Fractional Knapsack

Introduction

- Introduce the Greedy Method
- Use the greedy method to solve the fractional Knapsack problem

The Greedy Method Technique

- **The greedy method** is a general algorithm design paradigm, built on the following elements:
  - **configurations**: different choices, collections, or values to find
  - **objective function**: a score assigned to configurations, which we want to either maximize or minimize
- It works best when applied to problems with the **greedy-choice** property:
  - a globally-optimal solution can always be found by a series of local improvements from a starting configuration.

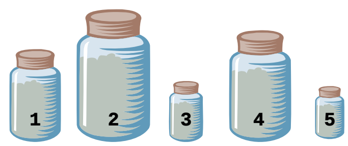
The Fractional Knapsack Problem

- **Given:** A set S of n items, with each item i having
  - b_i - a positive benefit
  - w_i - a positive weight
- **Goal:** Choose items with maximum total benefit but with weight at most W.
- If we are allowed to take fractional amounts, then this is the fractional knapsack problem.
  - In this case, we let x_i denote the amount we take of item i
  - Objective: maximize $\sum_{i \in S} b_i (x_i / w_i)$
  - Constraint: $\sum_{i \in S} x_i \le W, 0 \le x_i \le w_i$

Example

- Given: A set S of n items, with each item i having
  - b_i - a positive benefit
  - w_i - a positive weight
- Goal: Choose items with maximum total benefit but with total weight at most W.

Items:



Weight:	4 ml	8 ml	2 ml	6 ml	1 ml
Benefit:	$12	$32	$40	$30	$50
Value: ($ per ml)	3	4	20	5	50

“knapsack”

10 ml

Solution:
- 1 ml of 5
- 2 ml of 3
- 6 ml of 4
- 1 ml of 2

The Fractional Knapsack Algorithm

- Greedy choice: Keep taking item with highest value (benefit to weight ratio)
  - Since $\sum_{i\in S} b_i(x_i / w_i) = \sum_{i\in S} (b_i / w_i) x_i$

Algorithm ***fractionalKnapsack*****(*S, W*)**
Input: set *S* of items w/ benefit b_i and weight w_i; max. weight *W*
Output: amount x_i of each item *i* to maximize benefit w/ weight at most *W*

```
for each item i in S
    x_i ← 0
    v_i ← b_i / w_i        {value}
w ← 0                      {total weight}
while w < W
    remove item i with highest v_i
    x_i ← min{w_i , W - w}
    w ← w + min{w_i , W - w}
```

The Fractional Knapsack Algorithm

- Running time: Given a collection S of n items, such that each item i has a benefit b_i and weight w_i, we can construct a maximum-benefit subset of S, allowing for fractional amounts, that has a total weight W in O(nlogn) time.
  - Use heap-based priority queue to store S
  - Removing the item with the highest value takes O(logn) time
  - In the worst case, need to remove all items

The Fractional Knapsack Algorithm – contd.

- Correctness: Suppose there is a better solution
  - there is an item i with higher value than a chosen item j, but $x_i<w_i$, $x_j>0$ and $v_i<v_j$
  - If we substitute some i with j, we get a better solution
  - How much of i: $\min\{w_i-x_i, x_j\}$
  - Thus, there is no better solution than the greedy one

The End