Hao's Guide - Algorithm Crash Course - Greedy

Introduction

When it comes to the greedy method topics, the purpose of these algorithms is you are trying to either maximise or minimise an outcome. In other words, it's an **optimisation** problem.

This document will explain the basics and the considerations of both these topics using question examples and a step-by-step worked-out thought process.

The terms I use such as a greedy element or process are just terms I found easier to explain, they might not be what the standard is

---

Greedy 💰

What is it

A greedy method is simply where, given a problem, you are looking for 1 greedy element which you apply a maximising/minimising process over and over regardless of the circumstance. **The decision to apply the greedy method is irrespective of previous decisions**

Greedy: Money and Change

Below is a simple example of a greedy method in play

> Suppose I have a collection of $10, $5, and $1 notes, let's just assume that I have a large enough amount to cover any transaction.
> You ask me for $16 in said notes, but being the algorithmic thinker I am, I only want to give you the minimum units of currency so I have enough small change for future transactions.
> How should I decide on the number of notes I should give you

The solution is relatively trivial, as most people would have dealt with this in real life. Because in this case, I want to **minimise** the number of currency I give you, so I would want to give you the largest currency first, and for the remaining amount pick the largest currency I can give you until the transaction is fulfilled.
The currency is the element to be greedy over, and the decision on which currency is the greedy process.

Hence, the final result is me giving you: 1 x $10, 1 x $5, and 1 x $1

Greedy: Heroes and Monsters 🦹

This example was an assignment question when I did the course, it is much trickier and there are some outliers to consider. But the rules of greed apply.

> Suppose you are a superhero with an amount of energy. Your city is plagued with several monsters. Each monster will require a set amount of energy to be defeated, but will, on their defeat give you energy as well. If your energy reaches below 0, you will perish.

> Question: Does an order exist to defeat all the monsters in the city, and if an order does exist, what is the order

> Assumptions:
>
> - Defeating a monster takes energy first before giving it energy. So if you defeat a monster that results in energy lower than 0, you will still perish regardless of whether the giving energy returns anything.
> - There is no limit to the amount of energy you, the hero, can gain. Amazing.

So, as per tradition, let's apply the greedy method by finding a greedy element and the greedy process.

But wait, there is a problem already, what is the greedy element?

- If we were to pick the energy lost from defeating a monster, then surely we can take out the monsters starting from the lowest energy, but how can we ensure that the energy we gain is enough to defeat the next monster?
- If we pick the based on energy gain from defeating a monster, then surely we pick the monster with the highest energy received, but how do can we ensure we have enough energy to defeat the monster in the first place?

Worry not, because when you are not given a greedy element, what you can do is **make** a greedy element based on the given elements. So instead of dealing with two possible elements, let's instead create a *net energy* element which is the sum of the energy lost and the energy gained. A **unified** greedy element!

Now let's look at our greedy process. Given our unified element of net energy per monster, monsters will either be a net gaining monster, a net losing monster, or a neutral. We can forget neutrals for now.

Using our huge brains, we can see that we probably don't want to start losing energy by defeating net losing monsters because they won't help with defeating subsequent monsters. Instead, you want to **maximise** the amount of energy you gain. Hence, using what we know about divide and conquer, we can use merge sort to get the greatest net energy gain monster to the greatest net energy lost.

So great, now all we have to do is defeat the enemy from the greatest net gain monster to the greatest net loss monster. If at any point the hero drops below 0, we know no order exists.

But we are not done. For the keen readers, there might be some lingering edge cases that you might have noticed.

- What if the highest net gained monster removes more energy than the hero's starting energy?

This is an edge case, where you have to adjust the greedy process to accommodate. Your greedy process has not changed, it is simply just an extra consideration.

So to solve this edge case, since you are simply picking the highest net gain monster **that you can defeat**. Also, note that you need to store the order in a data structure because it is no longer always the sorted order of net energy.

Notes:

- Sometimes, you may have multiple greedy elements and processes for a single problem. Be aware