Tutorial 4: Greedy algorithms

1 Buying items from the SuperCheapStore

Suppose we would like to buy n items from the SuperCheapStore (SCS) where all items are currently priced at 1\$. Unfortunately, there is no delivery and we have to transport the items home. And to make matters worse:

- we can fit only one item in our truck; and
- it takes one day to drive home and back.

Sort in decending order, assume c1>c2>...>cn.
Let S' be an alternate solution which picks item j before item i, but j > i,
=> cj < ci
Cost includes terms, item j is picked on day d, item i is picked at day d+k
ci^(d+k) + cj^d > ci^d+cj^(d+k) ...

Worst of all, SCS charges us for the storage of undelivered items and the charge for storage of item i grows exponentially as the original price times a factor $c_i > 1$ each day. This means that if item i is picked up d days from now, the charge will be c_i^d dollars. In which order should we pick up our items from SCS so that total amount of charges is as small as possible?

Develop a greedy algorithm to solve this problem assuming that $c_i \neq c_j$ for $i \neq j$. Prove that your algorithm gives an optimal solution. What is the running time of your algorithm?

2 Total completion time

Definition 1. An instance of the *minimal total completion time* problem is a sequence of n jobs that have processing times $p_1, p_2, \ldots, p_n$ which are positive integers. A valid solution to such an instance is an ordering of the jobs $1, \ldots, n$ such that when the jobs are processed one at a time in that order, the sum of their completion times is minimized.

Design a greedy algorithm for solving the minimal total completion time problem and prove that it is correct.

Algo: Sort the pi in ascending order,
Take another solution S'
=> ordering p1,...,pn
Since this is not in ascending order
exists i such that p_i>p_i+1
swapping p_i and p_i+1 affects the cost for job i and i+1 only.
∑ k=1 to i p_k + ∑ k=1 to i+1 p_k = p_i+1+2p_i + ∑ k=1 to i-1 p_k
Cost after swap: p_i + 2p_i+1 + ∑ k=1 to i-1 p_k
p_i+1 + 2p_i-p_i-2p_i+1
p_i-p_i+1 > 0
p_i>p_i+1
exists i, a[i] > a[i+1] in ascending order, if a is sorted, the above cannot happen.
Otherwise, we can find such an index i