# Greedy Algorithm

June 1, 2019

First we need to introduce several concepts to understand what is greedy algorithm.
**Problem:** Given a problem instance, find a feasible solution that maximizes (or minimizes) a certain objective function.

**Optimal Solution:** A feasible solution $X \in feasible(I)$ sucht that the profit $f(X)$ is maximumized (or the cost $f(X)$ is minimized).

**Feasible Solution:** For any problem instance $I$, feasible(I) is the set of all outputs (i.e. solutions) for the instance $I$ that satisfy the given constraints.

**Greedy Algorithm**
Starting with the "empty" partial solution, repeatedly extend it until a feasible solution $X$ is constructed. This feasible solution may or may not be optimal.
For a greedy algorithm to be efficient, we need a fast way to find the best extension.
Note that for a greedy algorithm to be "correct", it has to find the optimal solution for every problem instance.
Greedy algorithm do NOT look ahead or backtracking.
There is only one feasible solution generated by greedy algorithm.

**Prove of correctness**
There are two ways to prove correctness.
1. Induction
2. Exchange argument: supposed there is an optimal solution –¿ compared with greedy –¿ contradiction

**Activity Selection Problem:**

Given a set of activities, each with a specified time interval, select a maximum set of disjoint intervals.

We apply greedy algorithm on selecting the activity that end earlies.

---

**Algorithm 1** Activity Selection

---

1: Sort activites 1...$n$ by end time
2: $A \leftarrow \emptyset$
3: **for** $i = 1 \cdots n$ **do**
4: **if** activity $i$ does not overlap with any previous (i.e. check the last one) activity in $A$ **then**
5: $A \leftarrow A \cup \{i\}$
6: **end if**
7: **end for**
8: **return** (A)

---

Proof for correctness:

We want to prove that $A$ returns a max size of disjoint intervals.

We prove this by using induction.

Let $A = \{a_1, \cdots, a_k\}$ sorted by end time.

Compare to the optimal solution $B = \{b_1, \cdots, b_l\}$ orderd by end time.

Thus $k \leq l$, we want to show that $k = l$.

Idea: we assume that at every step we can do better at a's.

Claim that: $\forall i, \quad a_1 \cdots, a_i, b_{i+1}, \cdots, b_l$ is an optimal solution.

Base case: consider $i = 1$ then $a_1$ has earlies end time of all intervals so,

$$end(a_1) \leq end(b_1)$$

Therefore replacing $b_1$ by $a_1$ gives a disjoint interval.

Inductive steps: suppose $a_1 \cdots, a_{i-1}, b_i, \cdots, b_l$ is an optimal solution.

There is no intersection between $a_{i-1}$ and $b_i$, so greedy algorithm could choose $b_i$. However, instead it chooses $a_i$ so

$$end(a_i) \leq end(b_i)$$

and replacing $b_i$ by $a_i$ leaves disjoint intervals.

suppose that $k < l$, then we have $a_1, \cdots, a_k, b_{k+1}, b_l$ as the optimal solution, but the greedy algorithm can keep add more activities after $a_k$, therefore $k = l$

**Coin Changing:**

A list of *coin denominations*, $d_1, d_2, \cdots, d_n$ and a positive interger $T$, which is called the target sum.

$1 = c_1 < c_2 < \cdots < c_k$ and a value $V$, where each $c_i$ is a power of 2. Design an algorithm, produces $V$ using the smallest number of coins, and prove the correctness;

Solution: We can apply greedy algorithm on this problem. Prove the correctness:

We prove this by using induction on $k$, the number of denominations of coin.

Base case: $k = 1$ we know that every algorithm gives solution sizes of $V$

Inductive Hypothesis: Greedy solution is optimal for all sets of $k-1$ coins.

Inductive Steps: consider $k > 1$, Let $g$ be the greedy algorithm $g[k] \to \mathbb{Z}_+$ and $h \neq g$ be optimal.

Case 1: consider $g(k) = h(k)$, then we define $g'$ be the same as $g$ except $g'(k) = 0$ and $h'$ defined similarily.

Then $g'$ is the greedy solution for $v - g(k)\dot{c}_k$ and $h'$ has also a better solution for $v - h(k)\dot{c}_k$, which is a controdiction

Case 2: consider $g(k) \neq h(k)$, since $h(k)$ is a better solution $\Rightarrow h(k) > g(k)$

consider we have

$$g'(k) = 0, g'(k-1) = g(k-1) + \frac{c_k}{c_{k-1}} g(k)$$

$$h'(k) = 0, h'(k-1) = h(k-1) + \frac{c_k}{c_{k-1}} h(k)$$

$g'(k)$ is greedy solution for $V$ using $c_1 \cdots c_{k-1}$ so by IH $\Sigma_{j=0}^{k-1} g'(j) \leq \Sigma_{j=0}^{k-1} h'(j)$

Since we assume that $h$ is an optimal solution, therefore we know that

$$\Sigma_{j=0}^{k} h(j) < \Sigma_{j=0}^{k} g(j) \quad h(k) < g(k)$$

$$\begin{aligned}
\Sigma_{j=0}^{k} h'(j) &= \Sigma_{j=0}^{k} h(j) - h(k) + \frac{c_k}{c_{k-1}} h(k) \\
&< \Sigma_{j=0}^{k} g(j) - h(k) + \frac{c_k}{c_{k-1}} h(k) \\
&< \Sigma_{j=0}^{k} g(j) - g(k) + \frac{c_k}{c_{k-1}} g(k) \quad \text{since } \frac{c_k}{c_{k-1}} \geq 2 \\
&< \Sigma_{j=0}^{k} g'(j)
\end{aligned}$$

That is the contradiction.

**Retrive problem:**

$n$ items with size $s_1 > \cdots > s_n$ stored at a storage service. You can only retrive exactly one item per day and the service charges a fee of $s_i^d$ to retrive item $i$ on day $d$. Give an algorithm that compute the order to retrive items with least cost and prove it correct.

The solution for this question is easy, sort the $n$ items based on its size. And always retrive the largest item in the remaining item per day.

The harder part is to prove the correctness.

However we can prove this by using exchange theorem.

Consider we have a better solution (i.e. retrive a item with smaller size instead of the largest one)

Assume that $s_1 \geq s_2 \geq \cdots \geq s_n$. Define $f : i \to d$

In the greedy algorithm, we retrive item $i$ on day $f(i)$ and item $i+1$ on day $f(i+1)$ day.

However there is an optimal solution such that it retrive $i^{th}$ item on $f(i+1)$ day and retrive item $i+1$ on $f(i)$ day. We define this relationship to be $f'$

Consider we have $f(i+1) - f(i) = d$. We want to show that:

$$\begin{aligned}
\Sigma_{j=0}^{n} s_j^{f(j)} &> \Sigma_{j=0}^{n} s_j^{f'(j)} \\
s_i^{f(i)} + s_{i+1}^{f(i+1)} &> s_i^{f(i+1)} + s_{i+1}^{f(i)} \\
s_i^{f(i)} - s_i^{f(i+1)} &> s_{i+1}^{f(i)} - s_{i+1}^{f(i+1)} \\
s_i^{f(i+1)} - s_i^{f(i)} &< s_{i+1}^{f(i+1)} - s_{i+1}^{f(i)} \\
s_i^{f(i+1)} - s_i^{f(i)} &< s_{i+1}^{f(i+1)} - s_{i+1}^{f(i)} \\
s_i^{f(i)}(s_i^d - 1) &< s_{i+1}^{f(i)}(s_{i+1}^d - 1)
\end{aligned}$$

Since we know that $s_i > s_{i+1}$, thus the above equation is contradiction.

Therefore the greedy algorithm is a correct and optimal algorithm.