Data Structures and Algorithms: The formal notes

Josh Felmeden

October 7, 2019

Contents

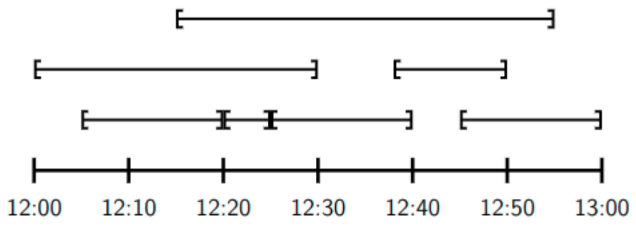
1 Greedy algorithms

1.1 Interval scheduling

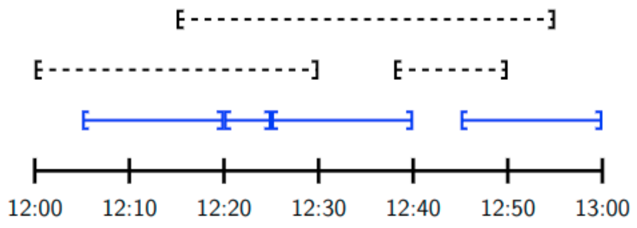
Let's suppose that you're running a satellite imaging service. Taking a satellite picture of an area isn't instant and can take some time. It can also only be done on the day where the satellite's orbit is lined up correctly. Say you want to request some images to be taken from said satellite, each of which can only be taken at certain times and you can only take one picture at a time. How do we satisfy as many requests as possible?

The requested satellite times that we have to deal with are: 12:00-12:30, 12:05-12:20, 12:15-12:55, 12:20-12:25, 12:38-12:50, and 12:45-13:00.

If we visualise this in a graph, we get:



If we take a greedy algorithm approach to assigning these slots, we could do something like assign the slot that finishes earliest, and then repeat doing this until we have reached the end. For example, the slot that finishes fastest is 12:05-12:20, so we assign this. This now removes the ability for both 12:10-12:30 and 12:15-12:55 to be assigned, so we remove these. This continues until we end up with something looking like this:



This means that we satisfy four requests (which is actually the maximum possible, so well done us).

We can formalise this by saying that a **request** is a pair of integers (s, f) with $0 \leq s \leq f$.

The algorithm that we're left with is this:

```
public sub greedySchedule
    sort R
    for each i in {1 ... n} do
        if s_i >= lastf then
            A.append(s_i, f_i)
            lastf = f_i
        end if
    next
end sub
```

Now, we need to prove that the output is actually a **compatible subset** of R. This is sort of intuitive because the set we added doesn't break compatibility, since $s \geq$ `lastf` and `lastf` is the latest finish time that's already in A.

We can formalise this with a loop invariant. At the start of the ith iteration, we see that

- A contains a compatible subset $\{(S_1, F_1), \ldots, (S_t, F_t)\}$ of R.
- `lastf` = $\max(\{0\} \cup \{F_j : j \leq t\})$

The base case ($i = 0$) is immediate because $A = []$. The induction step is that A was compatible at the start of the iteration, and therefore if we append a pair s_i, f_i to A then $s \geq$ `lastf` $\geq F_j$ for all $f \leq t$. This means that (s_i, f_i) is compatible with A.