Greedy
CSCI 432

Greedy Algorithms:

- Make the choice that best helps some objective.
- Do not look ahead, plan, or revisit past decisions.
- Hope that optimal local choices lead to optimal global solutions.

Greedy Algorithms:

- Make the choice that best helps some objective.
- Do not look ahead, plan, or revisit past decisions.
- Hope that optimal local choices lead to optimal global solutions.

Greedy algorithm for:
- Robbing a jewelry store?
- Eating at a fancy buffet?
- Triaging buddy after failed cornice huck?

Activity Selection

Input:
- $S = \{a_1, a_2, \ldots, a_n\}$ – set of courses that need rooms.
- $a_i = (s_i, f_i)$ – start and finish times for each course.

Activity Selection

Input:
- $S = \{a_1, a_2, \ldots, a_n\}$ – set of courses that need rooms.
- $a_i = (s_i, f_i)$ – start and finish times for each course.

Rules:
- a_i and a_j are compatible if $[s_i, f_i)$ and $[s_j, f_j)$ do not overlap.

Activity Selection

Input:
- $S = \{a_1, a_2, \ldots, a_n\}$ – set of courses that need rooms.
- $a_i = (s_i, f_i)$ – start and finish times for each course.

Rules:
- a_i and a_j are compatible if $[s_i, f_i)$ and $[s_j, f_j)$ do not overlap.

Goal: Select a maximum sized subset of mutually compatible courses.

Activity Selection

Input:

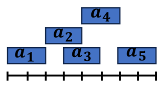
- $S = \{a_1, a_2, \dots, a_n\}$ – set of courses that need rooms.
- $a_i = (s_i, f_i)$ – start and finish times for each course.

Rules:

- a_i and a_j are compatible if $[s_i, f_i)$ and $[s_j, f_j)$ do not overlap.

Goal: Select a maximum sized subset of mutually compatible courses.

i	1	2	3	4	5
s_i	1	3	4	5	7
f_i	3	5	6	7	9



Activity Selection

Input:

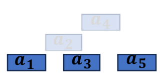
- $S = \{a_1, a_2, \dots, a_n\}$ – set of courses that need rooms.
- $a_i = (s_i, f_i)$ – start and finish times for each course.

Rules:

- a_i and a_j are compatible if $[s_i, f_i)$ and $[s_j, f_j)$ do not overlap.

Goal: Select a maximum sized subset of mutually compatible courses.

i	1	2	3	4	5
s_i	1	3	4	5	7
f_i	3	5	6	7	9



Activity Selection

Input:

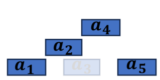
- $S = \{a_1, a_2, \dots, a_n\}$ – set of courses that need rooms.
- $a_i = (s_i, f_i)$ – start and finish times for each course.

Rules:

- a_i and a_j are compatible if $[s_i, f_i)$ and $[s_j, f_j)$ do not overlap.

Goal: Select a maximum sized subset of mutually compatible courses.

i	1	2	3	4	5
s_i	1	3	4	5	7
f_i	3	5	6	7	9



Activity Selection

Input:
- $S = \{a_1, a_2, \ldots, a_n\}$ – set of courses that need rooms.
- $a_i = (s_i, f_i)$ – start and finish times for each course.

Rules:
- a_i and a_j are compatible if $[s_i, f_i)$ and $[s_j, f_j)$ do not overlap.

Goal: Select a maximum sized subset of mutually compatible courses.

i	1	2	3	4	5
s_i	1	3	4	5	7
f_i	3	5	6	7	9

Greedy selection criteria?

Algorithm Outline?

Activity Selection

Input:
- $S = \{a_1, a_2, \ldots, a_n\}$ – set of courses that need rooms.
- $a_i = (s_i, f_i)$ – start and finish times for each course.

Rules:
- a_i and a_j are compatible if $[s_i, f_i)$ and $[s_j, f_j)$ do not overlap.

Goal: Select a maximum sized subset of mutually compatible courses.

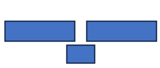
Greedy decision?
Smallest duration first.

Activity Selection

Input:
- $S = \{a_1, a_2, \ldots, a_n\}$ – set of courses that need rooms.
- $a_i = (s_i, f_i)$ – start and finish times for each course.

Rules:
- a_i and a_j are compatible if $[s_i, f_i)$ and $[s_j, f_j)$ do not overlap.

Goal: Select a maximum sized subset of mutually compatible courses.



Greedy decision?
Smallest duration first.

Activity Selection

Input:
- $S = \{a_1, a_2, \ldots, a_n\}$ – set of courses that need rooms.
- $a_i = (s_i, f_i)$ – start and finish times for each course.

Rules:
- a_i and a_j are compatible if $[s_i, f_i)$ and $[s_j, f_j)$ do not overlap.

Goal: Select a maximum sized subset of mutually compatible courses.

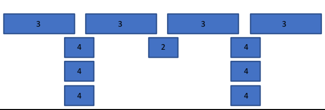
Greedy decision?
Smallest conflict first.

Activity Selection

Input:
- $S = \{a_1, a_2, \ldots, a_n\}$ – set of courses that need rooms.
- $a_i = (s_i, f_i)$ – start and finish times for each course.

Rules:
- a_i and a_j are compatible if $[s_i, f_i)$ and $[s_j, f_j)$ do not overlap.

Goal: Select a maximum sized subset of mutually compatible courses.



Greedy decision?
Smallest conflict first.

Activity Selection

Input:
- $S = \{a_1, a_2, \ldots, a_n\}$ – set of courses that need rooms.
- $a_i = (s_i, f_i)$ – start and finish times for each course.

Rules:
- a_i and a_j are compatible if $[s_i, f_i)$ and $[s_j, f_j)$ do not overlap.

Goal: Select a maximum sized subset of mutually compatible courses.

Greedy decision?
Earliest compatible finish time first.

Activity Selection

```
activity_selection(activities A)
  sort_by_finish(A)
  selected = {A[1]}
  last_added = 1
  for i = 2 to A.length
    if A[i].start ≥ A[last_added].finish
      selected  = selected  ∪ {A[i]}
      last_added = i
  return selected
```

i	1	2	3	4	5
s_i	1	3	4	5	7
f_i	3	5	6	7	9

Activity Selection

Running Time?

Validity?

Performance?

```
activity_selection(activities A)
  sort_by_finish(A)
  selected = {A[1]}
  last_added = 1
  for i = 2 to A.length
    if A[i].start ≥ A[last_added].finish
      selected  = selected  ∪ {A[i]}
      last_added = i
  return selected
```

i	1	2	3	4	5
s_i	1	3	4	5	7
f_i	3	5	6	7	9

Activity Selection

Running Time? $O(n \log n)$

Validity?

Performance?

```
activity_selection(activities A)
  sort_by_finish(A)
  selected = {A[1]}
  last_added = 1
  for i = 2 to A.length
    if A[i].start ≥ A[last_added].finish
      selected  = selected  ∪ {A[i]}
      last_added = i
  return selected
```

i	1	2	3	4	5
s_i	1	3	4	5	7
f_i	3	5	6	7	9

Activity Selection

Running Time? $O(n \log n)$

```
activity_selection(activities A)
  sort_by_finish(A)
  selected = {A[1]}
  last_added = 1
  for i = 2 to A.length
    if A[i].start ≥ A[last_added].finish
      selected = selected ∪ {A[i]}
      last_added = i
  return selected
```

Validity? `selected` consists of compatible courses.

Performance?

i	1	2	3	4	5
s_i	1	3	4	5	7
f_i	3	5	6	7	9

Activity Selection

Running Time? $O(n \log n)$

```
activity_selection(activities A)
  sort_by_finish(A)
  selected = {A[1]}
  last_added = 1
  for i = 2 to A.length
    if A[i].start ≥ A[last_added].finish
      selected = selected ∪ {A[i]}
      last_added = i
  return selected
```

Validity? `selected` consists of compatible courses.

Performance? Is `selected` the largest possible subset?

i	1	2	3	4	5
s_i	1	3	4	5	7
f_i	3	5	6	7	9

Activity Selection

<u>Greedy decision:</u> Select the next course with the earliest compatible finish time.

<u>Proof of optimality:</u>

Activity Selection

Greedy decision: Select the next course with the earliest compatible finish time.

Proof of optimality: Let A be the set of courses, $S \subseteq A$ be the greedy algorithm selection, and $OPT \subseteq A$ be an optimal selection, all sorted by increasing finish time.

Activity Selection

Greedy decision: Select the next course with the earliest compatible finish time.

Proof of optimality: Let A be the set of courses, $S \subseteq A$ be the greedy algorithm selection, and $OPT \subseteq A$ be an optimal selection, all sorted by increasing finish time.

$S[0] = A[0]$, since that is the greedy choice. Suppose $OPT[0] \neq A[0]$ (i.e. the optimal solution does not start with the greedy choice).

Activity Selection

Greedy decision: Select the next course with the earliest compatible finish time.

Proof of optimality: Let A be the set of courses, $S \subseteq A$ be the greedy algorithm selection, and $OPT \subseteq A$ be an optimal selection, all sorted by increasing finish time.

$S[0] = A[0]$, since that is the greedy choice. Suppose $OPT[0] \neq A[0]$ (i.e. the optimal solution does not start with the greedy choice).

What would happen if you replaced $OPT[0]$ with $S[0]$ in the optimal solution?

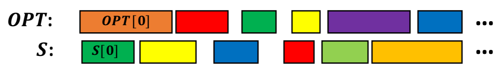
Activity Selection

Greedy decision: Select the next course with the earliest compatible finish time.

Proof of optimality: Let A be the set of courses, $S \subseteq A$ be the greedy algorithm selection, and $OPT \subseteq A$ be an optimal selection, all sorted by increasing finish time.

$S[0] = A[0]$, since that is the greedy choice. Suppose $OPT[0] \neq A[0]$ (i.e. the optimal solution does not start with the greedy choice).

What would happen if you replaced $OPT[0]$ with $S[0]$ in the optimal solution?



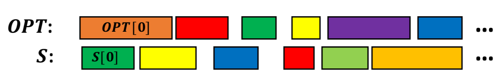
Activity Selection

Greedy decision: Select the next course with the earliest compatible finish time.

Proof of optimality: Let A be the set of courses, $S \subseteq A$ be the greedy algorithm selection, and $OPT \subseteq A$ be an optimal selection, all sorted by increasing finish time.

$S[0] = A[0]$, since that is the greedy choice. Suppose $OPT[0] \neq A[0]$ (i.e. the optimal solution does not start with the greedy choice).

Replacing $OPT[0]$ with $A[0]$ must also be an optimal solution:



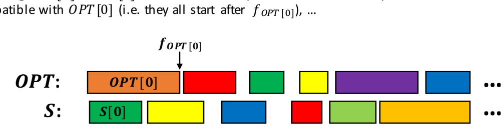
Activity Selection

Greedy decision: Select the next course with the earliest compatible finish time.

Proof of optimality: Let A be the set of courses, $S \subseteq A$ be the greedy algorithm selection, and $OPT \subseteq A$ be an optimal selection, all sorted by increasing finish time.

$S[0] = A[0]$, since that is the greedy choice. Suppose $OPT[0] \neq A[0]$ (i.e. the optimal solution does not start with the greedy choice).

Replacing $OPT[0]$ with $A[0]$ must also be an optimal solution: If every course in OPT is compatible with $OPT[0]$ (i.e. they all start after $f_{OPT[0]}$), …



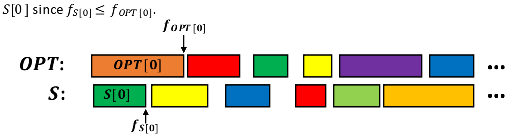
Activity Selection

Greedy decision: Select the next course with the earliest compatible finish time.

Proof of optimality: Let A be the set of courses, $S \subseteq A$ be the greedy algorithm selection, and $OPT \subseteq A$ be an optimal selection, all sorted by increasing finish time.

$S[0] = A[0]$, since that is the greedy choice. Suppose $OPT[0] \neq A[0]$ (i.e. the optimal solution does not start with the greedy choice).

Replacing $OPT[0]$ with $A[0]$ must also be an optimal solution: If every course in OPT is compatible with $OPT[0]$ (i.e. they all start after $f_{OPT[0]}$), they must be also be compatible with $S[0]$ since $f_{S[0]} \leq f_{OPT[0]}$.



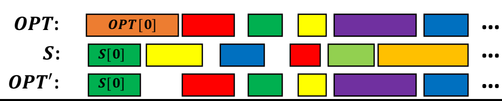
Activity Selection

Greedy decision: Select the next course with the earliest compatible finish time.

Proof of optimality: Let A be the set of courses, $S \subseteq A$ be the greedy algorithm selection, and $OPT \subseteq A$ be an optimal selection, all sorted by increasing finish time.

$S[0] = A[0]$, since that is the greedy choice. Suppose $OPT[0] \neq A[0]$ (i.e. the optimal solution does not start with the greedy choice).

Replacing $OPT[0]$ with $A[0]$ must also be an optimal solution: If every course in OPT is compatible with $OPT[0]$ (i.e. they all start after $f_{OPT[0]}$), they must be also be compatible with $S[0]$ since $f_{S[0]} \leq f_{OPT[0]}$. Thus, $OPT' = OPT \setminus OPT[0] \cup S[0]$ is also optimal.



Activity Selection

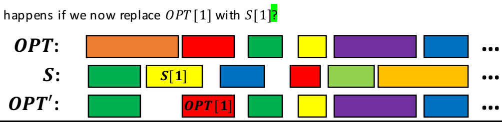
Greedy decision: Select the next course with the earliest compatible finish time.

Proof of optimality: Let A be the set of courses, $S \subseteq A$ be the greedy algorithm selection, and $OPT \subseteq A$ be an optimal selection, all sorted by increasing finish time.

$S[0] = A[0]$, since that is the greedy choice. Suppose $OPT[0] \neq A[0]$ (i.e. the optimal solution does not start with the greedy choice).

Replacing $OPT[0]$ with $A[0]$ must also be an optimal solution: If every course in OPT is compatible with $OPT[0]$ (i.e. they all start after $f_{OPT[0]}$), they must be also be compatible with $S[0]$ since $f_{S[0]} \leq f_{OPT[0]}$. Thus, $OPT' = OPT \setminus OPT[0] \cup S[0]$ is also optimal.

What happens if we now replace $OPT[1]$ with $S[1]$?



Activity Selection

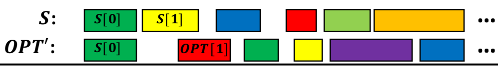
Greedy decision: Select the next course with the earliest compatible finish time.

Proof of optimality: Let A be the set of courses, $S \subseteq A$ be the greedy algorithm selection, and $OPT \subseteq A$ be an optimal selection, all sorted by increasing finish time.

$S[0] = A[0]$, since that is the greedy choice. Suppose $OPT[0] \neq A[0]$ (i.e. the optimal solution does not start with the greedy choice).

Replacing $OPT[0]$ with $A[0]$ must also be an optimal solution: If every course in OPT is compatible with $OPT[0]$ (i.e. they all start after $f_{OPT[0]}$), they must be also be compatible with $S[0]$ since $f_{S[0]} \leq f_{OPT[0]}$. Thus, $OPT' = OPT \setminus OPT[0] \cup S[0]$ is also optimal.

What happens if we now replace $OPT[1]$ with $S[1]$? We know that $S[1]$ is compatible with $S[0]$.



Activity Selection

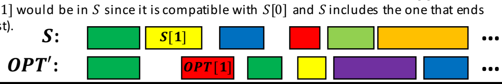
Greedy decision: Select the next course with the earliest compatible finish time.

Proof of optimality: Let A be the set of courses, $S \subseteq A$ be the greedy algorithm selection, and $OPT \subseteq A$ be an optimal selection, all sorted by increasing finish time.

$S[0] = A[0]$, since that is the greedy choice. Suppose $OPT[0] \neq A[0]$ (i.e. the optimal solution does not start with the greedy choice).

Replacing $OPT[0]$ with $A[0]$ must also be an optimal solution: If every course in OPT is compatible with $OPT[0]$ (i.e. they all start after $f_{OPT[0]}$), they must be also be compatible with $S[0]$ since $f_{S[0]} \leq f_{OPT[0]}$. Thus, $OPT' = OPT \setminus OPT[0] \cup S[0]$ is also optimal.

What happens if we now replace $OPT[1]$ with $S[1]$? We know that $S[1]$ is compatible with $S[0]$. We also know that $S[1]$ is compatible with $OPT[2]$, since $f_{S[1]} \leq f_{OPT[1]}$ (otherwise $OPT[1]$ would be in S since it is compatible with $S[0]$ and S includes the one that ends earliest).



Activity Selection

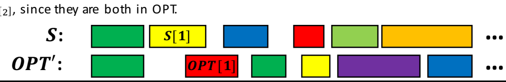
Greedy decision: Select the next course with the earliest compatible finish time.

Proof of optimality: Let A be the set of courses, $S \subseteq A$ be the greedy algorithm selection, and $OPT \subseteq A$ be an optimal selection, all sorted by increasing finish time.

$S[0] = A[0]$, since that is the greedy choice. Suppose $OPT[0] \neq A[0]$ (i.e. the optimal solution does not start with the greedy choice).

Replacing $OPT[0]$ with $A[0]$ must also be an optimal solution: If every course in OPT is compatible with $OPT[0]$ (i.e. they all start after $f_{OPT[0]}$), they must be also be compatible with $S[0]$ since $f_{S[0]} \leq f_{OPT[0]}$. Thus, $OPT' = OPT \setminus OPT[0] \cup S[0]$ is also optimal.

What happens if we now replace $OPT[1]$ with $S[1]$? We know that $S[1]$ is compatible with $S[0]$. We also know that $S[1]$ is compatible with $OPT[2]$, since $f_{S[1]} \leq f_{OPT[1]}$ and $f_{OPT[1]} \leq s_{OPT[2]}$, since they are both in OPT.



Activity Selection

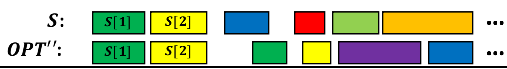
Greedy decision: Select the next course with the earliest compatible finish time.

Proof of optimality: Let A be the set of courses, $S \subseteq A$ be the greedy algorithm selection, and $OPT \subseteq A$ be an optimal selection, all sorted by increasing finish time.

$S[0] = A[0]$, since that is the greedy choice. Suppose $OPT[0] \neq A[0]$ (i.e. the optimal solution does not start with the greedy choice).

Replacing $OPT[0]$ with $A[0]$ must also be an optimal solution: If every course in OPT is compatible with $OPT[0]$ (i.e. they all start after $f_{OPT[0]}$), they must be also be compatible with $S[0]$ since $f_{S[0]} \leq f_{OPT[0]}$. Thus, $OPT' = OPT \setminus OPT[0] \cup S[0]$ is also optimal.

What happens if we now replace $OPT[1]$ with $S[1]$? We know that $S[1]$ is compatible with $S[0]$. We also know that $S[1]$ is compatible with $OPT[2]$, since $f_{S[1]} \leq f_{OPT[1]} \leq s_{OPT[2]}$.



Activity Selection

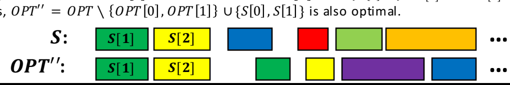
Greedy decision: Select the next course with the earliest compatible finish time.

Proof of optimality: Let A be the set of courses, $S \subseteq A$ be the greedy algorithm selection, and $OPT \subseteq A$ be an optimal selection, all sorted by increasing finish time.

$S[0] = A[0]$, since that is the greedy choice. Suppose $OPT[0] \neq A[0]$ (i.e. the optimal solution does not start with the greedy choice).

Replacing $OPT[0]$ with $A[0]$ must also be an optimal solution: If every course in OPT is compatible with $OPT[0]$ (i.e. they all start after $f_{OPT[0]}$), they must be also be compatible with $S[0]$ since $f_{S[0]} \leq f_{OPT[0]}$. Thus, $OPT' = OPT \setminus OPT[0] \cup S[0]$ is also optimal.

What happens if we now replace $OPT[1]$ with $S[1]$? We know that $S[1]$ is compatible with $S[0]$. We also know that $S[1]$ is compatible with $OPT[2]$, since $f_{S[1]} \leq f_{OPT[1]} \leq s_{OPT[2]}$.
Thus, $OPT'' = OPT \setminus \{OPT[0], OPT[1]\} \cup \{S[0], S[1]\}$ is also optimal.



Activity Selection

Greedy decision: Select the next course with the earliest compatible finish time.

Proof of optimality: Let A be the set of courses, $S \subseteq A$ be the greedy algorithm selection, and $OPT \subseteq A$ be an optimal selection, all sorted by increasing finish time.

$S[0] = A[0]$, since that is the greedy choice. Suppose $OPT[0] \neq A[0]$ (i.e. the optimal solution does not start with the greedy choice).

Replacing $OPT[0]$ with $A[0]$ must also be an optimal solution: If every course in OPT is compatible with $OPT[0]$ (i.e. they all start after $f_{OPT[0]}$), they must be also be compatible with $S[0]$ since $f_{S[0]} \leq f_{OPT[0]}$. Thus, $OPT' = OPT \setminus OPT[0] \cup S[0]$ is also optimal.

What happens if we now replace $OPT[1]$ with $S[1]$? We know that $S[1]$ is compatible with $S[0]$. We also know that $S[1]$ is compatible with $OPT[2]$, since $f_{S[1]} \leq f_{OPT[1]} \leq s_{OPT[2]}$. Thus, $OPT'' = OPT \setminus \{OPT[0], OPT[1]\} \cup \{S[0], S[1]\}$ is also optimal.

We can then proceed inductively and show that each course in OPT can be replaced by the corresponding course in S without violating compatibility restrictions. i.e., We translated OPT into S at no extra cost, thus S must be optimal.