Contents

1. Greedy algorithms
   - Introduction
   - Interval scheduling problem
   - Scheduling all Intervals problem
   - Fractional knapsack problem
   - Coin changing problem
   - What problems can be solved by greedy approach?
   - Conclusion

Fractional knapsack problem

Definition (fractional knapsack problem)

Given a set S of n items, such that each item i has a positive benefit b_i and a positive weight w_i, the goal is to find the maximum-benefit subset that does not exceed a given weight W, allowing for fractional items.

Fractional knapsack problem

Definition (fractional knapsack problem)

Given a set S of n items, such that each item i has a positive benefit b_i and a positive weight w_i, the goal is to find the maximum-benefit subset that does not exceed a given weight W, allowing for fractional items.

- Taking x_i of each item i, such that $0 \leq x_i \leq w_i$ for each $i \in S$, and $\sum_{i \in S} x_i \leq W$.

Fractional knapsack problem

Definition (fractional knapsack problem)

Given a set S of n items, such that each item i has a positive benefit b_i and a positive weight w_i, the goal is to find the maximum-benefit subset that does not exceed a given weight W, allowing for fractional items.

- Taking x_i of each item i, such that $0 \leq x_i \leq w_i$ for each $i \in S$, and $\sum_{i \in S} x_i \leq W$.
- Benefit for taking x_i of item i is then $b_i(x_i/w_i)$

Fractional knapsack problem

Definition (fractional knapsack problem)

Given a set S of n items, such that each item i has a positive benefit b_i and a positive weight w_i, the goal is to find the maximum-benefit subset that does not exceed a given weight W, allowing for fractional items.

- Taking x_i of each item i, such that $0 \leq x_i \leq w_i$ for each $i \in S$, and $\sum_{i \in S} x_i \leq W$.
- Benefit for taking x_i of item i is then $b_i(x_i/w_i)$
- Maximum-benefit subset is then maximizing $\sum_{i \in S} b_i(x_i/w_i)$.

Fractional knapsack problem

Definition (fractional knapsack problem)

Given a set S of n items, such that each item i has a positive benefit b_i and a positive weight w_i, the goal is to find the maximum-benefit subset that does not exceed a given weight W, allowing for fractional items.

- Taking x_i of each item i, such that $0 \leq x_i \leq w_i$ for each $i \in S$, and $\sum_{i \in S} x_i \leq W$.
- Benefit for taking x_i of item i is then $b_i(x_i/w_i)$
- Maximum-benefit subset is then maximizing $\sum_{i \in S} b_i(x_i/w_i)$.

Key question

- What strategy to use to select the next item (and the amount of it)?

Fractional knapsack problem

Definition (fractional knapsack problem)

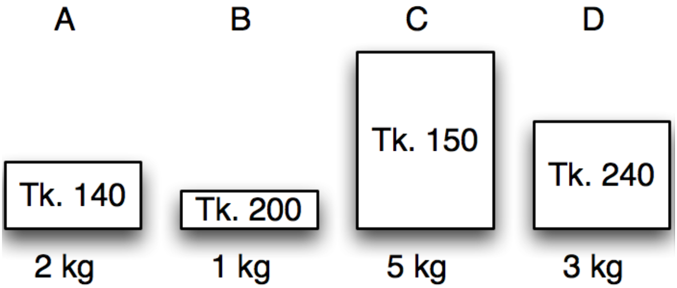
Given a set S of n items, such that each item i has a positive benefit b_i and a positive weight w_i, the goal is to find the maximum-benefit subset that does not exceed a given weight W, allowing for fractional items.

- Taking x_i of each item i, such that $0 \leq x_i \leq w_i$ for each $i \in S$, and $\sum_{i \in S} x_i \leq W$.
- Benefit for taking x_i of item i is then $b_i(x_i/w_i)$
- Maximum-benefit subset is then maximizing $\sum_{i \in S} b_i(x_i/w_i)$.

Key question

- What strategy to use to select the next item (and the amount of it)?
- Since we're maximizing the benefit, select the next item with the highest benefit per weight – b_i/w_i.

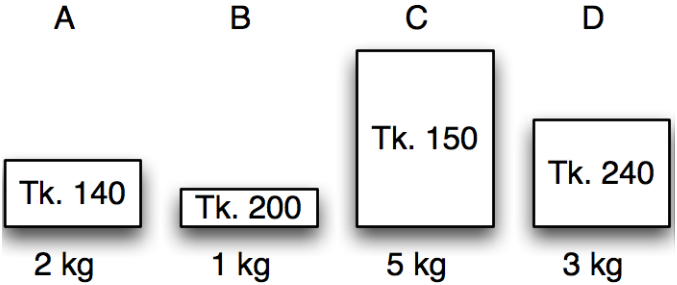
Fractional knapsack in action



Item	Price	Weight
A	140	2 kg
B	200	1 kg
C	150	5 kg
D	240	3 kg

Calculate price/kg – the value index.

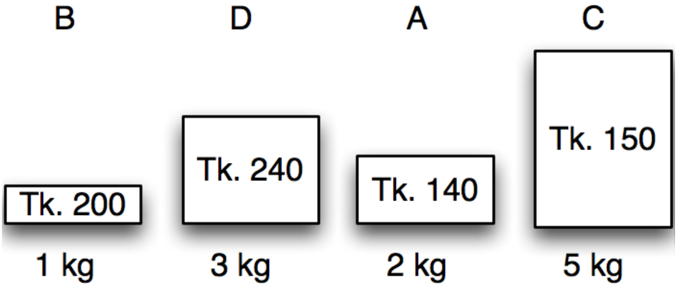
Fractional knapsack in action



Item	Price	Weight	Value index
A	140	2 kg	70
B	200	1 kg	200
C	150	5 kg	30
D	240	3 kg	80

Sort by non-increasing value index.

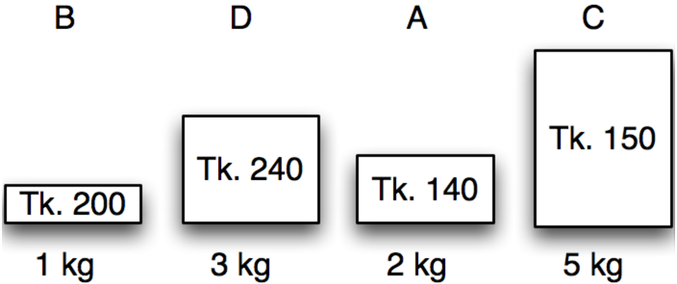
Fractional knapsack in action



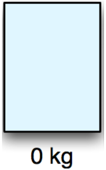
Item	Price	Weight	Value index
B	200	1 kg	200
D	240	3 kg	80
A	140	2 kg	70
C	150	5 kg	30

Maximum weight: 5 kg

Fractional knapsack in action

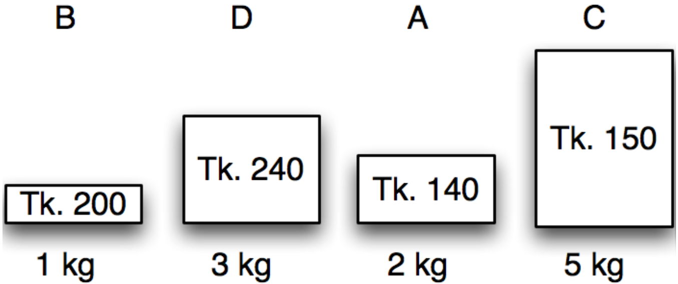


Item	Price	Weight	Value index	Chosen
B	200	1 kg	200	0 kg
D	240	3 kg	80	0 kg
A	140	2 kg	70	0 kg
C	150	5 kg	30	0 kg

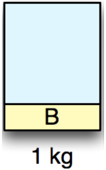


Maximum weight: 5 kg Remaining: 5 kg Benefit: 0 kg

Fractional knapsack in action

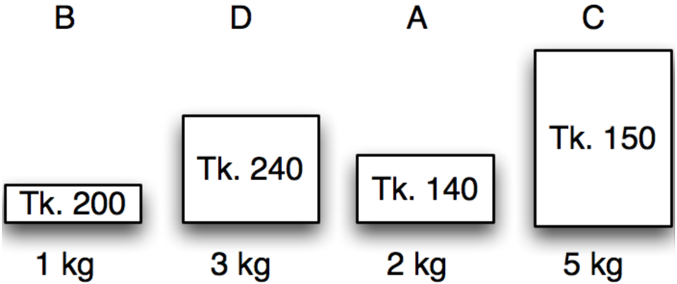


Item	Price	Weight	Value index	Chosen
B	200	1 kg	200	1 kg
D	240	3 kg	80	0 kg
A	140	2 kg	70	0 kg
C	150	5 kg	30	0 kg

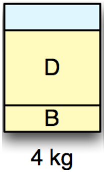


Maximum weight: 5 kg Remaining: 4 kg Benefit: 200 kg

Fractional knapsack in action

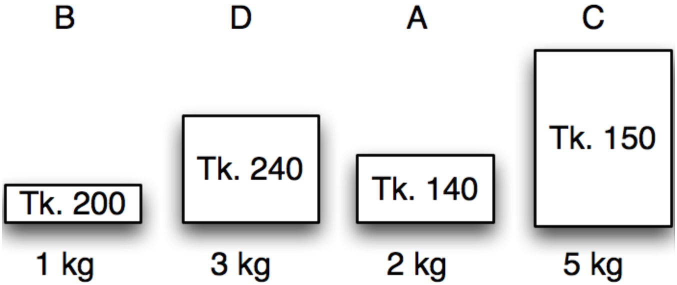


Item	Price	Weight	Value index	Chosen
B	200	1 kg	200	1 kg
D	240	3 kg	80	3 kg
A	140	2 kg	70	0 kg
C	150	5 kg	30	0 kg

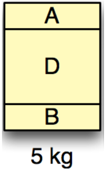


Maximum weight: 5 kg Remaining: 1 kg Benefit: 440 kg

Fractional knapsack in action

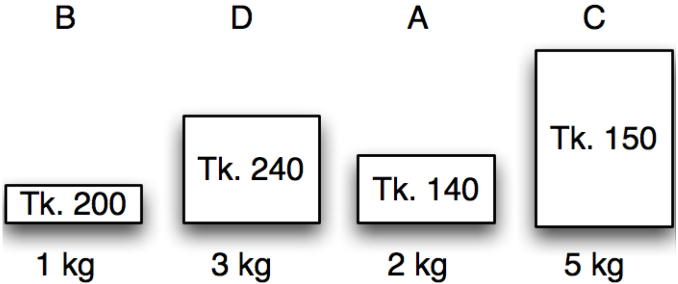


Item	Price	Weight	Value index	Chosen
B	200	1 kg	200	1 kg
D	240	3 kg	80	3 kg
A	140	2 kg	70	1 kg
C	150	5 kg	30	0 kg

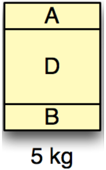


Maximum weight: 5 kg Remaining: 0 kg Benefit: 510 kg

Fractional knapsack in action



Item	Price	Weight	Value index	Chosen
B	200	1 kg	200	1 kg
D	240	3 kg	80	3 kg
A	140	2 kg	70	1 kg
C	150	5 kg	30	0 kg



Maximum weight: 5 kg Remaining: 0 kg Benefit: 510 kg

Fractional knapsack greedy algorithm

```
FRACTIONAL-KNAPSACK$(S, W)$    $\triangleright$ $S = \{(w_i, b_i)\}$
for each item $i \in S$
      do $x_i \leftarrow 0$    $\triangleright$ amount of item $i$ chosen $(0 \leq x \leq w_i)$
         $v_i \leftarrow b_i / w_i$    $\triangleright$ compute value index
$w \leftarrow 0$
while $w < W$
      do $i =$ extract from $S$ the item with highest value index
                $\triangleright$ greedy choice
         if $w + w_i \leq W$
            then $x_i = w_i$
            else $x_i = W - w$ $\triangleright$ fill up the remaining with $i$
         $w \leftarrow w + x_i$
return $x$    $\triangleright$ $x_i$ contains amount of item $i$ chosen
```

Fractional knapsack greedy algorithm

```
FRACTIONAL-KNAPSACK($S, W$)    ▷ $S = \{(w_i, b_i)\}$
for each item $i \in S$
    do $x_i \leftarrow 0$    ▷ amount of item $i$ chosen $(0 \le x \le w_i)$
       $v_i \leftarrow b_i/w_i$    ▷ compute *value index*
$w \leftarrow 0$
while $w < W$
    do $i$ = extract from $S$ the item with highest value index
            ▷ greedy choice
       if $w + w_i \le W$
          then $x_i = w_i$
          else $x_i = W - w$ ▷ fill up the remaining with $i$
       $w \leftarrow w + x_i$
return $x$    ▷ $x_i$ contains amount of item $i$ chosen
```

Complexity

$T(n) = O(n \lg n)$.

Fractional knapsack greedy algorithm

```
FRACTIONAL-KNAPSACK$(S, W)$      $\triangleright$ $S = \{(w_i, b_i)\}$
for each item $i \in S$
        do $x_i \leftarrow 0$      $\triangleright$ amount of item $i$ chosen $(0 \le x \le w_i)$
           $v_i \leftarrow b_i / w_i$                  $\triangleright$ compute *value index*
$w \leftarrow 0$
while $w < W$
        do $i =$ extract from $S$ the item with highest value index
                $\triangleright$ greedy choice
           if $w + w_i \le W$
              then $x_i = w_i$
              else $x_i = W - w$ $\triangleright$ fill up the remaining with $i$
           $w \leftarrow w + x_i$
return $x$      $\triangleright$ $x_i$ contains amount of item $i$ chosen
```

Complexity

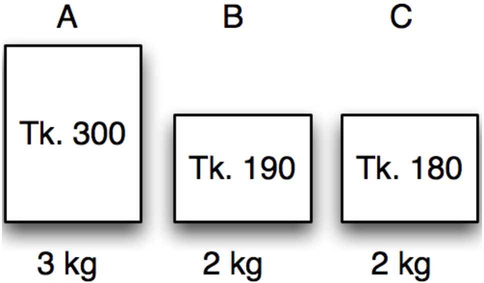
$T(n) = O(n \lg n)$. Prove it.

Extension: 0/1 knapsack problem

Exactly the same as the Fractional Knapsack Problem, except that fractional quantities are not allowed.

Extension: 0/1 knapsack problem

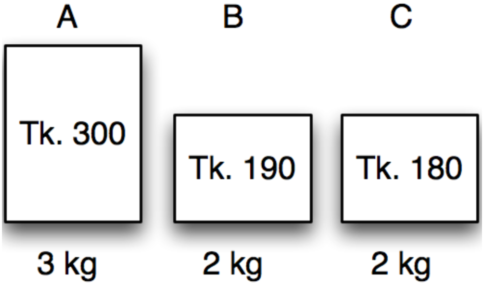
Exactly the same as the Fractional Knapsack Problem, except that fractional quantities are not allowed.



Maximum weight: 4 kg

Extension: 0/1 knapsack problem

Exactly the same as the Fractional Knapsack Problem, except that fractional quantities are not allowed.

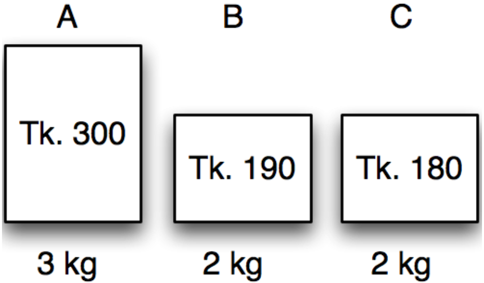


Maximum weight: 4 kg

Greedy solution: item A Benefit: 300

Extension: 0/1 knapsack problem

Exactly the same as the Fractional Knapsack Problem, except that fractional quantities are not allowed.



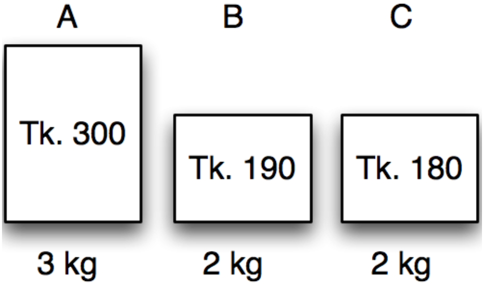
Maximum weight: 4 kg

Greedy solution: item *A* Benefit: 300

Optimal solution: items *B* and *C* Benefit: 370

Extension: 0/1 knapsack problem

Exactly the same as the Fractional Knapsack Problem, except that fractional quantities are not allowed.



Maximum weight: 4 kg

Greedy solution: item A Benefit: 300

Optimal solution: items B and C Benefit: 370

The 0/1 Knapsack Problem does not have a greedy solution!