CSE 202 NOTES FOR NOVEMBER 7, 2002

Methods for Proving Greedy Algorithms Correct

If you haven't figured it out yet, greedy algorithms are usually incorrect but efficient. Therefore, you need to be really careful to show that your greedy strategy doesn't somehow screw up the result. Here are three main arguments that you can use to show that your greedy rule is sound.

Modify-The-Solution: Covered in the last two lectures, this approach is often (but not always) very easy, and is usually the way to approach the problem.

Achieves-The-Bounds: This is a somewhat more insightful proof, though it is often a bit harder than the Modify-The-Solution approach. In this argument, you define a set of "obstacles" and a function bound$(o), o \in$ Obstacles. To show that the greedy algorithm works, prove two lemmas:

Lemma 0.1. *For all solutions S, obstacle o, Cost$(S) \geq$ bound(o), or if it's more appropriate, Value$(S) \leq$ bound(o).*

Lemma 0.2. *Let S_g be the greedy solution. Then there exists an $o \in$ Obstacles such that Cost$(S_g) =$ bound(o), or if it's more appropriate, Value$(S_g) =$ bound(o).*

If both of these lemmas are true, then the following theorem applies.

Theorem 0.3. *S_g is optimal.*

Proof: Let o be as in lemma 2, and let S' be any solution. Cost$(S_g) =$ bound$(o) \leq$ CostS' by lemma 1.

This technique is also useful for judging hill-climbing algorithms. It isn't always a possible approach for proving greedy algorithms correct, but as mentioned before, it's somewhat more insightful into the problem.

Unique local optimum: In this technique, you define and characterize "small" changes to a solution S, $\Delta_i(S)$. Then show that the following lemma is true.

Lemma 0.4. *If $S \neq S_g$, $\exists i$ such that $\Delta_i(S) \succ S$.*

Here I use $\prec$ and $\succ$ to mean "is worse than" and "is better than" respectively.

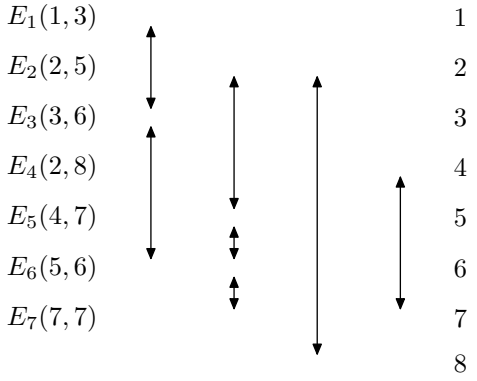
If this lemma holds, then you can prove the following theorem by contradiction.

Theorem 0.5. *S_g is the only optimal solution.*

Proof: Let $S_{\text{opt}} \neq S_g$. Then, there exists $\Delta_i(S_{\text{opt}})$ such that $\Delta_i(S_{\text{opt}}) \succ S_{\text{opt}}$, which is a contradiction. Therefore S_g is the only optimal solution.

This particular proof technique makes it very easy to prove S_g's optimality by using a fallacy: if you were to show that $\forall i, \Delta_i(S_g) \prec S_g$, then you are simply showing local optimality. The reason that there is a difference

FIGURE 1. A sample schedule for a set of events in conference rooms. Double arrows represent events. Two events that are aligned vertically occur in the same room.



between $\Delta_i(S_g) \prec S_g$ and $\Delta_i(S) \succ S$ is that there is only one S_g but many S's: by showing that every S (except S_g) has a Δ associated with it, you're actually making a strong statement about S_g). By showing that S_g has no small changes Δ that will improve it you're really only making a weak statement about S_g, unless you can show that the set of all Δ's is the set of all solutions.

Multiple Conference Room Scheduling

To illustrate the Achieves-The-Bound argument form, we will examine a new problem.

Instance: List of events $e = (s, f)$.
Solution Format: Integer k and mapping $r(e_i) \in \{1, 2, \ldots, k\}$ that assigns e_i to a room k.
Constraints: No two events overlap in time. All events are assigned to a room.
Objective: Minimize k, the number of rooms needed.

As an example, consider the following set: $\{(1,3),(2,5),(3,6),(2,8),(4,7),(5,6),(6,7)\}$. The situation is depicted graphically in figure 1. Notice that at 4:00, we have events E_3, E_2, E_4, E_5 all at the same time, so we need a minimum of 4 rooms.

Since we are focusing on the proof of a greedy algorithm and not the design of a greedy algorithm, I'll just blurt out the greedy algorithm: sort the events according to their start time. Walk down the list, in sorted order, of course, and place each event in the smallest numbered conference room that doesn't already have something going on at that time.

What were the obstacles?
Times t.

What are the bounds?
The number of events happening at time t: you can never do better than the number of things going on at one particular point in time.

True to Achieve-The-Bound form, we will prove the following lemmas.

Lemma 0.6. *Let t be any time, and let R : events $\mapsto$ Rooms be any valid schedule. Furthermore, let $B(t) = |\{e \in E : e_s \leq t \wedge e_e \geq t\}|$. Then, $R \geq B(t)$ for all t.*

Proof: Let $E_{i_1}, E_{i_2}, \ldots E_{i_{B(t)}}$ be the events running at time t. Each E_i scheduled in a separate room by Dirichlet's pigeonhole principle.

Lemma 0.7. *Let R_g be the k picked by the greedy algorithm. Then $\exists t$ such that $B(t) \geq k$. That is, at least k events are happening at t.*

Proof: Let t be the starting time of the event in room k. There was an event $e_{i_1} = (s_{i_1}, f_{i_1})$ in room 1, $e_{i_2} = (s_{i_2}, f_{i_2})$ in room 2, and so on to $e_{i_{k-1}}$, since $s_{i^*} < t < f_{i^*}$ for each $j = 1, 2, \ldots k-2$ and $s_i < s_j$ when $j > i$ since we sorted the events. Therefore, all of these events are going on at time t, and our lemma is proved. (In other words, if k was picked by the greedy solution, then there must have been k events going on because the greedy solution would have chosen some conference room $k' < k$, but it apparently didn't.)

Theorem 0.8. *The greedy algorithm uses the minimum number of rooms.*

Proof: Let k be the number of rooms that the greedy algorithm uses, and let R be any valid schedule. There exists a t when k events are happening simultaneously, so by lemma 1, R uses at least k rooms. So, R uses at lesat as many rooms as R_g, and this holds for $R = R_{\text{opt}}$. QED.

Dijkstra's Algorithm for Single-Source Shortest Path

To drive home the Achieves-The-Bound technique, let's look at one more example, Dijkstra's Single-Source Shortest Path algorithm. This is sometimes known as a "Breadth-first Search" of a graph G.

Instance: A directed weighted graph with all $w_{ij} > 0$. A source node $s \in V$.
Solution Format: A set of paths from s to all nodes $u \in V$.
Constraints: Paths are only listed once. (Ok, there aren't really any constraints that aren't blatantly obvious.)
Objective: Minimize the cost from s to each node $u \in V$.

Note that this is different than finding the minimum spanning tree because you're specifying the source. Is it the same as minimizing the total distance of the paths from s to each node u? I believe it is. Since the total distance is the sum of all the distances from s to each node u, by not having the total distance minimized, it means that there is some u' such that $d(s, u')$ is not as small as it could be. But that's a contradiction because these are supposed to be shortest paths!

What are the decisions?
For each node u, decide what predecessor node from u to take.

What is the greedy property?
Pick the decision point where the distance from s to u is smallest (and u has not yet been seen).

What is the greedy rule?
Choose the edge, and set the shortest path from s to u.

The algorithm is shown in more detail in algorithm 1.

Let $D(u)$ be the shortest distance from s to u. We want to prove the following invariant: at any step j, $\forall u \in S, d(u) = D(u)$. We'll prove this by induction on j. As a base case, $d(s) = 0 = D(s)$. As our inductive assumption, assume that this is true for every $u \in S$ before iteration j; we need to show that the distance is

SSSP($G = (V, E, w)$, s): directed weighted graph, all weights positive:

$S \leftarrow \{s\}$
$d(s) \leftarrow 0$
$p(s) \leftarrow s$ {Or some sentinel value}
$p(v) \leftarrow$ NIL, $d(v) \leftarrow \infty$ for all $v \in V, v \neq s$.
for $n - 1$ times **do**
  Let Σ be $\{e = (u, v) \in E : u \in S \wedge v \notin S\}$
  Let $e_g = (u_g, v_g) \in \Sigma$ be the edge with the smallest cost: $w[u, v] + d(u)$.
  $S \leftarrow S \cup \{v\}$
  $d(v_g) \leftarrow w[u_g, v_g] + d(p(u_g))$
  $p(v_g) \leftarrow u_g$
end for

Algorithm 1: Dijkstra's Single-Source Shortest Path Algorithm

properly configured after iteration j, that $d(v_g) = D(v_g)$. In order to do this, we will use the achieves-the-bound argument.

We will state and prove two lemmas about this algorithm.

Lemma 0.9. *Let P be any path between s to v_g. Then $D(v_g) \geq d(v_g)$. ($e_g = (u_g, v_g)$ is the edge that would be picked in iteration j by the greedy algorithm.)*

Proof: $P = sx_1x_2 \ldots x_{j-1}v_g$, where all the $s \cdots x_{j-1}$ are in S and v_g is not in S. Let $e' = (x_i, x_{i+1}) \in \Sigma$ be the edge in Σ with the shortest distance. Then $\text{Cost}(P) \geq \text{Cost}(s, \ldots, x_i) + w(e') \geq D(x_i) + w(e') = d(x_i) + w(e')$. Furthermore, by the algorithm above, $w(e') \geq w(e_g)$. So, e' is some edge in Σ, and e_g is the smallest edge in Σ, so $\text{Cost}(e') \geq \text{Cost}(e_g)$. Therefore, $D(v_g) \geq d(v_g)$.□

Lemma 0.10. *There is a path P such that Cost(P) = Cost(e_g).*

Proof: $\exists P' = s \cdots u_g$ of cost $d(u_g)$. Let $P = P' \cup e_g$.