Greedy Algorithms
(Programming Club 2)

Tomasz Kosciuszko

28.01.2019

1 Introduction

Greedy is a way of solving problems. Not only in programming you can either decide to be greedy and sure of short-run gains or more careful and hope for a long-run reward for your patience.

It depends on a problem if the greedy approach will be successful. Imagine a football game, it is always a good idea to score a goal, regardless of a situation. On the other hand imagine you are in a supermarket, it might not be the best to buy everything you can afford, you might not be able to carry it back home!

Similarly, in a more mathematical setup greedy approach can be very useful in some situations. The best known example is giving change. Let's say you have to pay 17p to your customer. You can greedily pay the biggest coin available and you will succeed, here it will be: 10p, then 5p, then 2p. [Can you prove that this works for any amount?].

2 How it works

Typically, you will have to come up with some simple strategy. Usually it will be something like "take the most valuable item available" or "take the lightest item available". Then you should convince yourself that your approach gives good results. Sometimes it is possible to write a formal proof.

3 Example problem

At the university there are n students and n supervisors. The strength of the i-th student is a_i and the strength of the j-th supervisor is b_j. You should pair up students with supervisors in such way to maximize the total quality of projects. If you pair a student of strength a with a supervisor of strength b they will produce a project of quality ab. What is the best result you can get? Assume $n \leq 10^5$ and $0 \leq a_i, b_j \leq 100$.

Solution: The idea is to sort both students and supervisors according to their strength and then pair up the strongest student with the strongest supervisor, 2nd with the 2nd and so on respectively.

We can prove that this is an optimal strategy by showing that if we paired strengths a with x and b with y and $a \leq b$ and $x \geq y$ then by swaping the pairs we would improve because:

$$(ay + bx) - (ax + by) = ay + bx - ax - by = (b - a)(x - y) \geq 0$$

since

$$b - a \geq 0 \; and \; x - y \geq 0$$

After swapping sufficiently many pairs we eventually sort our sequences, while still ensuring that our solution is optimal. One can write a Python function in the following way:

```
def max_score(students, supervisors):
    students.sort()
    supervisors.sort()
    return sum([students[i] * supervisors[i]
        for i in range(len(students))])
```

4 More problems

- Easy: `https://www.hackerrank.com/challenges/luck-balance/problem`
- Medium: `https://www.hackerrank.com/challenges/luck-balance/problem`
- Hard: `http://codeforces.com/problemset/problem/3/D`