Greedy 1: Activity Selection (Picking a Maximum Number of Disjoint Intervals)

Yufei Tao

Department of Computer Science and Engineering Chinese University of Hong Kong

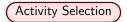


Activity Selection

1/10

- - E

In this lecture, we will commence our discussion of the **greedy** technique. In fact, this technique enforces a very simple strategy: simply make the **locally optimal** decision at each step. It is important to note that this technique does **not** always give a **globally optimal** solution. There are, however, problems where it does. The nontrivial part of applying the technique is to prove (or disprove) the global optimality.



Problem definition

Input: A set *S* of *n* intervals of the form [s, f] where *s* and *f* are integer values.

Output: A subset T of disjoint intervals in S with the largest size |T|.

Remark: You can think of [s, f] as the duration of an activity, and consider the problem as picking the largest number of activities that do not have time conflicts.

Example: Suppose

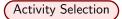
 $S = \{[1, 9], [3, 7], [6, 20], [12, 19], [15, 17], [18, 22], [21, 24]\}.$

An optimal solution is $T = \{[3,7], [15,17], [18,22]\}$. Optimal solutions may not be unique; here is another one: $T = \{[1,9], [12,19], [21,24]\}$.

-

4/10

・ 同 ト ・ ヨ ト ・ ヨ ト



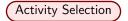
Complication: Once an interval is taken, those overlapping with it will have to be discarded. So one mistake may lead to a suboptimal solution.

It turns out that the following greedy strategy works: simply take the interval with the earliest finish time (i.e., smallest f-value) at each step.

Algorithm

Repeat the following steps until *S* becomes empty:

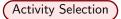
- Add to T the interval $\mathcal{I} \in S$ with the smallest finish time.
- Remove from S all the intervals intersecting \mathcal{I} (including \mathcal{I} itself)



Example: Suppose $S = \{[1,9], [3,7], [6,20], [12,19], [15,17], [18,22], [21,24]\}.$

Sort the intervals in S by finish time: $S = \{[3,7], [1,9], [15,17], [12,19], [6,20], [18,22], [21,24]\}.$

We first add [3,7] to T, after which intervals [3,7], [1,9] and [6,20] are removed. Now S becomes $S = \{[15,17], [12,19], [18,22], [21,24]\}$. The next interval added to T is [15,17], which shrinks S further to $S = \{[18,22], [21,24]\}$. After [18,22] is added to T, S becomes empty and the algorithm terminates.



Now comes the nontrivial part: prove the algorithm is **correct**, namely, it indeed returns an optimal solution. We will do so by mathematical induction.

Base Step: n = 1.

That is, S has only one interval, in which case the output of the algorithm is obviously optimal.

Inductive Step: Assuming that the algorithm is correct for all $n \le k$. We will prove that it is also correct for n = k + 1.

7/10

・ロト ・ 同ト ・ ヨト ・ ヨト

Claim: Let $\mathcal{I} = [s, f]$ be the interval in S with the smallest finish time. There must be an optimal solution that contains \mathcal{I} .

Proof: Let T^* be an arbitrary optimal solution that does not contain \mathcal{I} . We will turn T^* into another optimal solution T that contains \mathcal{I} , and thereby finish the proof.

Let $\mathcal{I}' = [s', f']$ be the interval in T^* with the **smallest** finish time. We construct T as follows: add all the intervals in T^* to T except \mathcal{I}' , and finally add \mathcal{I} to T.

We will prove that all the intervals in T are disjoint. This indicates that T is also an optimal solution, and hence, will complete the proof.

・ロット 全部 マート・ キョッ

-

It suffices to prove that \mathcal{I} cannot intersect with any other interval $\mathcal{J} \in \mathcal{T}$.

Suppose on the contrary that there is such a $\mathcal{J} = [a, b]$. By definition of \mathcal{I}' , we must have $f' \leq b$. Combining this and the fact that \mathcal{J} is disjoint with \mathcal{I}' , we assert that f' < a. On the other hand, by definition of \mathcal{I} , it must hold that $f \leq f'$. It thus follows that f < a. But this indicates that \mathcal{I} and \mathcal{J} are disjoint, giving a contradiction.

Think 1: Now that we know \mathcal{I} must be in an optimal solution, how do we proceed with the induction proof that the algorithm is correct for n = k + 1? This will be left as a regular exercise (solution provided in full).

Think 2: How to implement the algorithm in $O(n \log n)$ time? This will be left as another regular exercise (again, solution provided in full).

10/10

マロト イラト イラト