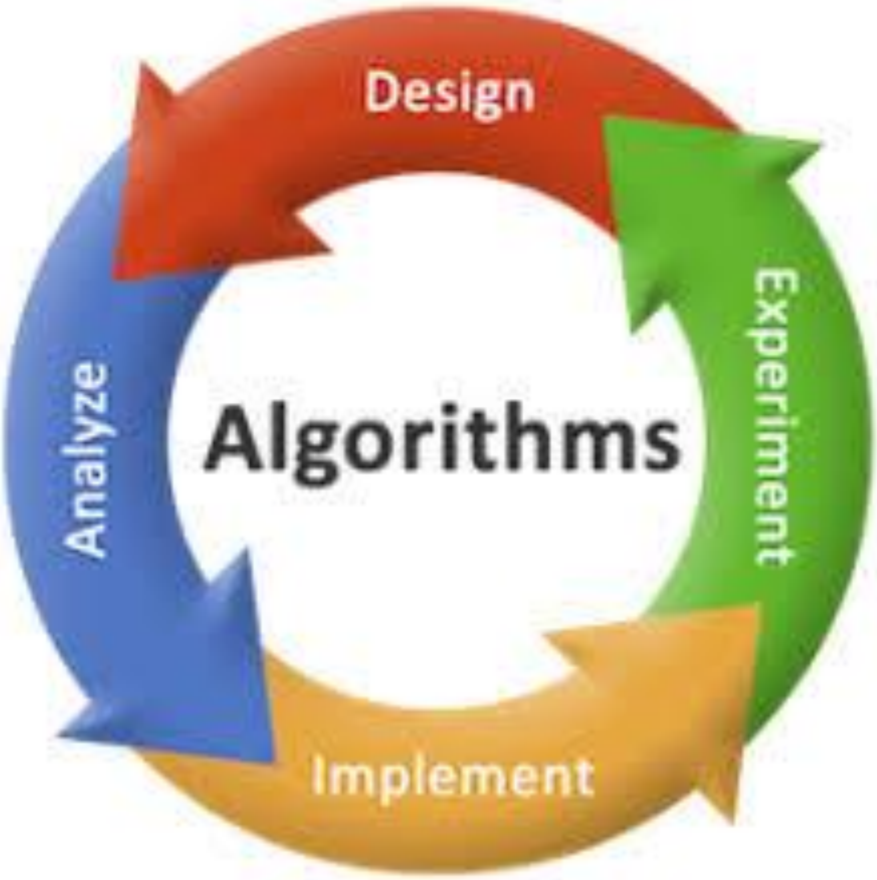
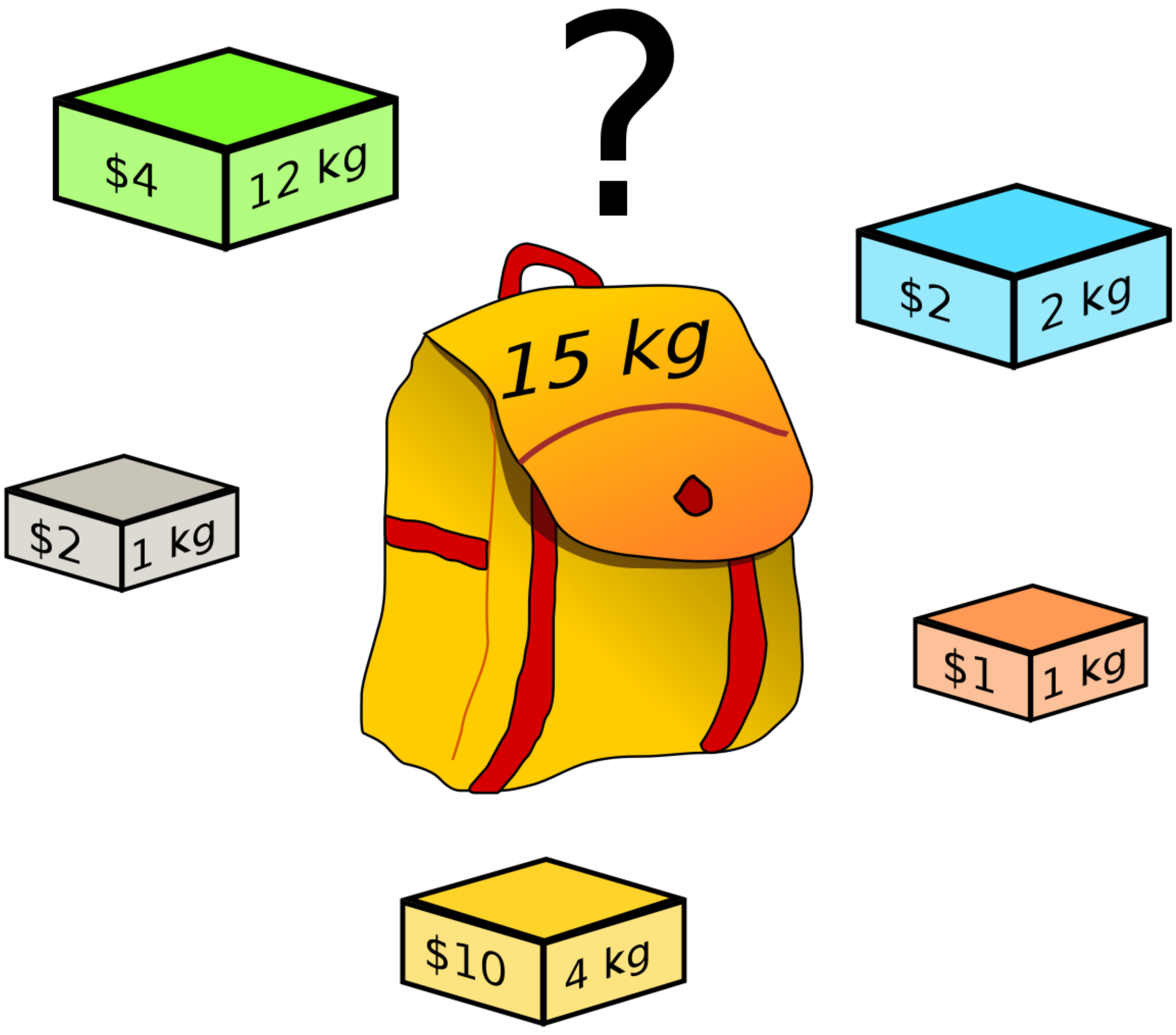


Design & Analysis of Algorithms

Lecture 12

Brute Force-
Knapsack
Problem

Knapsack Problem



Knapsack Problem

- Given n items of known weights $w_1, w_2, \ldots, w_n$ and values $v_1, v_2, \ldots, v_n$ and a **knapsack** of capacity W, find the most valuable subset of the items that fit into the knapsack.
  - A **thief** who wants to steal the **most valuable loot** that fits into his knapsack
  - A **transport plane** that has to deliver the **most valuable set of items** to a remote location without exceeding the plane's capacity.

Knapsack Problem- Types

- **0/1 Knapsack Problem**
  - Given a set of ***n*** items numbered from **1** up to ***n***, each with a weight w_i and a value v_i, along with a maximum weight capacity ***W***,

$$\text{maximize} \sum_{i=1}^{n} v_i x_i$$

$$\text{subject to} \sum_{i=1}^{n} w_i x_i \leq W \text{ and } x_i \in \{0,1\}.$$

where x_i represents the number of instances of item ***i*** to include in the knapsack.

Knapsack Problem- Types

- **Fractional Knapsack Problem**
  - Given a set of ***n*** items numbered from **1** up to ***n***, each with a weight w_i and a value v_i, along with a maximum weight capacity ***W***,

$$\text{maximize} \sum_{i=1}^{n} v_i x_i$$

$$\text{subject to} \sum_{i=1}^{n} w_i x_i \leq W \text{ and } x_i \geq 0$$

where $x_i \in \mathbf{R}$ represents the whole/fraction of instances of item ***i*** to include in the knapsack.

0/1 Knapsack Problem

- **Brute-Force/Exhaustive Solution**
  - **Generating all** the subsets of the set of n items given,
  - **Computing** the **total weight** of each subset in order to identify feasible subsets (i.e., the ones with the total weight not exceeding the knapsack capacity), and
  - **Finding** a subset of the **largest** value among them.

0/1 Knapsack Problem

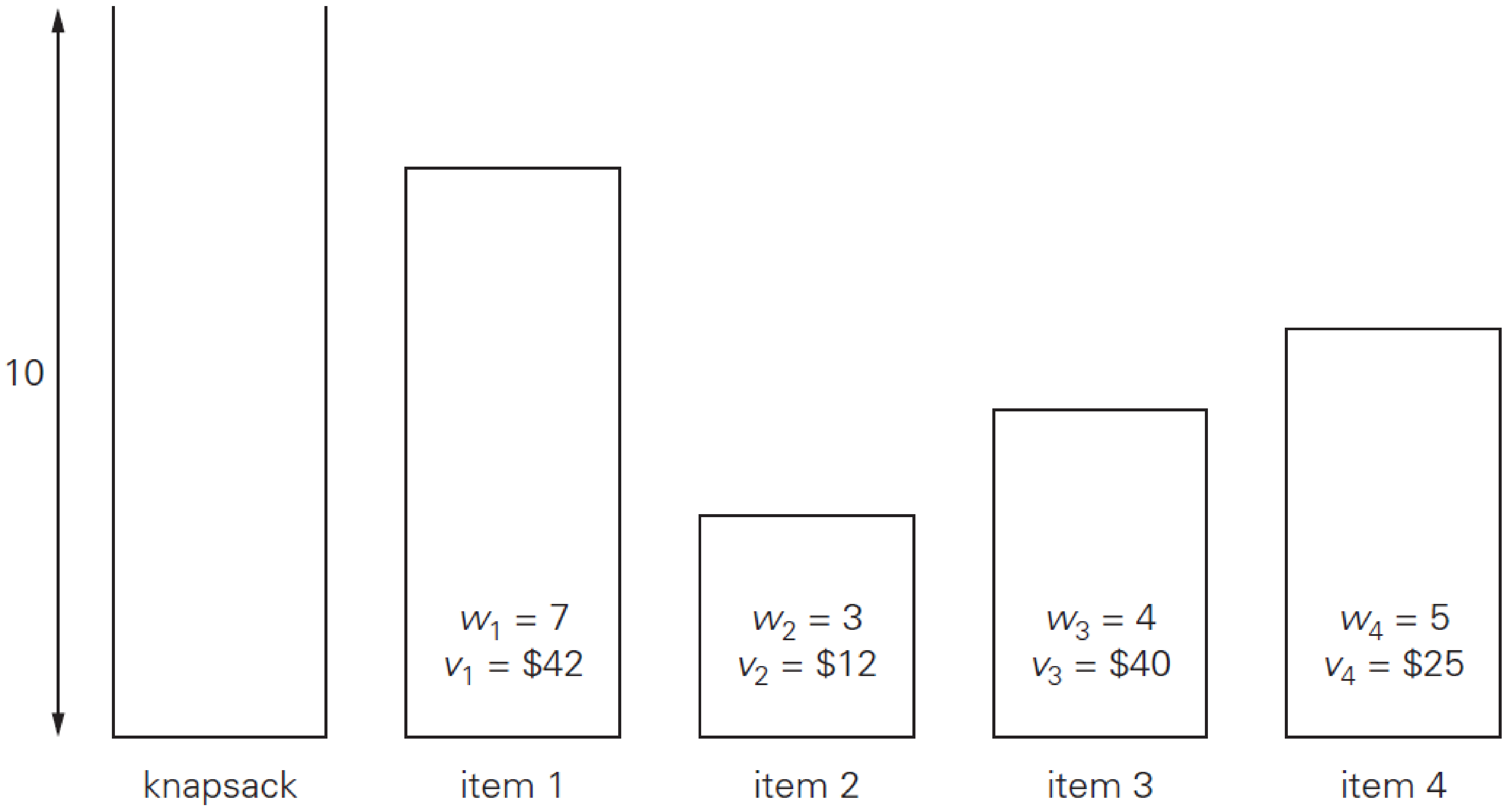
- **Brute-Force/Exhaustive Solution**
  - Since the number of subsets of an **n**-element set is $\mathbf{2^n}$, the exhaustive search leads to an algorithm of order

$$\Omega(2^n)$$

no matter how efficiently individual subsets are generated.

0/1 Knapsack Problem

Example:



0/1 Knapsack Problem

Example:

Subset	Total weight	Total value
∅	0	\$ 0
{1}	7	\$42
{2}	3	\$12
{3}	4	\$40
{4}	5	\$25
{1, 2}	10	\$54
{1, 3}	11	not feasible
{1, 4}	12	not feasible
{2, 3}	7	\$52
{2, 4}	8	\$37
{3, 4}	**9**	**\$65**
{1, 2, 3}	14	not feasible
{1, 2, 4}	15	not feasible
{1, 3, 4}	16	not feasible
{2, 3, 4}	12	not feasible
{1, 2, 3, 4}	19	not feasible

NP-Hard Problems

- Thus, for both the **traveling salesman** and **knapsack** problems considered above, exhaustive search leads to algorithms that are **extremely inefficient** on every input.
- In fact, these two problems are the best-known examples of so called ***NP-hard problems***. [**Non-deterministic Polynomial-time Hard**]
- No polynomial-time algorithm is known for any *NP*-hard problem.

References

- **Chapter 3**: Anany Levitin, “Introduction to the Design and Analysis of Algorithms”, Pearson Education, Third Edition, 2017.

Homework

- More **Cryptarithmetic** Problems
  - LETS + WAVE = LATER
  - BASE + BALL = GAMES
  - WRONG + WRONG = RIGHT
  - SEVEN – NINE = EIGHT
- Closest pair of points in a plane(2D)/space(3D).
- Water-Jug Problem

  [Out of 5,3,2 liters jug, required water of 4 liters]