Solutions -- Topic 13, Greedy Algorithms, Class 03/09

Activity Scheduling

In the activity scheduling problem, we are given the start and finish times s_i and f_i of a set S of candidate activities, and possibly other data about them. We know all the activities in advance: this is batch scheduling, not real time. We need to **select (schedule) a subset A of the activities that are *compatible* (do not overlap with each other)**. There are different versions of the problem depending on what we maximize.

Maximizing Count

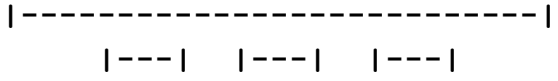
In this version of the problem we ***find the largest possible set (maximum <u>count</u>,*** not duration) *of activities that do not overlap with each other*. CLRS show that this problem has the ***greedy choice property***: a globally optimal solution can be assembled from locally optimal choices. They show it by example with this *greedy strategy*: Always select the remaining compatible activity that **ends first**, and then solve the subproblem of scheduling the activities that start after this activity.

There are other greedy strategies, but some work and some don't. In the following two problems (and the extra credit problem) you determine whether alternative locally greedy strategies lead to globally optimal solutions. For each strategy below,

- <u>either show that the strategy leads to an optimal solution</u> by outlining the algorithm or approach (be sure to show it works on *any* input, not just the example you drew),
- <u>or give a counterexample</u>: write down a set of activities (defined by their start and finish times) that the strategy fails on, and show what goes wrong.

1. *Greedy strategy*: Always select the remaining compatible activity with the **earliest start time**. *Proposed Rationale:* Don't waste any time getting started.

Solution (1 pt): Counterexample:



2. *Greedy strategy*: Always select the remaining compatible activity that has the **latest start time**. *Proposed Rationale:* Leave the most time remaining at the beginning for other activities.

Solution (2 pts): This is the same as the greedy approach "ends first", but

chronologically reversed. We can most easily solve it by changing the problem representation and using the existing algorithm. Define $a'_i = [f_i, s_i)$ and give the algorithm a' instead of a.

Maximizing Value

We now consider a variation of the problem. Suppose that different activities earn different amounts of revenue. In addition to their start and finish times s_i and f_i, each activity a_i has revenue r_i, and our objective is now to **maximize the total revenue:**

$$\sum_{a_i \in A} r_i$$

3. Can you identify a greedy strategy that shows that this problem exhibits the greedy choice property? If so, show how it works to maximize value. If not, show counterexamples for the strategies you considered and identify an alternative problem solving strategy that would work on this problem.

Solution (2 pts): a greedy algorithm won't work. Counterexamples for the above strategies all apply (they all ignore the value). Counter examples also exist for value-optimizing methods:

Choose highest value first:

|--100--|
|-------99-------| |-------99-------|

Choose highest density of value per unit time first:

Above is also counterexample for this

⇒ Use dynamic programming instead.

Extra credit: More Strategies for Maximizing Count

As you did in #1 and #2,

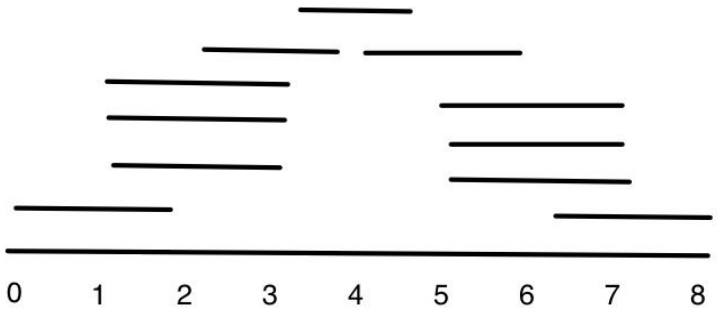
- either show that strategy below leads to an optimal solution
- or give a counterexample.

4. *Greedy strategy*: Always select the remaining compatible activity that has the **least duration**. *Proposed Rationale:* Leave the most time remaining for other activities.

Counterexample:

```
                |-------|
|---------------|     |---------------|
```

5. *Greedy strategy*: Always select the remaining compatible activity that **overlaps with the fewest number of remaining activities**. *Proposed Rationale:* Eliminate the fewest number of remaining activities from consideration.



Middle activity will be chosen first, forcing choice of only one activity on each side. However, it is possible to get a sequence of 4 activities.