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

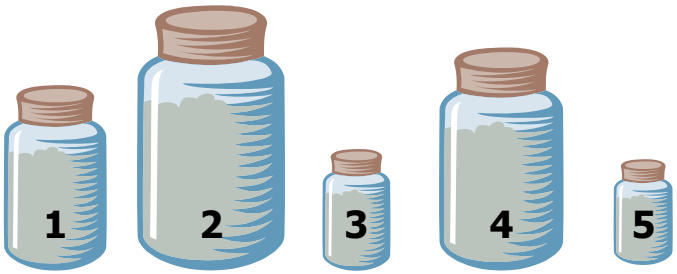
$$\sum_{i \in S} b_i (x_i / w_i)$$

  - Constraint:

$$\sum_{i \in S} x_i \leq W, 0 \leq x_i \leq w_i$$

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.



	1	2	3	4	5
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

10 ml

"knapsack"

Solution: P

- 1 ml of 5 50$
- 2 ml of 3 40$
- 6 ml of 4 30$
- 1 ml of 2 4$

•Total Profit:124$

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