V. Greedy Algorithms
Greedy algorithms – Overview

- Algorithms for solving (optimization) problems typically go through a sequence of steps, with a set of choices at each step.
Greedy algorithms – Overview

- Algorithms for solving (optimization) problems typically go through a sequence of steps, with a set of choices at each step.

- A greedy algorithm always makes the choice that looks best at the moment, without regard for future consequence, i.e., “take what you can get now” strategy.
Greedy algorithms – Overview

- Algorithms for solving (optimization) problems typically go through a sequence of steps, with a set of choices at each step.

- A greedy algorithm always makes the choice that looks best at the moment, without regard for future consequence, i.e., “take what you can get now” strategy

- Greedy algorithms do not always yield optimal solutions,

  \[
  \text{Local optimum } \iff \text{ Global optimum}
  \]

  but for many problems they do.
Activity-selection problem

Problem statement:

Input: Set $S = \{1, 2, \ldots, n\}$ of $n$ activities

$s_i =$ start time of activity $i$

$f_i =$ finish time of activity $i$

Output: Maximum-size subset $A \subseteq S$ of compatible activities

Remarks:

▶ Activities $i$ and $j$ are compatible if the intervals $(s_i, f_i)$ and $(s_j, f_j)$ do not overlap.

▶ Without loss of generality, assume $f_1 \leq f_2 \leq \cdots \leq f_n$.
Activity-selection problem

Problem statement:

*Input:* Set $S = \{1, 2, \ldots, n\}$ of $n$ activities
- $s_i =$ start time of activity $i$
- $f_i =$ finish time of activity $i$

Remarks:
- Activities $i$ and $j$ are compatible if the intervals $[s_i, f_i)$ and $[s_j, f_j)$ do not overlap.
- Without loss of generality, assume $f_1 \leq f_2 \leq \cdots \leq f_n$. 


Activity-selection problem

Problem statement:

**Input:** Set $S = \{1, 2, \ldots, n\}$ of $n$ activities

- $s_i = \text{start time of activity } i$
- $f_i = \text{finish time of activity } i$

**Output:** Maximum-size subset $A \subseteq S$ of compatible activities
Activity-selection problem

Problem statement:

\textbf{Input:} Set $S = \{1, 2, \ldots, n\}$ of $n$ activities

\begin{align*}
    s_i &= \text{start time of activity } i \\
    f_i &= \text{finish time of activity } i
\end{align*}

\textbf{Output:} Maximum-size subset $A \subseteq S$ of \textit{compatible} activities

Remarks:

- Activities $i$ and $j$ are \textit{compatible} if the intervals $[s_i, f_i)$ and $[s_j, f_j)$ do not overlap.
Activity-selection problem

Problem statement:

**Input:** Set $S = \{1, 2, \ldots, n\}$ of $n$ activities
$s_i = \text{start time of activity } i$
$f_i = \text{finish time of activity } i$

**Output:** Maximum-size subset $A \subseteq S$ of compatible activities

Remarks:
- Activities $i$ and $j$ are compatible if the intervals $[s_i, f_i)$ and $[s_j, f_j)$ do not overlap.
- Without loss of generality, assume

$$f_1 \leq f_2 \leq \cdots \leq f_n$$
Activity-selection problem

Example

<table>
<thead>
<tr>
<th></th>
<th>$s_i$</th>
<th>$f_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>7</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>8</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>9</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>10</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>11</td>
</tr>
<tr>
<td>9</td>
<td>8</td>
<td>12</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>13</td>
</tr>
<tr>
<td>11</td>
<td>12</td>
<td>14</td>
</tr>
</tbody>
</table>
Activity-selection problem

Example

<table>
<thead>
<tr>
<th>i</th>
<th>$s_i$</th>
<th>$f_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>7</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>8</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>9</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>10</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>11</td>
</tr>
<tr>
<td>9</td>
<td>8</td>
<td>12</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>13</td>
</tr>
<tr>
<td>11</td>
<td>12</td>
<td>14</td>
</tr>
</tbody>
</table>

$A = \{1, 4, 8, 11\}$ is an optimal (why?) solution.
Activity-selection problem

Example

<table>
<thead>
<tr>
<th>$i$</th>
<th>$s_i$</th>
<th>$f_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>7</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>8</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>9</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>10</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>11</td>
</tr>
<tr>
<td>9</td>
<td>8</td>
<td>12</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>13</td>
</tr>
<tr>
<td>11</td>
<td>12</td>
<td>14</td>
</tr>
</tbody>
</table>

$A = \{1, 4, 8, 11\}$ is an optimal (why?) solution.

$A = \{2, 4, 9, 11\}$ is also an optimal solution.
Activity-selection problem

Greedy algorithm:

- pick the compatible activity with the earliest finish time.
Activity-selection problem

Greedy algorithm:

- *pick the compatible activity with the earliest finish time.*

Why?

- Intuitively, this choice leaves as much opportunity as possible for the remaining activities to be scheduled
Activity-selection problem

Greedy algorithm:

- pick the compatible activity with the earliest finish time.

Why?

- Intuitively, this choice leaves as much opportunity as possible for the remaining activities to be scheduled.

- That is, the greedy choice is the one that maximizes the amount of unscheduled time remaining.
**Activity-selection problem**

Greedy_Activity_SELECTOR(s, f)
$n = \text{length}(s)$
$A = \{1\}$
$j = 1$

for $i = 2$ to $n$
    if $s[i] \geq f[j]$
        $A = A \cup \{i\}$
        $j = i$
    end if
end for
return $A$

Remarks
▶ Assume the array $f$ already sorted
▶ Complexity: $T(n) = O(n)$
Activity-selection problem

Greedy_Activity_Selector(s,f)
n = length(s)
A = {1}
j = 1
for i = 2 to n
    if s[i] >= f[j]
        A = A U {i}
        j = i
    end if
end for
return A

Remarks

▶ Assume the array f already sorted
▶ Complexity: $T(n) = O(n)$
Activity-selection problem

Example

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>$s_i$</td>
<td>$f_i$</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>7</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>8</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>9</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>10</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>11</td>
</tr>
<tr>
<td>9</td>
<td>8</td>
<td>12</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>13</td>
</tr>
<tr>
<td>11</td>
<td>12</td>
<td>14</td>
</tr>
</tbody>
</table>

Solution

$A = \{1, 4, 8, 11\}$ by Greedy Activity Selector.
Activity-selection problem

Example

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$i$</td>
<td>$s_i$</td>
<td>$f_i$</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>7</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>8</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>9</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>10</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>11</td>
</tr>
<tr>
<td>9</td>
<td>8</td>
<td>12</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>13</td>
</tr>
<tr>
<td>11</td>
<td>12</td>
<td>14</td>
</tr>
</tbody>
</table>

Solution $A = \{1, 4, 8, 11\}$ by Greedy_Activity_Selector.
Activity-selection problem

**Question:** Does $\text{Greedy\_Activity\_Selector}$ work?
Activity-selection problem

Question: Does `Greedy_Activity_SELECTOR` work?
Answer: Yes!
Activity-selection problem

Question: Does \texttt{Greedy\_Activity\_Selector} work?
Answer: Yes!

\textbf{Theorem.} Algorithm \texttt{Greedy\_Activity\_Selector} produces a solution of the activity-selection problem.
Activity-selection problem

The proof of **Theorem** is based on the following two properties:

Property 1. There exists an optimal solution $A$ such that the greedy choice "1" in $A$.

Proof:

1. Let's order the activities in $A$ by finish time such that the first activity in $A$ is "$k_1$".
2. If $k_1 = 1$, then $A$ begins with a greedy choice.
3. If $k_1 \neq 1$, then let $A' = (A - \{k_1\}) \cup \{1\}$.

Then:

1. The sets $A - \{k_1\}$ and $\{1\}$ are disjoint.
2. The activities in $A'$ are compatible.
3. $A'$ is also optimal, since $|A'| = |A|$.

Therefore, we conclude that there always exists an optimal solution that begins with a greedy choice.
Activity-selection problem

The proof of Theorem is based on the following two properties:

Property 1.

There exists an optimal solution $A$ such that the greedy choice “1” in $A$. 

Proof: 

- Let's order the activities in $A$ by finish time such that the first activity in $A$ is $k_1$.

  - If $k_1 = 1$, then $A$ begins with a greedy choice.

  - If $k_1 \neq 1$, then let $A' = (A - \{k_1\}) \cup \{1\}$.

  1. The sets $A - \{k_1\}$ and $\{1\}$ are disjoint.

  2. The activities in $A'$ are compatible.

  3. $A'$ is also optimal, since $|A'| = |A|$. 

Therefore, we conclude that there always exists an optimal solution that begins with a greedy choice.
Activity-selection problem

The proof of Theorem is based on the following two properties:

Property 1.
There exists an optimal solution $A$ such that the greedy choice “1” in $A$.

Proof:
- let’s order the activities in $A$ by finish time such that the first activity in $A$ is “$k_1$”.

$\therefore$ Therefore, we conclude that there always exists an optimal solution that begins with a greedy choice.
Activity-selection problem

The proof of **Theorem** is based on the following two properties:

**Property 1.**

There exists an optimal solution $A$ such that the greedy choice “1” in $A$.

**Proof:**

- Let's order the activities in $A$ by finish time such that the first activity in $A$ is “$k_1$”.
- If $k_1 = 1$, then $A$ begins with a greedy choice
Activity-selection problem

The proof of Theorem is based on the following two properties:

Property 1.

There exists an optimal solution $A$ such that the greedy choice "1" in $A$.

Proof:

- let's order the activities in $A$ by finish time such that the first activity in $A$ is \( k_1 \).
- If $k_1 = 1$, then $A$ begins with a greedy choice.
- If $k_1 \neq 1$, then let $A' = (A - \{k_1\}) \cup \{1\}$.
Activity-selection problem

The proof of **Theorem** is based on the following two properties:

**Property 1.**

*There exists an optimal solution A such that the greedy choice “1” in A.*

**Proof:**

- **let's order the activities in A by finish time such that the first activity in A is “k₁”**.
- **If k₁ = 1, then A begins with a greedy choice**
- **If k₁ ≠ 1, then let A' = (A − {k₁}) ∪ {1}**. Then
  1. the sets A − {k₁} and {1} are disjoint
Activity-selection problem

The proof of Theorem is based on the following two properties:

Property 1.

There exists an optimal solution $A$ such that the greedy choice “1” in $A$.

Proof:

- Let’s order the activities in $A$ by finish time such that the first activity in $A$ is “$k_1$”.
- If $k_1 = 1$, then $A$ begins with a greedy choice.
- If $k_1 \neq 1$, then let $A' = (A - \{k_1\}) \cup \{1\}$.

Then

1. the sets $A - \{k_1\}$ and $\{1\}$ are disjoint
2. the activities in $A'$ are compatible
Activity-selection problem

The proof of Theorem is based on the following two properties:

Property 1.

There exists an optimal solution $A$ such that the greedy choice “1” in $A$.

Proof:

- let's order the activities in $A$ by finish time such that the first activity in $A$ is “$k_1$”.
- If $k_1 = 1$, then $A$ begins with a greedy choice
- If $k_1 \neq 1$, then let $A' = (A - \{k_1\}) \cup \{1\}$.

Then
1. the sets $A - \{k_1\}$ and $\{1\}$ are disjoint
2. the activities in $A'$ are compatible
3. $A'$ is also optimal, since $|A'| = |A|$
Activity-selection problem

The proof of **Theorem** is based on the following two properties:

**Property 1.**

*There exists an optimal solution \( A \) such that the greedy choice “1” in \( A \).*

**Proof:**

- Let's order the activities in \( A \) by finish time such that the first activity in \( A \) is “1”.
- If \( k_1 = 1 \), then \( A \) begins with a greedy choice.
- If \( k_1 \neq 1 \), then let \( A' = (A - \{ k_1 \}) \cup \{ 1 \} \).
  
  Then
  
  1. the sets \( A - \{ k_1 \} \) and \( \{ 1 \} \) are disjoint
  2. the activities in \( A' \) are compatible
  3. \( A' \) is also optimal, since \( |A'| = |A| \)

Therefore, we conclude that there always exists an optimal solution that begins with a greedy choice.
Activity-selection problem

Property 2.

If $A$ is an optimal solution, then $A' = A - \{1\}$ is an optimal solution to $S' = \{i \in S, s[i] \geq f[1]\}$.
Activity-selection problem

Property 2.

If \( A \) is an optimal solution, then \( A' = A - \{1\} \) is an optimal solution to \( S' = \{i \in S, s[i] \geq f[1]\} \).

Proof: By contradiction. If there exists \( B' \) to \( S' \) such that \( |B'| > |A'| \), then let 

\[ B = B' \cup \{1\}, \]

we have 

\[ |B| > |A|, \]

which is contradicting to the optimality of \( A \).
Activity-selection problem

Proof of **Theorem**: By Properties 1 and 2, we know that

- After each greedy choice is made, we are left with an optimization problem of the same form as the original.
Activity-selection problem

Proof of **Theorem**: By Properties 1 and 2, we know that

- After each greedy choice is made, we are left with an optimization problem of the same form as the original.
- *By induction* on the number of choices made, making the greedy choice at every step produces an optimal solution.

Therefore, the **Greedy_Activity_Selector** produces an optimal solution of the activity-selection problem.
Activity-selection problem

- Property 1 is called the **greedy-choice property**, generally casted as

  a globally optimal solution can be arrived at by making a locally optimal (greedy) choice.
Activity-selection problem

- Property 1 is called **the greedy-choice property**, generally casted as
  
  *a globally optimal solution can be arrived at by making a locally optimal (greedy) choice.*

- Property 2 is called **the optimal substructure property**, generally casted as

  *an optimal solution to the problem contains within it optimal solution to subprograms.*

These are **two key properties** for the success of greedy algorithms!