COMP108
Data Structures and Algorithms

Greedy Algorithm (Part I Knapsack Problem)

Professor Prudence Wong

pwong@liverpool.ac.uk

2022-23

Outline

Greedy algorithms

- What is greedy algorithm?
- See some examples

Learning outcomes:

- Understand what greedy algorithm is
- Able to apply greedy algorithm on the Knapsack problem
- Able to apply Kruskal's algorithm to find minimum spanning tree
- Able to apply Dijkstra's algorithm to find single-source shortest-paths

Coin Change Problem

Suppose we have 3 types of coins

Minimum number of coins to make
£0.8, £1.0, £1.4 ?

Greedy Algorithm

Coin Change Problem

What if we have different types of coins?

Minimum number of coins to make
£0.8, £1.0, £1.4 ?

Greedy Algorithm still works?

Greedy Algorithms

How to be greedy?

- At every step, make the best move you can make
- Keep going until you're done

Greedy Algorithms

How to be greedy?

- At every step, make the best move you can make
- Keep going until you're done

Advantages

- Don't need to pay much effort at each step
- Usually finds a solution very quickly
- The solution found is usually not bad

Greedy Algorithms

How to be greedy?

- At every step, make the best move you can make
- Keep going until you're done

Advantages

- Don't need to pay much effort at each step
- Usually finds a solution very quickly
- The solution found is usually not bad

Possible problem

- The solution found may NOT be the best one

Greedy Algorithms - examples

Greedy algorithms that do NOT find the BEST solution

Coin change problem (previous slide)

Knapsack problem

Greedy algorithms that find the BEST solution

Minimum spanning tree

- Kruskal's algorithm

Single-source shortest-paths

- Dijkstra's algorithm

Knapscak Problem

Knapscak Problem

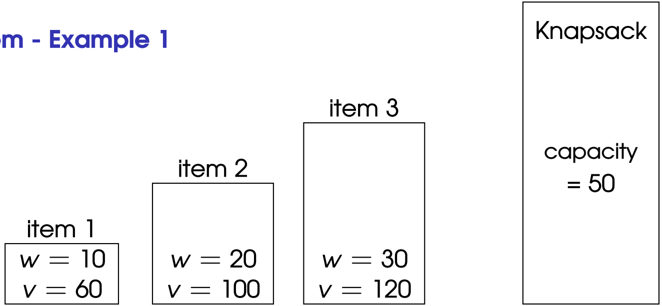
What to take? so that…
1. Not too heavy
2. Most valuable

Knapscak Problem

orthogonal

- Input: Given n items with weights $w_1, w_2, \cdots, w_n$ and values $v_1, v_2, \cdots, v_n$, and a knapsack with capacity W
- Output: Find the **most valuable** subset of items that can **fit** into the knapsack
- Applications: A transport plane is to deliver the most valuable set of items to a remote location without exceeding its capacity

Knapsack problem - Example 1



Knapsack problem - Example 1

item 1: $w = 10$, $v = 60$

item 2: $w = 20$, $v = 100$

item 3: $w = 30$, $v = 120$

Knapsack

capacity = 50

$2^n - 1$

non-empty subset

subset	weight	value
$\emptyset$	0	0
$\{1\}$	10	60
$\{2\}$	20	100
$\{3\}$	30	120
$\{1,2\}$	30	160
$\{1,3\}$	40	180
$\{2,3\}$	**50**	**220**
$\{1,2,3\}$	60	N/A

1	2	3	4	…	n-1	n
1	0	1	1		0	0
0	0	0	0		0	0
1	1	1	1		1	1
1	0	0	0		0	0
0	1	1	0		0	0

2	1
	0
1	1
1	0
0	1
0	0

$2^2 \times 2 \times 2$

2^n

Knapsack problem - Example 2

item 1
$w = 7$
$v = 42$

item 2
$w = 3$
$v = 12$

item 3
$w = 4$
$v = 40$

item 4
$w = 5$
$v = 25$

Knapsack

capacity
= 10

Knapsack problem - Example 2

item 1: $w = 7$, $v = 42$

item 2: $w = 3$, $v = 12$

item 3: $w = 4$, $v = 40$

item 4: $w = 5$, $v = 25$

Knapsack

capacity = 10

subset	weight	value
$\emptyset$	0	0
$\{1\}$	7	42
$\{2\}$	3	12
$\{3\}$	4	40
$\{4\}$	5	25
$\{1, 2\}$	10	54
$\{1, 3\}$	11	N/A
$\{1, 4\}$	12	N/A

subset	weight	value
$\{2, 3\}$	7	52
$\{2, 4\}$	8	37
$\{3, 4\}$	**9**	**65**
$\{1, 2, 3\}$	14	N/A
$\{1, 2, 4\}$	15	N/A
$\{1, 3, 4\}$	16	N/A
$\{2, 3, 4\}$	12	N/A
$\{1, 2, 3, 4\}$	19	N/A

Exhaustive algorithm

Approach

- Try **every subset** of the set of n given items
- Compute total weight of each subset
- Compute total value of those subsets that do NOT exceed knapsack's capacity
- Pick the subset that has the largest value

Exhaustive algorithm

Approach

- Try **every subset** of the set of n given items
- Compute total weight of each subset
- Compute total value of those subsets that do NOT exceed knapsack's capacity
- Pick the subset that has the largest value

Analysis

- How many subsets to consider?

Exhaustive algorithm

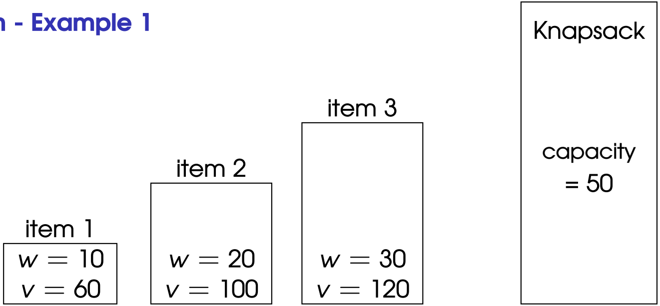
Approach

- Try **every subset** of the set of n given items
- Compute total weight of each subset
- Compute total value of those subsets that do NOT exceed knapsack's capacity
- Pick the subset that has the largest value

Analysis

- How many subsets to consider?
- $2^n - 1$
- Exponential in n
- Why?

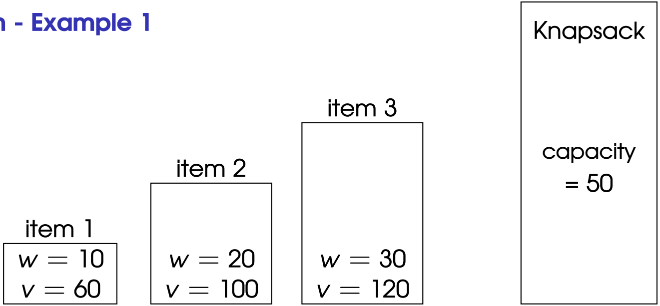
Greedy algorithm - Example 1



Greedy: pick the item with the next largest value if total weight $\leq$ capacity

Result:

Greedy algorithm - Example 1

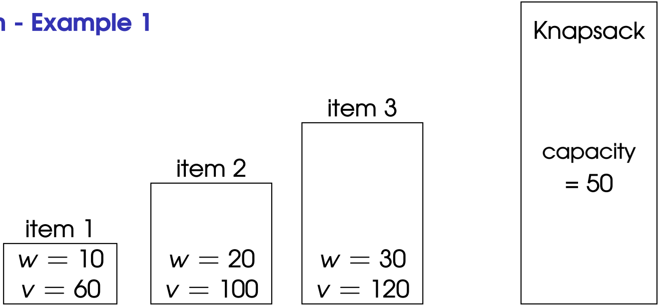


Greedy: pick the item with the next largest value if total weight $\leq$ capacity

Result:

- item 3 is picked, total value = 120, total weight = 30

Greedy algorithm - Example 1

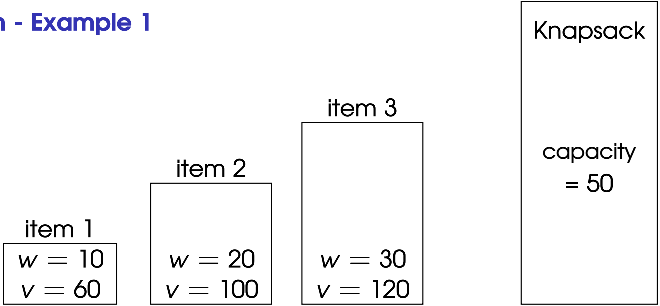


Greedy: pick the item with the next largest value if total weight $\leq$ capacity

Result:

- item 3 is picked, total value = 120, total weight = 30
- item 2 is picked, total value = 220, total weight = 50

Greedy algorithm - Example 1

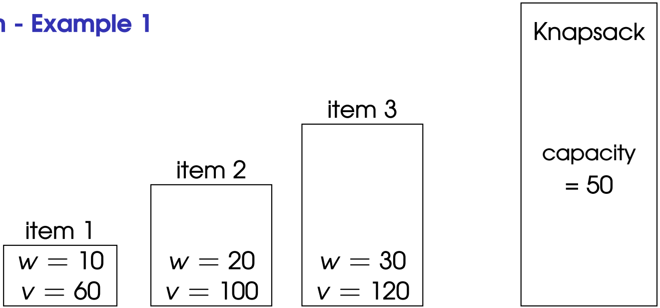


Greedy: pick the item with the next largest value if total weight $\leq$ capacity

Result:

- item 3 is picked, total value = 120, total weight = 30
- item 2 is picked, total value = 220, total weight = 50
- item 1 cannot be picked

Greedy algorithm - Example 1



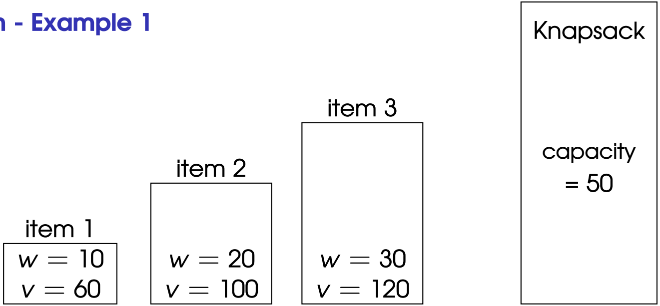
Greedy: pick the item with the next largest value if total weight $\leq$ capacity

Result:

- item 3 is picked, total value = 120, total weight = 30
- item 2 is picked, total value = 220, total weight = 50
- item 1 cannot be picked

Time complexity?

Greedy algorithm - Example 1



Greedy: pick the item with the next largest value if total weight $\leq$ capacity

Result:

- item 3 is picked, total value = 120, total weight = 30
- item 2 is picked, total value = 220, total weight = 50
- item 1 cannot be picked

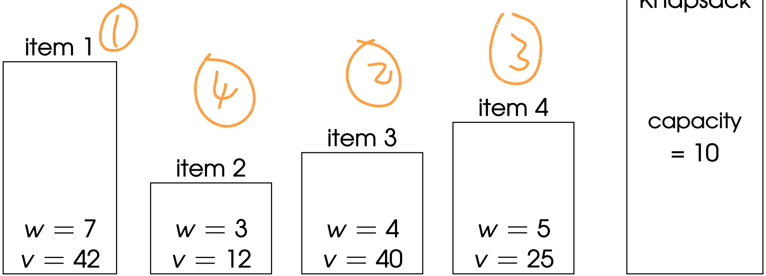
$O(n \log n)$

$O(n^2)$

Time complexity?

Always work?

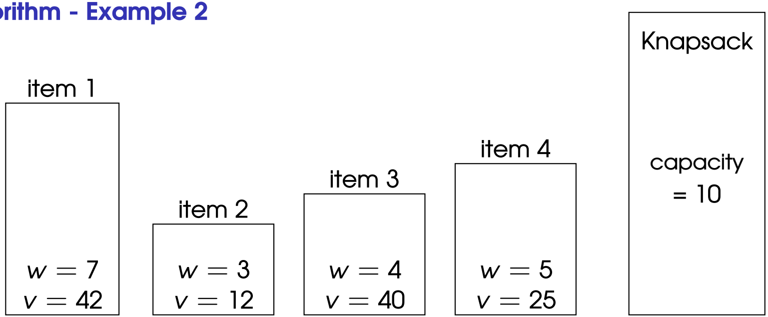
Greedy algorithm - Example 2



Greedy: pick the item with the next largest value if total weight $\leq$ capacity

Result:

Greedy algorithm - Example 2

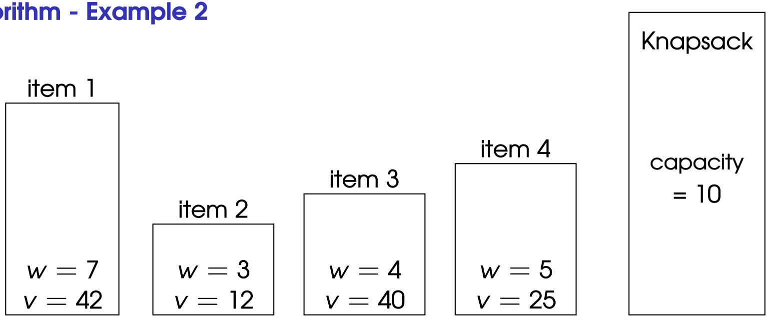


Greedy: pick the item with the next largest value if total weight $\leq$ capacity

Result:

- item 1 is picked, total value = 42, total weight = 7

Greedy algorithm - Example 2

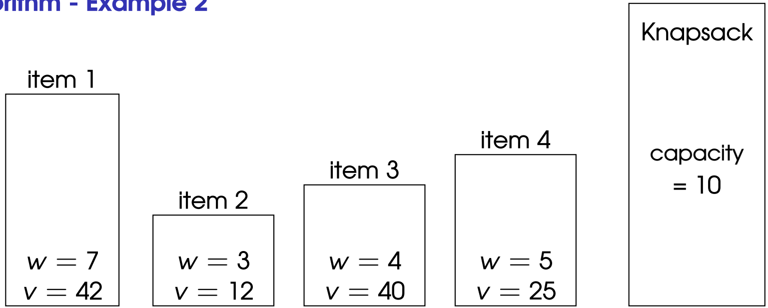


Greedy: pick the item with the next largest value if total weight $\leq$ capacity

Result:

- item 1 is picked, total value = 42, total weight = 7
- item 3 cannot be picked

Greedy algorithm - Example 2

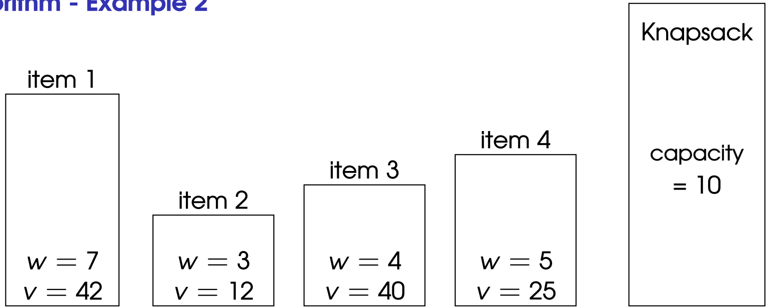


Greedy: pick the item with the next largest value if total weight $\leq$ capacity

Result:

- item 1 is picked, total value = 42, total weight = 7
- item 3 cannot be picked
- item 4 cannot be picked

Greedy algorithm - Example 2

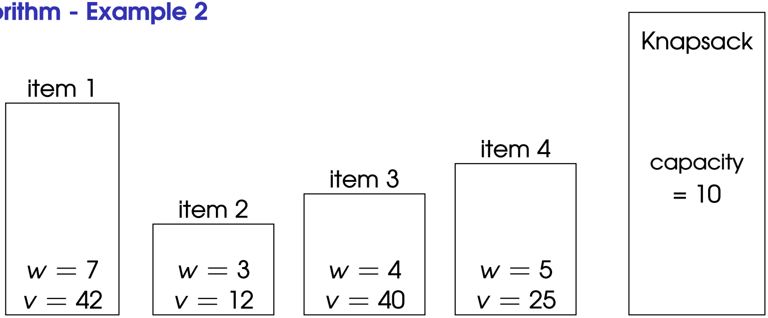


Greedy: pick the item with the next largest value if total weight $\leq$ capacity

Result:

- item 1 is picked, total value = 42, total weight = 7
- item 3 cannot be picked
- item 4 cannot be picked
- item 2 is picked, total value = 54, total weight = 10

Greedy algorithm - Example 2



Greedy: pick the item with the next largest value if total weight $\leq$ capacity

Result:

- item 1 is picked, total value = 42, total weight = 7
- item 3 cannot be picked
- item 4 cannot be picked
- item 2 is picked, total value = 54, total weight = 10

Not the best!

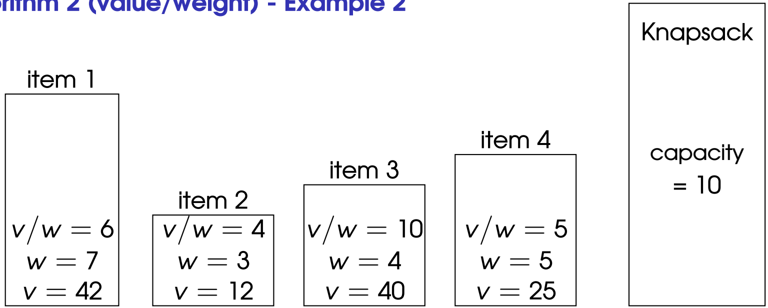
Greedy algorithm 2 (value/weight) - Example 2

item 1 (2)	item 2 (4)	item 3 (1)	item 4 (3)	Knapsack
$v/w = 6$	$v/w = 4$	$v/w = 10$	$v/w = 5$	capacity = 10
$w = 7$	$w = 3$	$w = 4$	$w = 5$	
$v = 42$	$v = 12$	$v = 40$	$v = 25$	

Greedy: pick the item with the next largest (value/weight) if total weight $\leq$ capacity

Result:

Greedy algorithm 2 (value/weight) - Example 2

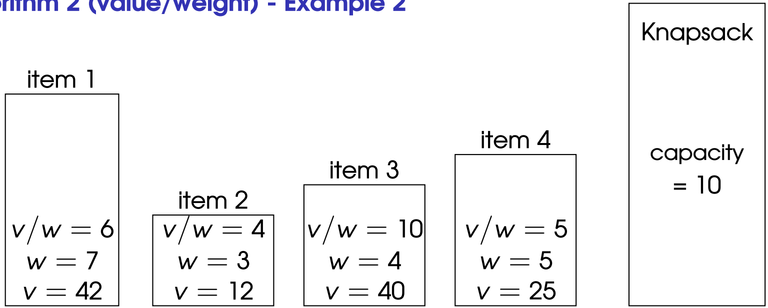


Greedy: pick the item with the next largest (value/weight) if total weight $\leq$ capacity

Result:

- item 3 is picked, total value = 40, total weight = 4

Greedy algorithm 2 (value/weight) - Example 2



Greedy: pick the item with the next largest (value/weight) if total weight $\leq$ capacity

Result:

- item 3 is picked, total value = 40, total weight = 4
- item 1 cannot be picked

Greedy algorithm 2 (value/weight) - Example 2

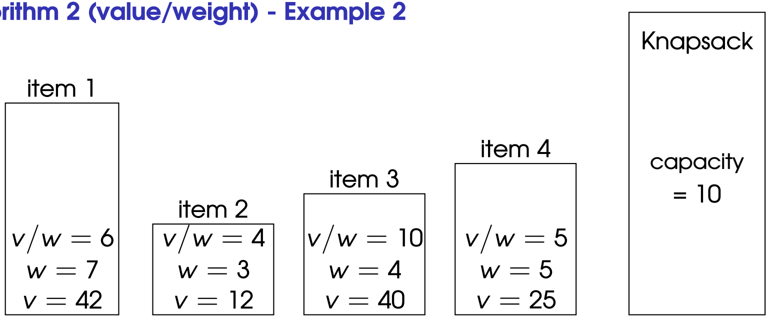
	item 1	item 2	item 3	item 4	Knapsack
	$v/w = 6$	$v/w = 4$	$v/w = 10$	$v/w = 5$	capacity = 10
	$w = 7$	$w = 3$	$w = 4$	$w = 5$	
	$v = 42$	$v = 12$	$v = 40$	$v = 25$	

Greedy: pick the item with the next largest (value/weight) if total weight $\leq$ capacity

Result:

- item 3 is picked, total value = 40, total weight = 4
- item 1 cannot be picked
- item 4 is picked, total value = 65, total weight = 9

Greedy algorithm 2 (value/weight) - Example 2

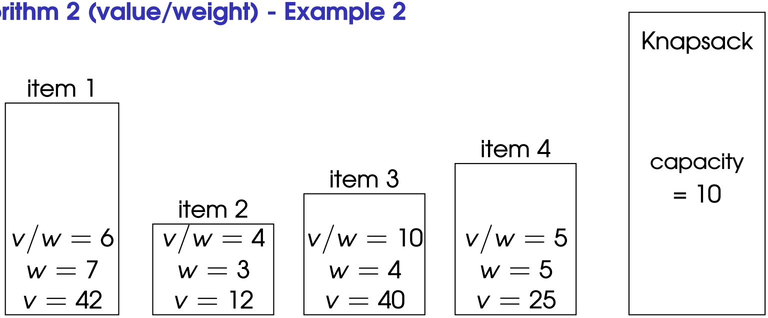


Greedy: pick the item with the next largest (value/weight) if total weight $\leq$ capacity

Result:

- item 3 is picked, total value = 40, total weight = 4
- item 1 cannot be picked
- item 4 is picked, total value = 65, total weight = 9
- item 2 cannot be picked

Greedy algorithm 2 (value/weight) - Example 2



Greedy: pick the item with the next largest (value/weight) if total weight $\leq$ capacity

Result:

- item 3 is picked, total value = 40, total weight = 4
- item 1 cannot be picked
- item 4 is picked, total value = 65, total weight = 9
- item 2 cannot be picked

Work for E.g. 1?

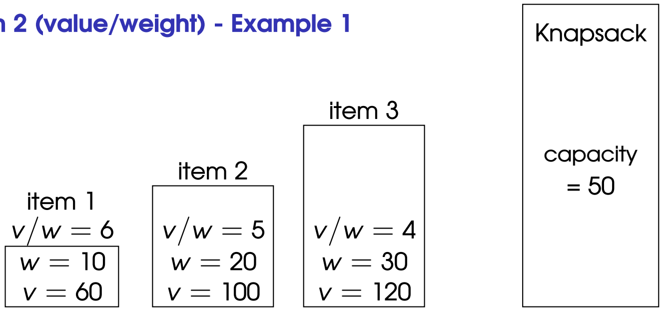
Greedy algorithm 2 (value/weight) - Example 1

1	2	3	
item 1	item 2	item 3	Knapsack
$v/w = 6$	$v/w = 5$	$v/w = 4$	capacity = 50
$w = 10$	$w = 20$	$w = 30$	
$v = 60$	$v = 100$	$v = 120$	

Greedy: pick the item with the next largest (value/weight) if total weight $\leq$ capacity

Result:

Greedy algorithm 2 (value/weight) - Example 1

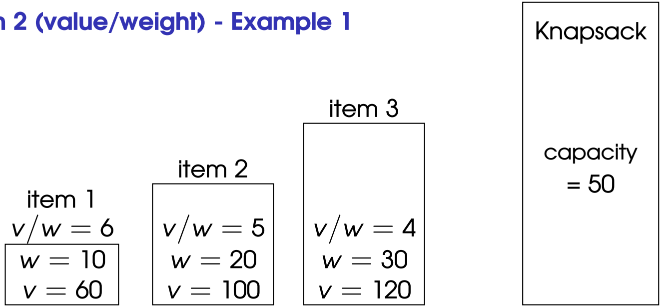


Greedy: pick the item with the next largest (value/weight) if total weight $\leq$ capacity

Result:

- item 1 is picked, total value = 60, total weight = 10

Greedy algorithm 2 (value/weight) - Example 1

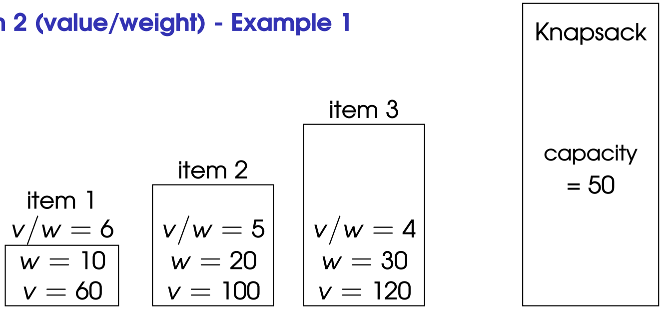


Greedy: pick the item with the next largest (value/weight) if total weight $\leq$ capacity

Result:

- item 1 is picked, total value = 60, total weight = 10
- item 2 is picked, total value = 160, total weight = 30

Greedy algorithm 2 (value/weight) - Example 1

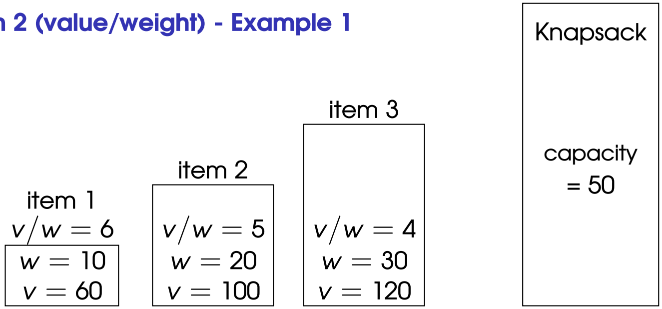


Greedy: pick the item with the next largest (value/weight) if total weight $\leq$ capacity

Result:

- item 1 is picked, total value = 60, total weight = 10
- item 2 is picked, total value = 160, total weight = 30
- item 3 cannot be picked

Greedy algorithm 2 (value/weight) - Example 1



Greedy: pick the item with the next largest (value/weight) if total weight $\leq$ capacity

Result:

- item 1 is picked, total value = 60, total weight = 10
- item 2 is picked, total value = 160, total weight = 30
- item 3 cannot be picked

Not the best!

These greedy algorithms can be arbitrarily bad

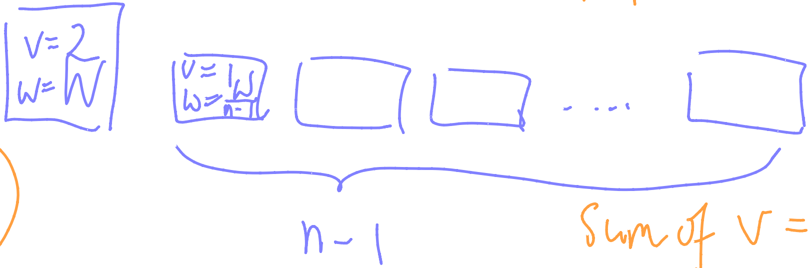
Greedy algorithm 1 - order by value

- Given n items, one item with $v = 2$ and $w = W$, and $n-1$ items with $v = 1$ and $w = \frac{W}{n-1}$.

OPT: picks all small items, total value is $n-1$

Greedy1: picks the one item with $v = 2$, total value is 2.

$$\frac{W}{n-1} \times (n-1) = W$$



2

Sum of $v = n-1$

These greedy algorithms can be arbitrarily bad

Greedy algorithm 1 - order by value

- Given n items, one item with $v = 2$ and $w = W$, and $n - 1$ items with $v = 1$ and $w = \frac{W}{n-1}$.

OPT: picks all small items, total value is $n - 1$

Greedy1: picks the one item with $v = 2$, total value is 2.

Greedy algorithm 2 - order by value/weight

- Given 2 items, one item with $v = 2$ and $w = 1$, and the other item with $v = W$ and $w = W$.

OPT: picks second item, total value is W

Greedy2: picks first item, total value is 2.

$w = 1$
$v = 2$

$v = W$
$w = W$

Greedy algorithm 3

1. Sort the items in decreasing order of $\frac{v_i}{w_i}$, like Greedy algorithm 2. I.e., $\frac{v_1}{w_1} \geq \frac{v_2}{w_2} \geq \cdots \geq \frac{v_n}{w_n}$.
2. Pick the next items as long as total weight does not exceed W.
3. Suppose items $I_1, I_2, \cdots I_{k-1}$ can be picked but I_k cannot.
4. Consider $v_1 + v_2 + \cdots + v_{k-1}$ and v_k and pick the set with larger value.

What does this mean for the example with 3 items?

$(I_1 : v_1 = 2, w_1 = 1, I_2 : v_2 = 2, w_2 = 1$, and $I_3 : v_3 = W, w_3 = W)$

total value = 4

$v_1 = 2$, $w_1 = 1$, $v/w = 2$

$v_2 = 2$, $w_2 = 1$, $v/w = 2$

$v_3 = W$, $w_3 = W$, $v/w = 1$

total value = W

Greedy algorithm 3

1. Sort the items in decreasing order of $\frac{v_i}{w_i}$, like Greedy algorithm 2. I.e., $\frac{v_1}{w_1} \geq \frac{v_2}{w_2} \geq \cdots \geq \frac{v_n}{w_n}$.
2. Pick the next items as long as total weight does not exceed W.
3. Suppose items $I_1, I_2, \cdots I_{k-1}$ can be picked but I_k cannot.
4. Consider $v_1 + v_2 + \cdots + v_{k-1}$ and v_k and pick the set with larger value.

What does this mean for the example with 3 items?

$(I_1 : v_1 = 2, w_1 = 1, I_2 : v_2 = 2, w_2 = 1$, and $I_3 : v_3 = W, w_3 = W)$

Theorem

Greedy algorithm 3 has an approximation ratio of at least $\frac{1}{2}$, i.e., it always select a subset with total value at least $\frac{1}{2}OPT$.

Greedy3: $\max\{v_1 + v_2 + \cdots + v_{k-1}, v_k\} \geq \dfrac{v_1 + v_2 + \cdots + v_{k-1} + v_k}{2}$

OPT $\leq v_1 + v_2 + \cdots + v_{k-1} + v_k$

Greedy3 $\geq \dfrac{OPT}{2}$

Lesson Learned

Greedy algorithm does **NOT** always return the best solution

Summary

Summary: Greedy algorithm for Knapsack Problem

Next: Kruskal's algorithm for Minimum Spanning Tree

For note taking