

## Lecture 2, Sept 19

## Chapter 4: Greedy Algorithms

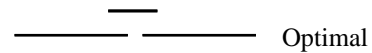
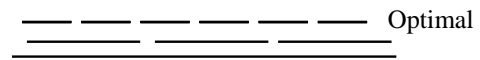
- A “greedy idea”: keep choosing some “local” *best*, and keep going without backtracking.
- No “global” insight while selecting the “local” best.
- Sometimes, it works.
- Defining “*best*” is the key challenge in devising greedy algorithms.

## Example: interval/activity scheduling

### Ingredients:

- Instances: Activities with starting and finishing times  $\langle\langle s_1, f_1 \rangle, \langle s_2, f_2 \rangle, \dots, \langle s_n, f_n \rangle\rangle$ .
- Solutions: A set of events that do not overlap.
- Payoff of Solution: The number of activities scheduled.
- Goal: Given a set of activities, schedule as many as possible.

## Examples



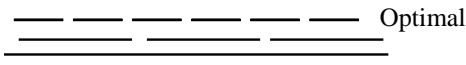
## Attempt 0

- Try all possible schedules, and choose the one that has no conflicts and the most activities scheduled.
- Impractical, because there are  $2^n$  possibilities – too long to run even for small  $n$ 's.

## Designing a greedy algorithm

- The main ingredient in designing a greedy algorithm is choosing the right criterion for making the greedy choices.
- Let's try some of the natural potential criteria.

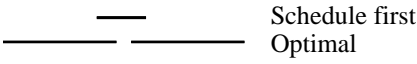
### Attempt 1



Optimal

Greedy Criteria: The Shortest Event

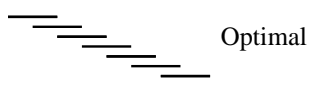
Motivation: Does not book the room for a long period of time.



Schedule first  
Optimal

Counter Example

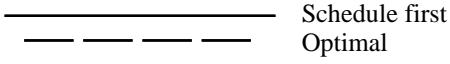
### Attempt 2



Optimal

Greedy Criteria: The Earliest Starting Time

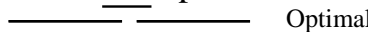
Motivation: Common scheduling algorithm.



Schedule first  
Optimal

Counter Example

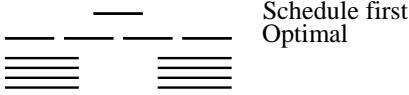
### Attempt 3



Optimal

Greedy Criteria:  
Conflicting with the Fewest Other Events

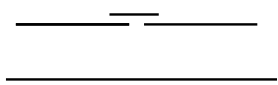
Motivation: So many can still be scheduled.



Schedule first  
Optimal

Counter Example

### Attempt 4



Optimal

Greedy Criteria: Earliest Finishing Time

Motivation: Schedule the event who will free up your room for someone else as soon as possible. **Works!**

### Outline of the algorithm

- Order the activities in non-decreasing order of finishing times. Denote it by  $A_1, A_2, \dots, A_n$ .
- For  $i=1, \dots, n$ :
  - If  $A_i$  does not conflict with previous choices, include it in the schedule.

### Proof of correctness

- Important to prove correctness, since we've seen other "natural" algorithms turning out to be wrong.
- The strategy of the proof is common to many greedy algorithms.

### Proof – cont'd

- The algorithm makes two types of irrevokable decisions:
  - Accepting an event, or
  - rejecting an event.
- Once an event is either accepted or rejected, the decision is never reconsidered.

### Proof – cont'd

- The goal is to show that no bad choices have been made – any partial solution of the algorithm can be extended to an optimal solution with the remaining activities.
- Then when the algorithm finishes, it has the optimal solution (there are no activities remaining).

### Outline of the proof – generic greedy algorithm

- Since there is not backtracking in the greedy algorithms, we must show that the algorithm never makes irreversible mistakes.
- In other words, the partial solution the algorithm currently holds can be extended to an optimal solution.
- At the end, when the algorithm holds a solution, it must be optimal (a solution is a partial solution).

### Generic greedy algorithms – cont'd

- We say that a partial solution is promising, if it can be extended to an optimal solution.
- Prove by induction the loop invariant:  
*“any partial solution the algorithm holds is promising”*
- In the interval scheduling case, we prove that the interval selections  $S_i$  for the first  $i$  intervals is promising.

### Generic greedy algorithms – cont'd

- The induction step is usually done using a replacement argument.
- “Massage” the optimal solution extending the  $i$ -th step to extend the  $i+1$ -st step.
- For interval scheduling, if the solution  $S_i^\wedge$  was not extending  $S_{i+1}$ , it was modified by replacing some interval in  $S_i^\wedge$  by an interval the algorithm has chosen to obtain  $S_{i+1}^\wedge$ .

### Generic greedy algorithms – cont'd

- Finally, use the fact that the last partial solution is promising to show that the algorithm works.
- For interval scheduling, the solution  $S_n$  (which includes *all* the choices that algorithm made about *all* the intervals) is promising, and hence is optimal.