1, 2, \ldots, \lfloor \frac{V_{total}}{\mu} \rfloor$ instead of $\mu, 2\mu, \ldots, \lfloor \frac{V_{total}}{\mu} \rfloor \mu$:

Alg 3.2

$M \leftarrow \max_{1 \le i \le n} v_i$

$\mu \leftarrow \frac{\varepsilon M}{n}$

for $i \leftarrow 1$ to n

  $v_i' \leftarrow \lfloor \frac{v_i}{\mu} \rfloor$

Do dyn. prg. with values v_i' (and sizes s_i)

Theorem 3.5

Alg. 3.2 is a $(1-\varepsilon)$-approx. alg. with a running time poly. in both input size and $\frac{1}{\varepsilon}$