CSCI3160 Design and Analysis of Algorithms (2025 Fall)

Greedy 1: Activity Selection

Instructor: Xiao Liang[1]

Department of Computer Science and Engineering
Chinese University of Hong Kong

[1]These slides are primarily based on materials prepared by Prof. Yufei Tao (please refer to Prof. Tao's version from 2024 Fall for the original content). Some modifications have been made to better align with this year's teaching progress, incorporating student feedback, in-class interactions, and my own teaching style and research perspective.

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 — Real-World Example

Story: You're planning a one-day mini-conference. You have many candidate talks, each with a start time and an end time. You only have one room, so talks can't overlap.

Goal: Fit in as many talks as possible without time conflicts.

What does "overlap" mean? Two talks overlap if their time intervals share any time. Example:

- [9:00, 10:00] and [9:30, 10:30] overlap;
- [9:00, 10:00] and [10:00, 11:00] do *not* (if we allow back-to-back).

Your choices:

- You can include a talk or skip it.
- If you include it, you must avoid any other talk that overlaps with it.

Activity Selection: Formal Setup

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.

Activity Selection

Example: Suppose

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

$T = \{[3,7],[15,17],[18,22]\}$ is an optimal solution, and so is $T = \{[1,9],[12,19],[21,24]\}$.

A Greedy Algorithm for Activity Selection

Activity Selection

One-line summary: Pick the earliest-finishing interval, keep it, and discard all intervals that overlap it. Repeat until none remain.

Algorithm (formal)
Repeat 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)

Time Complexity: Convince yourself that this algorithm can be implemented in $O(n \log n)$ time.

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

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.

Why the Greedy Choice?

Greedy Choice Property:

- Picking the earliest finishing activity leaves the most room for future choices.

Key Insight:

- If we always choose the earliest finishing compatible activity, we never miss out on a better solution.

Proof Outline: Exchange Argument

Note: there may be more than one optimal solution (think: can you construct an example for this fact?). Therefore, it is not accurate to say that our algorithm outputs "the" optimal solution. It is sufficient to show that our algorithm outputs an optimal solution.

Let:

- $G = \{g_1, g_2, ..., g_k\}$ be the greedy solution
- $O = \{o_1, o_2, ..., o_m\}$ be an optimal solution, ordered by increasing finish time.

Remarks:

- Each g_i (or o_i) is an interval!
- Since there is no overlap among the o_i's, it holds for each i that $f_{o_{i-1}}$ is earlier than s_{o_i}, which is the start time of o_i.

Goal: Show that $k = m$ (greedy is optimal)

Some Simple Facts

Let:

- $G = \{g_1, g_2, ..., g_k\}$ be the greedy solution
- $O = \{o_1, o_2, ..., o_m\}$ be an optimal solution, ordered by increasing finish time.

Facts:

1. Each g_i (or o_i) is an interval!
2. Since there is no overlap among the o_i's, it holds for each i that $f_{o_{i-1}}$ is earlier than s_{o_i}, which is the start time of o_i.
3. It must be that $k \leq m$; Otherwise, it violates the optimality of the optimal solution O.
4. So, to prove $k = m$ (our goal), it suffices to show that k is no less than m.

Exchange Argument in Action

Strategy: Transform O into G without reducing the number of activities using an **exchange argument**.

Assume:

- Greedy picks g_1 (earliest finishing)
- Optimal picks o_1 (maybe different)
- $f_{g_1} \leq f_{o_1}$ (where f_{g_1} denotes the finish time of activity g_1. Similar for f_{o_1}.)

Exchange Step:

- Replace o_1 with g_1 in O
- Since g_1 finishes no later than o_1, it's compatible with all following activities in O

Repeat the above steps to replace $o_2, o_3, \ldots$.

Finishing the Proof

If we repeat the above exchange argument, we can gradually update O in the following order:

$$\begin{aligned} O &= \{o_1, o_2, o_3, \dots, o_m\} \\ O_1 &= \{g_1, o_2, o_3, \dots, o_m\} \\ O_2 &= \{g_1, g_2, o_3, \dots, o_m\} \\ &\vdots \end{aligned}$$

Assuming for the sake of contradiction that $k < m$, the following sequence would stop at

$$O_k = \{g_1, g_2, \dots, g_k, o_{k+1}, o_{k+2} \dots, o_m\}.$$

Finishing the Proof

Such an O_k contradicts the description of the greedy algorithm:

- Note that the finish time of g_k is earlier than the finish time of all intervals in

$$\{o_{k+1}, o_{k+2} \ldots, o_m\},$$

  and g_k do not over lap with any one of them.
- Given the above, by definition, the greedy algorithm would **not** stop at g_k, because there are still intervals satisfying the algorithm's condition to be added to the solution set G.
- **Contradiction:** However, by our assumption, g_k is the last element in G. This gives a contradiction.

Thus, our assumption that $k < m$ is incorrect. It must be that $k = m$.

This finish the proof