Greedy Algorithm

Greedy algorithms are a powerful tool in algorithm design. They should always be the first thing you try when you encounter a new problem. They are intuitive and simple to implement—but sometimes proving their correctness can be tricky. Let us see a few examples.

Interval Scheduling Problem

Given a set J of jobs, where each job $j \in J$ comes with a release time $r(j)$ and a deadline $d(j)$, we want to choose a maximum subset $J' \subseteq J$ so that no two jobs chosen in J' "overlap".

There are quite a few possible greedy algorithms (based on your strategy). The following is the "right" one: recursively, we take the first job with the earliest deadline that does not conflict with any job that we have already taken so far.

How do we prove this is the right algorithm? Our proof strategy is to establish the following invariant. Let us call the jobs that are chosen by our greedy $a_1, \cdots a_h$ and those chosen by optimal solution $b_1, \cdots, b_k$. We will show by induction that for all i, $d(a_i) \leq d(b_i)$. Certainly this is true when $i = 1$. For the induction step $i > 1$, we know that $d(a_{i-1}) \leq d(b_{i-1}) \leq r(b_i)$—this implies that after our algorithm has chosen $a_1, \cdots, a_{i-1}$, b_i will be a candidate (it does not conflict any of the jobs chosen by greedy so far), so by our greedy, we know that $d(a_i) \leq d(b_i)$. Now the correctness of our greedy follows from this simple invariant.

Minimum Cost Spanning Tree

Given a graph $G = (V, E)$ with edge costs $c : E \rightarrow \mathcal{R}^+$, we want a spanning tree (a tree that connects all vertices) T with the minimum cost. To make our analysis simpler, we assume that all edge costs are different (this assumption can be removed if we just add some tiny "perturbation" to the edge costs).

There are two famous algorithms to solve the problem otpimally—both are greedy: Kruskal and Prim. In the former, we sort edges by increasing costs. We proceed by adding a new edge as long as it does not create a cycle. In the latter, we choose an arbitrary vertex as the initial tree T. We then "grow" the tree T by choosing the cheapest edge connecting T with the rest of the vertices that are not part of T.

Both algorithms can be proved by the important *cut property*, which states the following.

> (**Cut Property**) Given a subset S so that $\emptyset \subsetneq S \subsetneq V$, the cheapest edge e in $\delta(S)$ (which means the set of edges with one endpoint in S and the other in $V \backslash S$) must be part of the minimum spanning tree.

Let us prove this by contradiction. Suppose that $e = (u, v)$ is the cheapest edge in $\delta(S)$ but it is not part of the optimal tree T. Suppose further that $u \in S$ and $v \in V \backslash S$. Remember that T is a *spanning tree*. So there must be a path $P \subseteq T$ between u and v.

Notice that there must exist (w,x) along this path so that $w \in S$ and $x \in V \backslash S$ (why?) and we know that $c(w,x) > c(u,v)$.

Now as $(u,v) \cup P$ forms cycle C, we can modify the tree T so that it becomes $T \backslash (w,x) \cup (u,v)$. This new tree is still a spanning tree and the cost strictly goes down, giving a contradiction to the assumption that T is the optimal tree.

It should be a simple matter to use this cut property to prove the correctness of algorithms of Kruskal and Prim.

Minimum Weighted Sum of Completion Time

We are given a set of jobs $j \in J$, each with a size s_j and a weight w_j. Given a linear order $\prec$ over the jobs in J, the completion time of the job j is defined as $C_j = \sum_{j' \preceq j} s_{j'}$. Our goal is to find a good order so that $\sum_{j \in J} w_j C_j$ is minimized—this is one of the famous scheduling problems.

This problem again can be solved by a greedy algorithm: this greedy is usually called *Smith Rule*. We order the jobs by their increasing *density*: s_j/w_j.

So suppose that Smith Rule does *not* give the optimal solution and we will try to give a contradiction. To simplify notation, we assume that in the optimal solution, the order of the jobs is simply $1, 2, \cdots n$. Since optimal solution does not use Smith Rule, it means that there exist two consecutive jobs t and $t+1$ so that $\frac{s_t}{w_t} > \frac{s_{t+1}}{w_{t+1}}$.

Let us "flip" the order of the two jobs t and $t+1$. Notice that after this flip, all other jobs still have the same completion time. So only these two jobs have their completion times changed. Now what difference of the objective will be

$$w_t(\sum_{j=1}^{t} s_j) + w_{t+1}(\sum_{j=1}^{t+1} s_j) - w_t(\sum_{j=1}^{t+1} s_j) - w_{t+1}(\sum_{j=1}^{t-1} s_j + s_{t+1}) = w_{t+1}s_t - w_t s_{t+1} > 0,$$

a contradiction.