Greedy algorithms – Recap

- A greedy algorithm makes the choice that looks best at the moment, without regard for future consequence

Greedy algorithms – Recap

- A greedy algorithm makes the choice that looks best at the moment, without regard for future consequence
- The proof of the greedy algorithm producing an optimal solution is based on the following two key properties:

Greedy algorithms – Recap

- A greedy algorithm makes the choice that looks best at the moment, without regard for future consequence
- The proof of the greedy algorithm producing an optimal solution is based on the following two key properties:
  - **The greedy-choice property**
    a *globally* optimal solution can be arrived at by making a *locally* optimal (greedy) choice.

Greedy algorithms – Recap

- A greedy algorithm makes the choice that looks best at the moment, without regard for future consequence
- The proof of the greedy algorithm producing an optimal solution is based on the following two key properties:
  - **The greedy-choice property**
    a *globally* optimal solution can be arrived at by making a *locally* optimal (greedy) choice.
  - **The optimal substructure property**
    an optimal solution to the problem contains within it optimal solution to subprograms.

Greedy algorithms – Recap

- A greedy algorithm makes the choice that looks best at the moment, without regard for future consequence
- The proof of the greedy algorithm producing an optimal solution is based on the following two key properties:
  - **The greedy-choice property**
    a *globally* optimal solution can be arrived at by making a *locally* optimal (greedy) choice.
  - **The optimal substructure property**
    an optimal solution to the problem contains within it optimal solution to subprograms.
- Greedy algorithms do not always yield optimal solutions, but for many problems they do.

0-1 knapsack problem

Problem statement:

- Given n items $\{1, 2, \ldots, n\}$
- Item i is worth v_i, and weight w_i
- Find a most valuable subset of items with total weight $\leq W$

0-1 knapsack problem

Problem statement:

- Given n items $\{1, 2, \ldots, n\}$
- Item i is worth v_i, and weight w_i
- Find a most valuable subset of items with total weight $\leq W$

Rule: have to either take an item or not take it ("0-1 Knapsack") – cannot take part of it.

0-1 knapsack problem

Problem statement:

- Given n items $\{1, 2, \ldots, n\}$
- Item i is worth v_i, and weight w_i
- Find a most valuable subset of items with total weight $\leq W$

Rule: have to either take an item or not take it ("0-1 Knapsack") – cannot take part of it.

Example:

- Given

i	v_i	w_i	v_i/w_i
1	6	1	6
2	10	2	5
3	12	3	4

Total weight $W = 5$

- Find a most valuable subset of items with total weight $\leq W = 5$

0-1 knapsack problem

Problem statement, *mathematically* – version 1:

Find a subset $\mathcal{S} \subseteq \{1, 2, \ldots, n\}$ *such that*

$$\begin{aligned} \textit{maximize} \quad & \sum_{i \in \mathcal{S}} v_i \\ \textit{subject to} \quad & \sum_{i \in \mathcal{S}} w_i \leq W \end{aligned}$$

0-1 knapsack problem

Problem statement, *mathematically* – version 2:

Let $x = (x_1, x_2, \ldots, x_n)$, *and*

$$x_i = \begin{cases} 1 & i\text{-th item is in the knapsack} \\ 0 & i\text{-th item is not in the knapsack} \end{cases}$$

Then the knapsack problem is

$$\begin{aligned} \textit{maximize} \quad & \sum_{i=1}^{n} v_i x_i \\ \textit{subject to} \quad & x_i \in \{0, 1\} \\ & \sum_{i=1}^{n} w_i x_i \leq W \end{aligned}$$

0-1 knapsack problem

The brute-force algorithm

0-1 knapsack problem

The brute-force algorithm

- 2^n feasible solutions

0-1 knapsack problem

The brute-force algorithm

- 2^n feasible solutions
- Total cost $= O(n \cdot 2^n)$

0-1 knapsack problem

Three possible **greedy** strategies:

1. Greedy by highest value v_i

0-1 knapsack problem

Three possible **greedy** strategies:

1. Greedy by highest value v_i

2. Greedy by least weight w_i

0-1 knapsack problem

Three possible **greedy** strategies:

1. Greedy by highest value v_i
2. Greedy by least weight w_i
3. Greedy by largest value density $\frac{v_i}{w_i}$

0-1 knapsack problem

Example

i	v_i	w_i	v_i/w_i
1	6	1	6
2	10	2	5
3	12	3	4

Total weight $W = 5$

0-1 knapsack problem

Example

i	v_i	w_i	v_i/w_i
1	6	1	6
2	10	2	5
3	12	3	4

Total weight $W = 5$

Greedy by value density v_i/w_i:

- take items 1 and 2.
- value $=$ 16, weight $=$ 3
- Leftover capacity $=$ 2

0-1 knapsack problem

Example

i	v_i	w_i	v_i/w_i
1	6	1	6
2	10	2	5
3	12	3	4

Total weight $W = 5$

Greedy by value density v_i/w_i:

- take items 1 and 2.
- value = 16, weight = 3
- Leftover capacity = 2

Optimal solution

- take items 2 and 3.
- value = 22, weight = 5
- no leftover capacity

0-1 knapsack problem

Example

i	v_i	w_i	v_i/w_i
1	6	1	6
2	10	2	5
3	12	3	4

Total weight $W = 5$

Greedy by value density v_i/w_i:

- take items 1 and 2.
- value = 16, weight = 3
- Leftover capacity = 2

Optimal solution

- take items 2 and 3.
- value = 22, weight = 5
- no leftover capacity

Question: how about greedy by highest value? by least weight?

0-1 knapsack problem

Another example

Given the following six items with $W = 100$:

				Greedy by			optimal solution
i	v_i	w_i	v_i/w_i	value	weight	v_i/w_i	
1	40	100	0.4	1	0	0	0
2	35	50	0.7	0	0	1	**1**
3	18	45	0.4	0	1	0	**1**
4	4	20	0.2	0	1	1	0
5	10	10	1	0	1	1	0
6	2	5	0.4	0	1	1	**1**
Total value				40	34	51	**55**
Total weight				100	80	85	**100**

0-1 knapsack problem

Another example

Given the following six items with $W = 100$:

				Greedy by			optimal solution
i	v_i	w_i	v_i/w_i	value	weight	v_i/w_i	
1	40	100	0.4	1	0	0	0
2	35	50	0.7	0	0	1	**1**
3	18	45	0.4	0	1	0	**1**
4	4	20	0.2	0	1	1	0
5	10	10	1	0	1	1	0
6	2	5	0.4	0	1	1	**1**
Total value				40	34	51	**55**
Total weight				100	80	85	**100**

All three greedy approaches generate feasible solutions, but none of them generate the optimal solution. Greedy algorithms doesn't work for the 0-1 knapsack problem!