Greedy Algorithms

Greedy

Make best *local* choice, then solve remaining subproblem.

E.g. optimal solution uses the greedy choice + optimal solution to remaining subproblem

Unlike dynamic programming, we haven't solved the subproblems yet and don't need to pick the best subsolution.

Activity Selection

Given n activities $1, 2, \ldots, n$, the ith activity corresponding to an interval starting at s_i and finishing at f_i, find a compatible set with maximum size.

Make a choice: at each step, select the next activity to include in the set.

Is there a rule to construct largest set?

"Rules" for Activity Selection

- Earliest start time
- Earliest finish time
- Smallest interval
- Least conflicts

Make a decision that is good locally before consulting more subproblems.

"Rules" for Activity Selection

- Earliest start time
- **Earliest finish time**
- Smallest interval
- Least conflicts

Make a decision that is good locally before consulting more subproblems.

Activity Selection Algorithm

$R \leftarrow$ all activities
$A \leftarrow \{\}$
while $R \neq \{\}$ **do**
    let $t =$ activity in R with earliest finish time
    $R \leftarrow R \setminus \{s : s \text{ conflicts with } t, s \in R\}$
    $A \leftarrow A \cup \{t\}$
return A

Is this optimal?

Example

Find the largest subset of non-overlapping events, based on the following timetable:

event i	a	b	c	d	e	f	g	h	i	j	k
s_i	1	3	0	5	3	5	6	8	8	2	12
f_i	4	5	6	7	9	9	10	11	12	14	16

Optimal Substructure

Optimal substructure:

1. Let $\langle a_1, a_2, \ldots, a_i \rangle$ be an optimal schedule.
2. Assume subschedule $\langle a_k, \ldots, a_i \rangle$ is suboptimal for time after activity a_{k-1}.
3. So, $\exists$ a sequence $\langle b_1, \ldots, b_j \rangle$ that is a better schedule for our time interval $(j > i - k)$.
4. Then $\langle a_1, \ldots, a_{k-1}, b_1, \ldots, b_j \rangle$ must be a better schedule.
5. Then there is a better schedule than our optimal schedule. Our assumption must be false

Optimal Substructure

Optimal substructure:

1. Let $\langle a_1, a_2, \ldots, a_i \rangle$ be an optimal schedule.
2. Assume subschedule $\langle a_k, \ldots, a_i \rangle$ is suboptimal for time after activity a_{k-1}.
3. So, $\exists$ a sequence $\langle b_1, \ldots, b_j \rangle$ that is a better schedule for our time interval $(j > i - k)$.
4. Then $\langle a_1, \ldots, a_{k-1}, b_1, \ldots, b_j \rangle$ must be a better schedule.
5. Then there is a better schedule than our optimal schedule. Our assumption must be false

Optimal Substructure

Optimal substructure:

1. Let $\langle a_1, a_2, \ldots, a_i \rangle$ be an optimal schedule.
2. Assume subschedule $\langle a_k, \ldots, a_i \rangle$ is suboptimal for time after activity a_{k-1}.
3. So, $\exists$ a sequence $\langle b_1, \ldots, b_j \rangle$ that is a better schedule for our time interval $(j > i - k)$.
4. Then $\langle a_1, \ldots, a_{k-1}, b_1, \ldots, b_j \rangle$ must be a better schedule.
5. Then there is a better schedule than our optimal schedule. Our assumption must be false

Optimal Substructure

Optimal substructure:

1. Let $\langle a_1, a_2, \ldots, a_i \rangle$ be an optimal schedule.
2. Assume subschedule $\langle a_k, \ldots, a_i \rangle$ is suboptimal for time after activity a_{k-1}.
3. So, $\exists$ a sequence $\langle b_1, \ldots, b_j \rangle$ that is a better schedule for our time interval $(j > i - k)$.
4. Then $\langle a_1, \ldots, a_{k-1}, b_1, \ldots, b_j \rangle$ must be a better schedule.
5. Then there is a better schedule than our optimal schedule. Our assumption must be false

Optimal Substructure

Optimal substructure:

1. Let $\langle a_1, a_2, \ldots, a_i \rangle$ be an optimal schedule.
2. Assume subschedule $\langle a_k, \ldots, a_i \rangle$ is suboptimal for time after activity a_{k-1}.
3. So, $\exists$ a sequence $\langle b_1, \ldots, b_j \rangle$ that is a better schedule for our time interval $(j > i - k)$.
4. Then $\langle a_1, \ldots, a_{k-1}, b_1, \ldots, b_j \rangle$ must be a better schedule.
5. Then there is a better schedule than our optimal schedule. Our assumption must be false

The Greedy Choice Property

Greedy choice:

1. Let $\langle a_1, a_2, \ldots, a_i \rangle$ be an optimal schedule.
2. If a_1 is the activity with the earliest finish time, then the greedy choice is part of an optimal solution.
3. If a_1 does not have the earliest finish time, then $\exists$ an activity b with an earlier finish time $(f(b) < f(a_1))$.
4. Then $\langle b, a_2, \ldots, a_i \rangle$ must be an optimal solution.

This applies recursively to the subproblems:
Recall that $\langle a_2, \ldots, a_i \rangle$ is an optimal subsolution

The Greedy Choice Property

Greedy choice:

1. Let $\langle a_1, a_2, \ldots, a_i \rangle$ be an optimal schedule.
2. If a_1 is the activity with the earliest finish time, then the greedy choice is part of an optimal solution.
3. If a_1 does not have the earliest finish time, then $\exists$ an activity b with an earlier finish time $(f(b) < f(a_1))$.
4. Then $\langle b, a_2, \ldots, a_i \rangle$ must be an optimal solution.

This applies recursively to the subproblems:
Recall that $\langle a_2, \ldots, a_i \rangle$ is an optimal subsolution

The Greedy Choice Property

Greedy choice:

1. Let $\langle a_1, a_2, \ldots, a_i \rangle$ be an optimal schedule.
2. If a_1 is the activity with the earliest finish time, then the greedy choice is part of an optimal solution.
3. If a_1 does not have the earliest finish time, then $\exists$ an activity b with an earlier finish time $(f(b) < f(a_1))$.
4. Then $\langle b, a_2, \ldots, a_i \rangle$ must be an optimal solution.

This applies recursively to the subproblems:
Recall that $\langle a_2, \ldots, a_i \rangle$ is an optimal subsolution

The Greedy Choice Property

Greedy choice:

1. Let $\langle a_1, a_2, \ldots, a_i \rangle$ be an optimal schedule.
2. If a_1 is the activity with the earliest finish time, then the greedy choice is part of an optimal solution.
3. If a_1 does not have the earliest finish time, then $\exists$ an activity b with an earlier finish time $(f(b) < f(a_1))$.
4. Then $\langle b, a_2, \ldots, a_i \rangle$ must be an optimal solution.

This applies recursively to the subproblems:
Recall that $\langle a_2, \ldots, a_i \rangle$ is an optimal subsolution

Summary of Greedy Algorithms

Make the best *local* choice, then solve remaining subproblem.
An optimal solution uses the greedy choice + the optimal solution to the remaining subproblem.

1. prove greedy choice: can convert optimal solution to one that uses a greedy choice
2. prove optimal substructure: optimal solution uses optimal solutions of subproblems