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?

Input:

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

Activity Selection

Input:

- $S = \{a_1, a_2, ..., a_n\}$ 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: • $S = \{a_1, a_2, ..., a_n\}$ - 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.

Input:

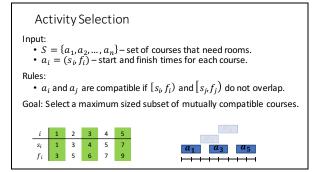
• $S = \{a_1, a_2, ..., a_n\}$ - set of courses that need rooms. • $a_i = (s_i, f_i)$ - start and finish times for each course.

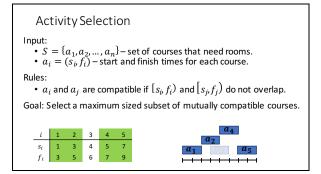
Rules:

• 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.

			2				
	Si	1	3	4	5	7	$\begin{bmatrix} a_1 \\ a_3 \end{bmatrix} \begin{bmatrix} a_5 \end{bmatrix}$
$f_i 3 5 6 7 9$	fi	3	5	6	7	9	

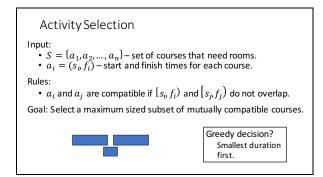






Activity Selection Input: • $S = \{a_1, a_2, ..., a_n\}$ - set of courses that need rooms. • $a_i = (s_i, f_i)$ - start and finish times for each course. Rules: • a_i and a_j are compatible if $[s_i, f_i)$ and $[s_j, f_j)$ do not overlap. Goal: Select a maximum sized subset of mutually compatible courses. $\frac{i}{s_i} \frac{1}{1} \frac{2}{3} \frac{3}{4} \frac{4}{5} \frac{5}{7}}{f_i}$ Greedy selection criteria? Algorithm Outline?

Activity Selection Input: • $S = \{a_1, a_2, ..., a_n\}$ - set of courses that need rooms. • $a_i = (s_b, f_i)$ – start and finish times for each course. Rules: • a_i and a_j are compatible if $[s_b, f_i)$ and $[s_j, f_j)$ do not overlap. Goal: Select a maximum sized subset of mutually compatible courses. Greedy decision? Smallest duration first.



Input:

• $S = \{a_1, a_2, ..., a_n\}$ - 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:

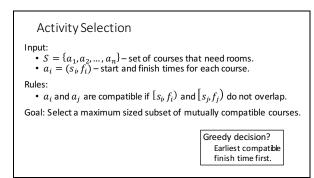
• $S = \{a_1, a_2, ..., a_n\}$ - set of courses that need rooms. • $a_i = (s_i, f_i)$ - start and finish times for each course.

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.

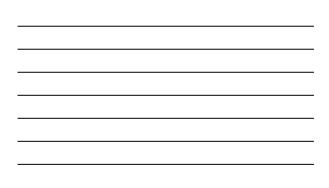
3	3	3		3	Greedy decision?
4		2	4		Smallest conflict
4			4		first.
4			4		



Activity Selection								
activity_selection(activities A)								
sort_by_finish(A)								
<pre>selected = {A[1]}</pre>								
last_added = 1								
for i = 2 to A.length								
if A[i].start ≥ A[last_added].fi	nish							
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							
	R	unnin	g Tim	e?			
activity_selection(activities A)	v	alidity	?				
<pre>sort_by_finish(A) selected = {A[1]} last_added = 1</pre>	Р	erform	nance	?			
<pre>for i = 2 to A.length</pre>							
if A[i].start ≥ A[last_added].fi	nısn						
selected = selected ∪ {A[i]}							
last_added = i						-	
return selected	i	1	2			5	
	Si	1 3	3	4	5	7	
	fi	3	5	6	7	9	

Activity Selection							
	R	unnin	g Tim	e? 0	(nlog	n)	
<pre>activity_selection(activities A) sort_by_finish(A)</pre>	v	alidity	?				
<pre>selected = {A[1]} last_added = 1</pre>	Ρ	erform	nancei	?			
<pre>for i = 2 to A.length if A[i].start ≥ A[last_added].fin</pre>	ish						
selected = selected ∪ {A[i]}							
last_added = i return selected	i	1	2	3	4	5	
Tetum serected	si	1	3	4	5	7	
	f_i	3	5	6	7	9	



Activity Selection							
	R	unnin	g Tim	e? 0	(nlog	n)	
activity_selection(activities A) sort_by_finish(A)		alidity f com				onsists	
<pre>selected = {A[1]} last_added = 1</pre>	Performance?						
for i = 2 to A.length							
if A[i].start ≥ A[last_added].fir	nish						
selected = selected $\cup \{A[i]\}$							
$last_added = i$	i	1	2	3	4	5	
return selected	si	1	3	4	5	7	
	f_i	3	3 5	6	7	9	

Activity Selection	P	unnin	a Tim	~ 0	(n log	. m)	
		umm	5	C: 01	(1106	10)	
activity_selection(activities A)						onsists	
<pre>sort_by_finish(A)</pre>	0	f com	patibi	e cou	rses.		
<pre>selected = {A[1]}</pre>	Р	erform	ance	ls s	seled	ted	
<pre>last_added = 1</pre>	tł	ne larg	gest p	ossibl	le sub	set?	
for i = 2 to A.length							
if $A[i]$.start $\geq A[last_added]$.fin	iisn						
selected = selected $\cup \{A[i]\}$							
last_added = i	i	1	2	3	4	5	
return selected	Si	1	3	4	5	7	
	fi	1 3	5	6	7	9	

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

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

<u>Proof of optimality</u>: 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).

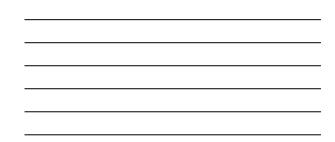
What would happen if you replaced OPT[0] with S[0] in the optimal solution

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

<u>Proof of optimality</u>: Let A be the set of course, $S \subseteq A$ 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.

<u>Proof of optimality</u>: 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

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

<u>Proof of optimality</u>: Let A be the set of course, $S \subseteq A$ 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 $f_{OPT[0]}$), ...

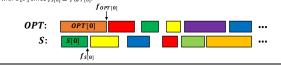


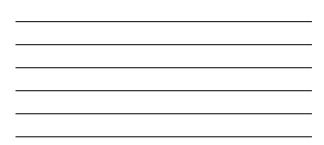
Greedy decision: Select the next course with the earliest compatible finish time.

<u>Proof of optimality:</u> 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.

<u>Proof of optimality</u>: 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).

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



Activity Selection

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

<u>Proof of optimality</u>: 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 $f_{S10} \leq f_{OPT}$ [0]. Thus, $OPT' = OPT \setminus OPT$ [0] $\cup S$ [0] is also optimal. What happens if we now replace OPT [1] with S [1]



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

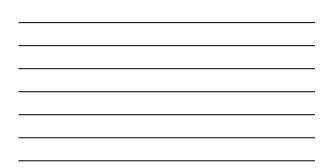
<u>Proof of optimality</u>: Let A be the set of courses, $S \subseteq A$ be the greedy algorithm selection, and $OPT \subseteq A$ be an optimal selection, all sorted by increasing finish time.

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].

<i>S</i> :	<i>S</i> [0]	<u><i>S</i>[1]</u>			•••	
OPT ':	<i>S</i> [0]	OPT [1]				



Activity Selection

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

<u>Proof of optimality</u>: 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).

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

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

 OPT [1] would be in S since it is compatible with S[0] and S includes the one that ends earliest).

 S:
 S[1]

 OPT [1]

Activity Selection

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

<u>Proof of optimality</u>: 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.

<i>S</i> :	<u><i>S</i>[1]</u>			
OPT ':	6	PT [1]		



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

<u>Proof of optimality</u>: 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 $f_{S[1]} \leq f_{OPT[1]} \leq S_{OPT}[2]$.

<i>S</i> :	<i>S</i> [1] <i>S</i> [2]	
OPT '':	<i>S</i> [1] <i>S</i> [2]	



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

<u>Proof of optimality</u>: 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).

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

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.

<i>S</i> :	<i>S</i> [1] <i>S</i> [2]	
OPT '':	<i>S</i> [1] <i>S</i> [2]	

Activity Selection

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

<u>Proof of optimality:</u> 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).

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

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 $f_{s[1]} \leq f_{ort[1]} \leq s_{ort[2]}$. 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.

