Greedy Algorithms
Greedy Algorithms

**Greedy!**

- solution can be build by making a sequence of choices
- in each step take the locally best choice
- development (from Cormen)
  1. Characterise the structure of an optimal solution.
  2. Recursive solution
  3. prove that at any stage the greedy choice leads to the optimum
Examples of Greedy algorithms

- Kruskal
- Huffman-Codes
- Problems with a matroid structure
- to some extend Dijkstra’s Algorithm
- some instances of the coin exchange problem
Coin Exchange

- assume a coin system with denominations $x_1, \ldots, x_k$
- $gcd(x_1, \ldots, x_k) = 1$ (most of the time $x_1 = 1$)
- what is the minimal number of coins to pay amount $S$
- formally: minimise $n = \sum_{i=1}^{k} a_i$ under the restriction
  \[ \sum_{i=1}^{k} a_i x_i = S \]
- example CHF: pay $13\ CHF = 10\ CHF + 2\ CHF + 1\ CHF$
- example coin system 1, 3, 4: pay $6 = 4 + 1 + 1 = 3 + 3$
DP for Coin Exchange

- recursive solution:

$$minCoins(S, maxIndex) = \min \left( \min_{1<i<maxIndex} \ (minCoins(S-x_i, maxIndex)), \\
minCoins(S, maxIndex - 1) \right)$$

- DP table $T[S][k] = minCoins(S, k)$
- just fill the table
- running time $O(S \cdot k)$
Greedy for some coin systems

Nice coin systems

- coin system 1 = \( x_1, \ldots, x_k \)
- for all \( i < k \): \( x_i \leq \frac{x_k}{2} \)
- for these systems greedy is optimal
- greedy: take \( \left\lfloor \frac{S}{x_k} \right\rfloor \) coins with value \( x_k \)
- continue with \( S' = S - \left\lfloor \frac{S}{x_k} \right\rfloor \) and \( x_1, \ldots, x_{k-1} \)
- running time \( O(k) \) which is exponentially better than \( O(S \cdot k) \)
- optimality proof → blackboard