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 s/v ratio

Does not have any constant approximation factor:

<u>Ex</u>:

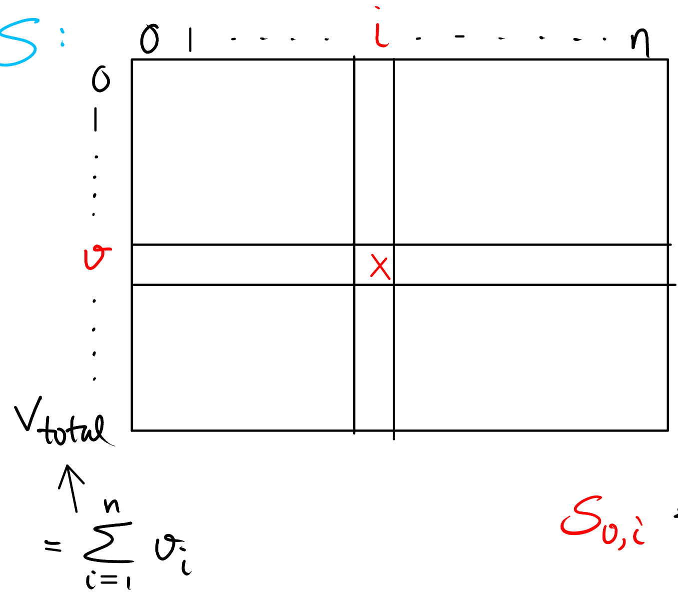
i	1	2
v_i	1	$B-1$
s_i	1	B

$$\frac{v_1}{s_1} = 1 > \frac{v_2}{s_2} = 1 - \frac{1}{B}$$

$$\Rightarrow \text{Greedy} = 1 = \frac{1}{B-1} \cdot OPT$$

Dynamic prg alg:

S:



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

$$S_{v,i} = \min_{I \subseteq \{1,\dots,i\}} \left\{ \sum_{j\in I} s_j \;\middle|\; \sum_{j \in I} v_j = v \right\}$$

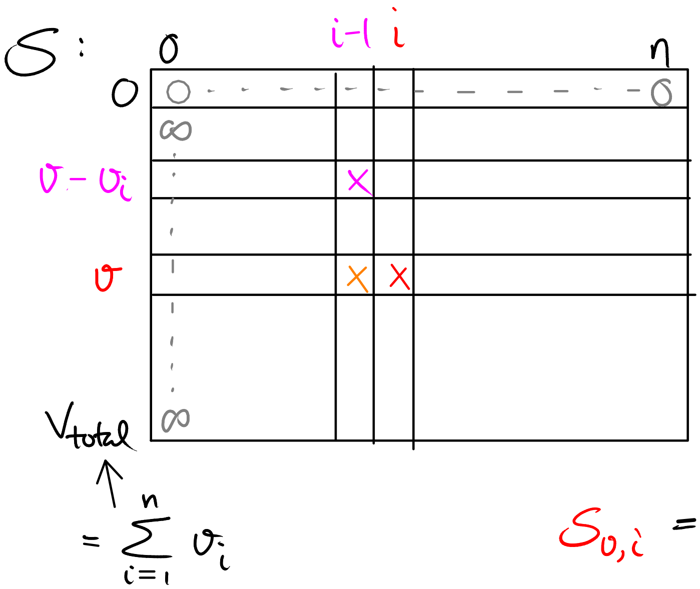
Ex:

$B=5$

i	1	2	3
v_i	3	1	2
s_i	2	3	2

v_i \ i	0	1	2	3	
0	0	0	0	0	
1	∞	∞	3	3	
2	∞	∞	∞	2	
3	∞	2	2	2	
4	∞	∞	5	5	
Optimal total value $\rightarrow$ 5	∞	∞	∞	4	$\leq B$
6	∞	∞	∞	7	$> B$

How to fill the table



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

$$S_{v,i} = \min_{I \subseteq \{1,\dots,i\}} \left\{ \sum_{j \in I} s_j \;\middle|\; \sum_{j \in I} v_j = v \right\}$$

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

$S_{v,i} = \infty$

Otherwise,

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

Not necessary to fill in the ∞-entries (A[i] corresponds to column i):

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 $\max_{(S,V)\in A[n]} \{V\}$

$B=5$

i	1	2	3
v_i	3	1	2
s_i	2	3	2

$A[1] = \{(0,0), (3,2)\}$

$A[2] = A[1] \cup \{(1,3), (4,5)\}$
$= \{(0,0), (1,3), (3,2), (4,5)\}$

$A[3] = A[2] \cup \{(2,2), (3,5), (5,4)\}$
$= \{(0,0), (1,3), (2,2), (3,2), \cancel{(3,5)}, (4,5), (5,4)\}$

dominated by

Analysis

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

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

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

Poly. time?

Ex: Consider a family of instances where

$V_{total} = 2^n$ and

$B, S \le 2^n$

Then

Running time $T(n) \in \Omega(n \cdot 2^n)$ and

Input size $S(n) \in O(n^2)$

$\Rightarrow T(n) \in \Omega((S(n))^{c\sqrt{n}})$

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

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

$M \leftarrow \max\limits_{1 \le i \le n} v_i$

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

for $i \leftarrow 1$ to n

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

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

Theorem 3.5

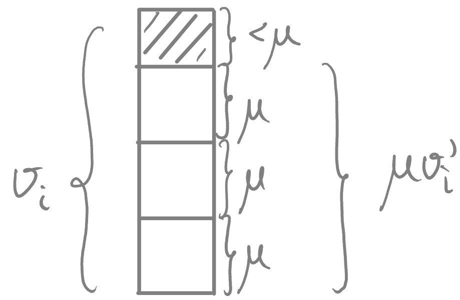
Alg. 3.2 is a $(1-\varepsilon)$-approx. alg. with a running time poly. in both input size and $\frac{1}{\varepsilon}$

Proof:

Approximation ratio:

For each item i, $\mu v_i'$ equals v_i rounded down to the nearest multiple of μ. (*)

Thus, $v_i - \mu v_i' < \mu$ (each item „loses" less than μ in the rounding.) (**)



Let A be the set of items selected by Alg. 3.2

This is an optimal solution to the instance with values v_i', and hence, to the instance with values $\mu v_i'$. (***)

Let O be the set of items in an optimal solution to the original instance with values v_i.

The total value produced by Alg. 3.2 is

$$
\begin{aligned}
\sum_{i \in A} v_i &\geq \sum_{i \in A} \mu v_i', && \text{by } (*) \\
&\geq \sum_{i \in O} \mu v_i', && \text{by } (***) \\
&> \sum_{i \in O} (v_i - \mu), && \text{by } (**) \\
&\geq \left(\sum_{i \in O} v_i \right) - n\mu, && \text{since } |O| \leq n \\
&= OPT - n \cdot \frac{\varepsilon M}{n} \\
&= OPT - \varepsilon M \\
&\geq (1-\varepsilon)\, OPT, && \text{since } OPT \geq M
\end{aligned}
$$

Running time:

$O(\frac{1}{\varepsilon} \cdot n^3)$ as proven above.

$\square$

According to Thm 3.5, Alg. 3.2 is a

fully polynomial time approximation scheme (FPTAS)

fully: also poly. in $1/\varepsilon$

polynomial time: poly. in input size

approximation scheme: Family $\{A_\varepsilon\}$ of alg., where A_ε has precision ε.

($(1-\varepsilon)$-approx. alg for max. problems,
$(1+\varepsilon)$-approx. alg for min. problems)

fully: Def. 3.4

polynomial time approximation scheme: Def. 3.3

Thus, Thm 3.5 could also be stated like this:

$\boxed{\text{Theorem 3.5: Alg 3.2 is a FPTAS}}$

Multiple Knapsack problem: Fixed #knapsacks

Bin Packing can be seen as a dual version of Multiple Knapsack.

Bin Packing

Input: n items with sizes between 0 and 1.

Objective: Pack items in bins of size 1, using as few bins as possible.

Simple approximation algorithms:

Alg.	Running time	Asymp. approx. factor
Next-Fit	$O(n)$	2
First-Fit	$O(n \log n)$	1.7
Best-Fit	— " —	— " —
Next-Fit-Decreasing	— " —	≈ 1.69
First-Fit-Decreasing	— " —	$1.\overline{2}$
Best-Fit-Decreasing	— " —	— " —

Approximation scheme?

Can we do the same kind of rounding for Bin Packing as we did for Knapsack?