Greedy Algorithms

1 The knapsack problem

We consider two variants:

1. The **0-1 knapsack problem**: A thief robs a store and finds n items. Item i is worth v_i euros and weighs w_i kilos.The thief can carry at most W kilos in knapsack.
   Question: Which items should they take in order to maximise their profit?
2. The **fractional knapsack problem**: Same set-up, but this time the thief can carry fractions of items

We assume that the knapsack is not big enough to carry all the items

$$W < w_1 + ... + w_n$$

1.1 Greedy Strategy

Fractional knapsack problem is solvable by following **greedy** strategy

1. Compute the value per kilo $r_i = v_i/w_i$ for each item
2. Sort items by value per kilo in nonincreasing order
3. Take as much as possible of item with greatest value per kilo, then take as much as possible of item with the next greatest value per kilo etc. until total weight equals knapsack capacity W

Important 1.1: nonincreasing

Nonincreasing allows for items of the same weight, unlike decreasing

1.2 Running time of greedy strategy

Step 1 (computing r_i for each i) can be done in $\Theta(n)$

Step 2 (sorting the r_i array) can be done in $\Theta(n \log n)$

Step 3 (selecting the items) can be done in $\Theta(n)$

Thus, the worst-case running time is $\Theta(n \log n)$

1.3 Greedy strategy for 0-1 knapsack

The same algorithm does NOT work for the 0-1 knapsack problem

For instance W=6 and

$$\begin{aligned} v_1 &= 3, & w_1 &= 4, & r_1 &= 3/4 \\ v_2 &= 2, & w_2 &= 3, & r_2 &= 2/3 \\ v_3 &= 2, & w_3 &= 3, & r_3 &= 2/3 \end{aligned}$$

Optimal solution: steal objects 2 and 3 for a total of 4 euros
Greedy strategy: steal object 1 for a total of 3 euros

1.4 Correctness for fractional knapsack

n items are sorted by $r_i = v_i/w_i$ in nonincreasing order. So

$$r_i \geqslant r_j \quad \text{for any} \quad i < j$$

Two items p and q with $r_p = r_q$ are considered to be equivalent. Hence, we may assume that

$$r_i > r_j \quad \text{for any} \quad i < j$$

The total available amount of item i is w_i. Let k be such that

$$w_1 + \ldots + w_k \leq W \quad \text{and} \quad w_1 + \ldots + w_{k+1} > W$$

(Recall that we assume that the knapsack is not big enough to carry all items, hence such a k exists)

Let $x = (x_1, ..., x_n)$ denote a solution: x_i is the amount the thief takes of item i for $i = 1, ..., n$
The total value stolen is then $\sum_{i=1}^{n} r_i x_i$

The greedy solution x is given by

$$x_i = w_i \text{ for } 1 \leq i \leq k$$
$$x_{k+1} = W - (w_1 + \ldots + w_k)$$
$$x_i = 0 \quad \text{for } k+2 \leq i \leq n$$

Let $x*$ be an optimal solution. First of all, we must have $x_1^* + ... + x_n^* = W$ (We might as well fill up the sack!)

We will show that $x^* = x$

Suppose $x^* \neq x$. Then $x_i^* < x_i$ for some $i \leqslant k+1$: let $j := min\{i : x_i^* < x_i\}$. Also, $x_I^* > x_I$ for some $I > j$

Let us add more of item j to the optimal choice: Let y be another solution, defined by

$$\begin{cases} y_j = \min\left\{w_j, x_j^* + x_I^* - x_I\right\} \\ y_I = x_I^* - y_j + x_j^* \\ y_i = x_i^* \quad \text{otherwise} \end{cases}$$

Exercise:

1. Check that y is a valid solution, i.e. $0 \leqslant y_i \leqslant w_i$ for all $1 \leqslant i \leqslant n$ and that $y_1 + ... + y_n = W$
2. Check that y is a better solution than x^*, i.e. $\sum_{i=1}^{n} r_i y_i > \sum_{i=1}^{n} r + i x_i^*$. Contradiction

2 Greedy Algorithms

The knapsack problem is an example of an optimization problem. In an optimization problem there can be many solutions. Each solution has a value. An optimal solution is a solution with optimal (minimum or maximum) value. We want to find an optimal solution

Algorithms for optimization problems go through sequence of steps. At each step there is a set of choices.

Important 2.1: Greedy Algorithm

A greedy algorithm makes in each step the choice that looks the best at the moment. This choice is called the greedy choice or locally optimal choice.

2.1 Performance of greedy algorithms

This strategy does not have to lead to optimal solutions. Example: 0-1 knapsack problem.

For several problems solution obtained by greedy algorithm is optimal. Examples:

- Fractional knapsack problem
- Minimum spanning tree (Prim, Kruskal)
- Shortest paths from a single source (Dijkstra)
- Data compression (Huffman coding)