1 Introduction

Greedy is one of the more straightforward techniques that we use in computer science, but it can also be one of the more useful. It is also a dangerous method; when working on a problem, it will often appear that greedy is the answer, but in fact it will not be. However, sometimes greedy can be helpful to get partial credit on a problem where it isn’t the intended solution. In fact, you might have used a greedy solution to a problem without even knowing it.

2 The Greedy Algorithm

Here’s an example to get us started with greedy:

Milking Cows [Traditional]
You are given an integer number of liters of milk which you wish to purchase, and a list of farmers that you may purchase it from. Each farmer charges a certain amount for each liter of milk, and has a certain number of liters to sell. Find the least amount of money that you have to pay to buy the required amount of milk.

The key to this problem is that we want to buy milk at the lowest price at which it is available. In other words, we want to buy the cheapest milk until it is all gone, then move on to the next cheapest milk, and so on until we have all the milk that we need. This can easily be accomplished: we just sort the milk by price per liter, and then keep buying the cheapest milk available until we have all the milk that we need. Notice the sorting: in every greedy problem, we will sort a group of items by some value, and then work our way along the items in order. Now we will try another example that is a bit harder:

Joyful Tasks [Kolstad, 2002]
Farmer John has a list of tasks that he must do, as well as the joy that each task gives him. Each task takes him Ti time units to complete. Whenever he finishes a task, the joy of that task is added to his current joy. Farmer John wants to maximize his average joy of the day,
which is defined as \((\text{joy}[1]+\text{joy}[2]+...+\text{joy}[N]) / N\), where \(\text{joy}[t]\) is his current joy at time \(t\). Find the maximum average joy that Farmer John can achieve.

Hint: This is an example of a problem where we have to "do the math". It may be helpful to look at just two adjacent tasks and find a comparison that will tell us whether swapping these tasks will improve Farmer John’s average joy. Then try to expand this comparison method to all the tasks.

double maxaverage(int N, int joys[N]) {

}

3 Minimal Spanning Tree

A simple application of the greedy algorithm gives us an algorithm for finding the minimal spanning tree of a graph. The minimal spanning tree is the set of edges such that there is a path from every vertex to every other vertex, and the sum of the weights of the edges is a minimum. Notice that there will be no cycles in the minimal spanning tree, because then we could take out one of the edges in this cycle and it would still be a spanning tree; therefore, there will be exactly \(V-1\) edges in the minimal spanning tree. Now comes the greedy part: we sort all the edges so that they are in ascending order by weight. We move along this list of edges, and for each edge, we check if there is already a path in our current minimal spanning tree that connects the two endpoints of the edge, and if there isn’t then we add this edge to our current minimal spanning tree. To see why this works, we just note that when we add an edge, we are connecting the two endpoints of the edge, which must by definition be connected in the final graph, but aren’t connected yet (since the current edge doesn’t result in any cycles). Therefore, the current edge’s weight is the minimal weight that will connect these two points, so we can safely add it to the minimal spanning tree. When we have repeated this \(V-1\) times, we will have built a complete minimal spanning tree.