【Detailed analysis】Let me lead you to the solution step by step

kaiwensun ★ 1594 Last Edit: February 16, 2020 1:24 AM 4.8K VIEWS

143

Where to start your thinking

Always keep in mind: interviewers rarely expect you to invent new algorithms. They almost always test your skills to apply algorithms you've learned at school.
So, what algorithms have you learned at schools that are usually used to solve questions involving an array? divide and conquer, greedy.... Hmm... this question reminds me of the question about scheduling meetings meeting rooms, which is solved by greedy algorithm. Even if you don't know the scheduling meeting questi attempt with DP and divide-and-conquer, and will find it is not very straight forward to define the subprobl the split point of divide-and-conquer. Hmm... so greedy algorithm looks like the right one. Let's try that.

Greedy algorithm intuition

Greedy algorithms are usually very intuitive (but not necessarily correct. it requires proof). What would you multiple equally important meetings to run, but can only make some of them? Most people probably woulc the one that is going to end soon. And after that meeting, pick the next meeting from those that are still av

Greedy algorithm proof

At some day, suppose both events `E1` and `E2` are available to choose to attend. For contradictory purpose `E1` that is going to end sooner is not the best choice for now. Instead, `E2` that ends later is the best choice choosing `E2` now, you come up with a schedule `S1`.

I claim that I can always construct another schedule `S2` in which we choose `E1` instead of `E2` for now, and than `S1`.
In `S1`, from now on, if `E1` is picked some time after, then I can always swap `E1` and `E2` in `S1`, so I constr not worse than `S1`.
In `S1`, from now on, if `E1` is **not** picked some time after, then I can aways replace `E2` in `S1` with `E1`, so I which is not worse than `S1`.

So it is always better (at least not worse) to always choose the event that ends sooner.

Greedy algorithm implementation

As we go through each days to figure out the availability of each events, it is very intuitive to first sort the `e` starting day of the events. Then the question is, how to find out which (still available) event ends the earlies need to sort the **currently available** events according to the ending day of the events. How to do that? Aga don't expect you to invent something realy new! What data structures / algorithm have you learned that ca track of the biggest value, while you can dynamically add and remove elements? ..... Yes! Binary search/inse