Techniques for Greedy Algorithms

Winter 2022

[[Have a running example.]]

1 Greedy Algorithm Design

In a greedy algorithm you build a complicated solution by building it one step at a time. At each step, you make the “best” remaining decision available to you by some decision criteria. The algorithm itself is usually pretty simple, along the lines of:

Let PartialSolution <- {}  
While(PartialSolution is not a full solution)  
    Consider all ways to extend PartialSolution by one choice  
    Take the best remaining choice and use it to update PartialSolution  
Return PartialSolution

The main difficulty in designing such an algorithm is finding the right notion of the “best” remaining choice. Often, there are many plausible alternatives, but most of them usually don’t work.

[[Discuss some possible options for direct greedy to the running example that don’t work.]]

When trying to find the right greedy criteria, some things to consider:

• Is there some choice that you can easily determine is a correct choice? If so, make that one.
• Try many things and look for counter-examples to see which ones don’t work.
• Is there some other way to think about what the ‘choices’ you are making are?

2 Proof of Correctness

Because it is often easy to find greedy algorithms for problems that don’t work, proving the correctness of your greedy algorithms is extra important. While many greedy algorithms can be proved correct using ad hoc methods, one generally useful technique is the exchange argument.

The idea of an exchange argument is to turn an arbitrary solution into the greedy solution without making it any worse.

In particular, you need to show two things:

1. Given any solution $A_t$ that agrees with the first $t$ choices made by your greedy algorithm (note: you have to figure out what it means for a solution to “agree” with these first $t$ choices), show that there is a solution $A_{t+1}$ that agrees with the first $t + 1$ choices and is no worse than $A_t$.

2. Any solution that agrees with all choices made by your greedy algorithm is no better than your greedy solution.

Once you have proved these, the rest of the proof is generic. Let $A$ be an arbitrary solution and $G$ be your greedy solution (which made $n$ choices). Letting $A = A_0$ we repeatedly find a sequence of solutions $A_t$ with  

$$A = A_0 \leq A_1 \leq A_2 \leq \ldots \leq A_n \leq G.$$  

This $G$ is no worse than $A$. But since $A$ can be any solution, this means that $G$ is no worse than the best solution.
3 Exam Review

You should know the following algorithms:

• BFS
• Dijkstra
• Bellman-Ford
• Shortest Paths in DAGs
• Karatsuba Multiplication
• Marge Sort
• Order Statistics
• Binary Search

In particular, for each of these you should:

• Know how and why the algorithm works.
• Be able to implement the algorithm by hand on simple inputs.
• Be able to use the algorithm as part of a bigger problem.
• Be able to modify the algorithm to adapt it to some other circumstance.