Greedy 1: Activity Selection

Yufei Tao

Department of Computer Science and Engineering
Chinese University of Hong Kong
In this lecture, we will commence our discussion of **greedy** algorithms, which enforce a simple strategy: make the **locally optimal** decision at each step. Although this strategy does not always guarantee finding a **globally optimal** solution, sometimes it does. The nontrivial part is to prove (or disprove) the global optimality.
Activity Selection

Input: A set $S$ of $n$ intervals of the form $[s, f]$ where $s$ and $f$ are integers.
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

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

\[ T = \{[3, 7], [15, 17], [18, 22]\} \text{ is an optimal solution, and so is } T = \{[1, 9], [12, 19], [21, 24]\}. \]
Algorithm
Repeat until $S$ becomes empty:
- Add to $T$ the interval $I \in S$ with the smallest finish time.
- Remove from $S$ all the intervals intersecting $I$ (including $I$ itself)
Example: Suppose $S = \{[1, 9], [3, 7], [6, 20], [12, 19], [15, 17], [18, 22], [21, 24]\}$.

For convenience, let us rearrange the intervals in $S$ in ascending order of 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 $\{[15, 17], [12, 19], [18, 22], [21, 24]\}$. The next interval added to $T$ is $[15, 17]$, which shrinks $S$ further to $\{[18, 22], [21, 24]\}$. After $[18, 22]$ is added to $T$, $S$ becomes empty and the algorithm terminates.
Next, we will prove that the algorithm returns an optimal solution. Let us start with a crucial claim.

**Claim 1:** Let $I_1$ be the first interval picked by our algorithm. There must be an optimal solution containing $I_1$.

**Proof:** Let $T^*$ be an arbitrary optimal solution. If $I_1 \in T^*$, Claim 1 is true and we are done. Next, we assume $I_1 \notin T^*$.

We will turn $T^*$ into another optimal solution $T$ containing $I$. For this purpose, first identify the interval $I'_1$ in $T^*$ with the *smallest* finish time. Construct $T$ as follows: add all the intervals in $T^*$ to $T$ except $I'$, and finally add $I$ 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.
It suffices to prove that $I_1$ cannot intersect with any other interval in $J \in T$. This is true because

- the start time of $J$ is after the finish time of $I_1'$;
- the finish time of $I_k$ is less than or equal to the finish time of $I_1'$.
Claim 2: Let $I_1, I_2, \ldots, I_k$ be the first $k \geq 2$ intervals picked by our algorithm (in the order shown). Assume that there is an optimal solution containing $I_1, \ldots, I_{k-1}$. Then, there must exist an optimal solution containing $I_1, \ldots, I_{k-1}, I_k$.

Proof: Let $T^*$ be an optimal solution containing $I_1, \ldots, I_{k-1}$. Observe:

All the intervals in $T^* \setminus \{I_1, \ldots, I_{k-1}\}$ must start strictly after the finish time of $I_{k-1}$.

Think: Why is the observation true?
If $I_k \in T^*$, Claim 2 is true and we are done. Next, we consider the case where $I_k \notin T^*$.

Let $I'_k$ be the interval in $T^* \setminus \{I_1, \ldots, I_{k-1}\}$ that has the smallest finish time. Construct a set $T$ of intervals as follows: add all the intervals of $T^*$ to $T$ except $I'_k$, and finally add $I_k$ to $T$.

To prove that $T$ is an optimal solution, it suffices to prove that $I_k$ is disjoint with every interval $J \in T^* \setminus \{I_1, \ldots, I_{k-1}, I'_k\}$. This is true because

- the start time of $J$ is after the finish time of $I'_k$;
- the finish time of $I_k$ is less than or equal to the finish time of $I'_k$.

☐
Think: How to implement the algorithm in $O(n \log n)$ time?