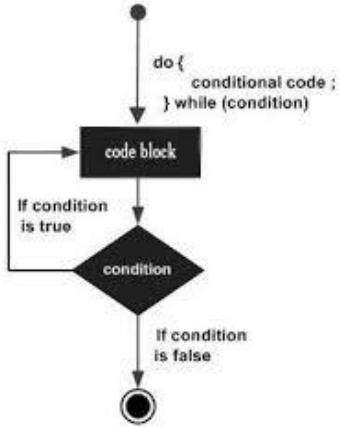
Lecture 10

Greedy Algorithms I



Announcements

- Homework 4 due Sunday night
- Group Meetings continue
- Quiz 1 due next Friday
  - Released earlier next week
  - A few questions from each topic

Greedy Algorithms

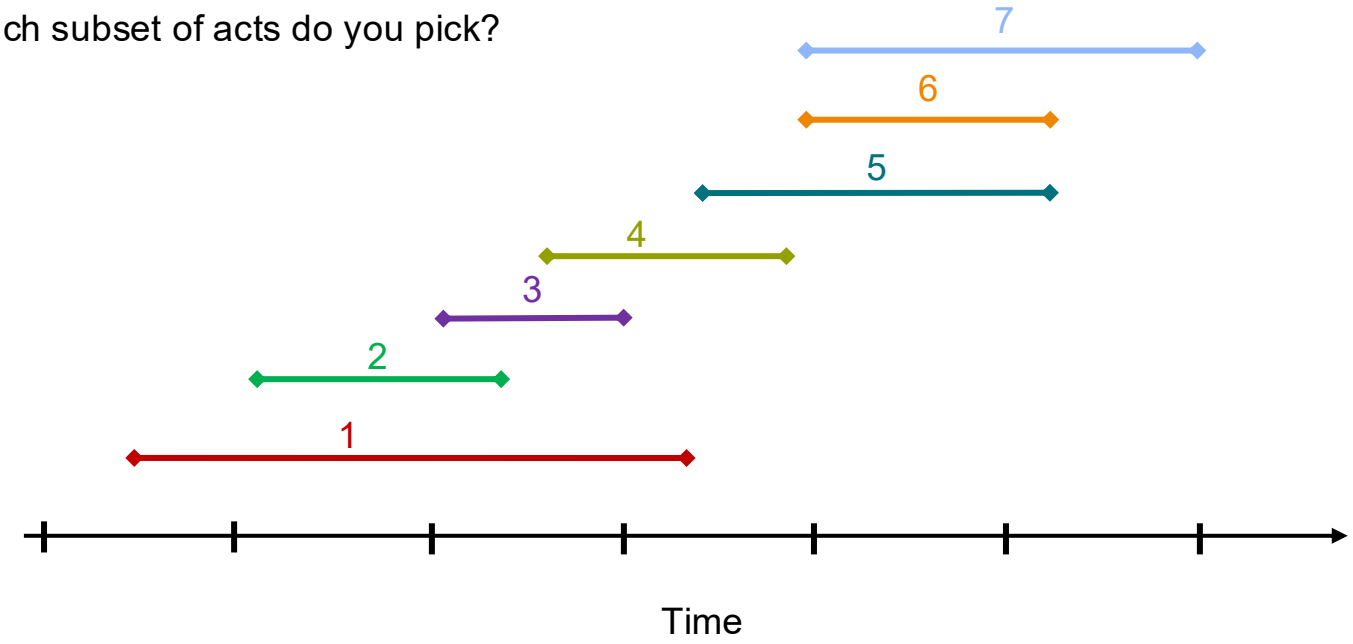
We are moving on to our study of algorithm design

- **Greedy**
- Divide-and-conquer
- Dynamic Programming
- Network Flow

Interval Scheduling

Imagine you are going to a music festival and want to see as many different acts as possible

- Which subset of acts do you pick?



Interval Scheduling

Let write this problem more generally

- Acts $1, 2, \ldots, n$
- $s(i)$: start time of act i
- $f(i)$: finish time of act i
- Acts i, j are **compatible** if they do not overlap
- Set A of acts is **compatible** if all pairs in A are compatible
- Set A of acts is **optimal** if it is compatible and no other compatible set is larger

Interval Scheduling

Let write this problem more generally

- Acts $1, 2, \ldots, n$
- $s(i)$: start time of act i
- $f(i)$: finish time of act i
- Acts i, j are **compatible** if they do not overlap
- Set A of acts is **compatible** if all pairs in A are compatible
- Set A of acts is **optimal** if it is compatible and no other compatible set is larger

Goal: find optimal set of acts

Greedy Algorithms

Main idea is to make one choice at a time

- Roughly: choose the thing that looks best and never look back
- We will sort arts in some "natural order" and choose acts one by one if they are compatible with the acts already chosen

Greedy Algorithms

Main idea is to make one choice at a time

- Roughly: choose the thing that looks best and never look back
- We will sort arts in some "natural order" and choose acts one by one if they are compatible with the acts already chosen

Algorithm sketch:

$R \leftarrow$ set of all shows sorted by some property
$A \leftarrow \emptyset$
while R is not empty **do**
    take first show i from R
    add i to A
    delete i and all overlapping acts from R

Exercise I

Q: Suppose an algorithm includes a step that sorts a list of n items. Then its running time is:

a) $O(n \log n)$

b) $\Theta(n \log n)$

c) $\Omega(n \log n)$

d) None of the above

Exercise I

Q: Suppose an algorithm includes a step that sorts a list of n items. Then its running time is:

a) $O(n \log n)$

b) $\Theta(n \log n)$

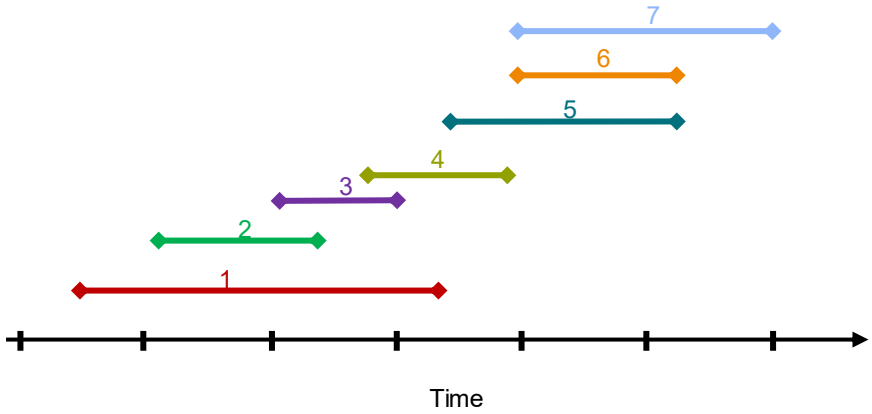
c) $\Omega(n \log n)$

d) None of the above

We can't provide an upper bound, because we don't know everything the algorithm is doing; however, we can provide a lower bound because MergeSort is $\Theta(n \log n)$

"Natural Order"

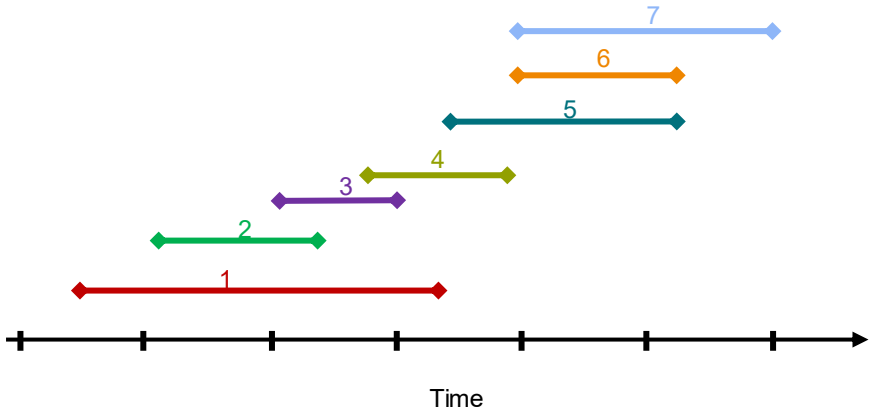
Start time: consider acts in ascending order of $s(i)$



"Natural Order"

Start time: consider acts in ascending order of $s(i)$

- Not optimal in this example

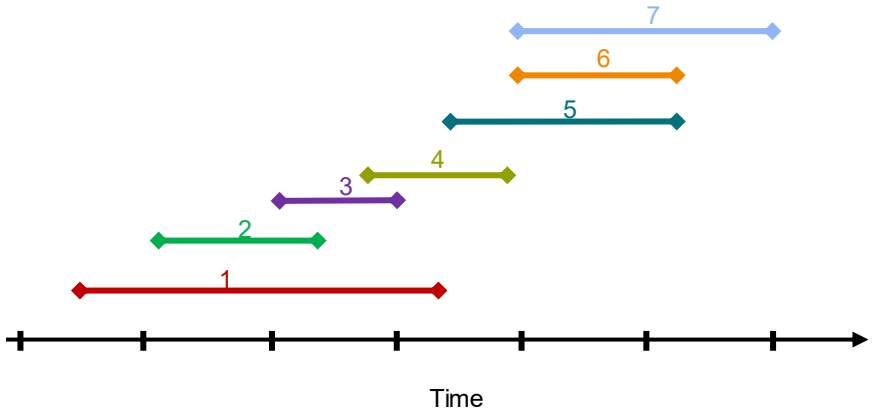


"Natural Order"

Start time: consider acts in ascending order of $s(i)$

- Not optimal in this example

Shortest time: consider acts in ascending order of $f(i) - s(i)$



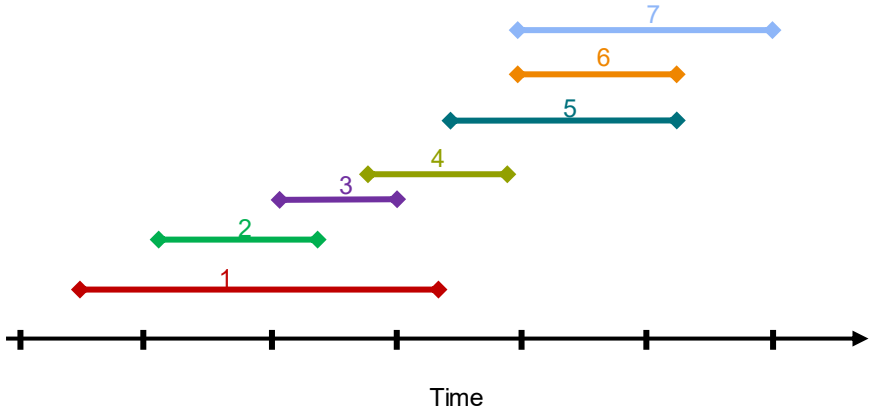
"Natural Order"

Start time: consider acts in ascending order of $s(i)$

- Not optimal in this example

Shortest time: consider acts in ascending order of $f(i) - s(i)$

- Not optimal in this example



"Natural Order"

Start time: consider acts in ascending order of $s(i)$

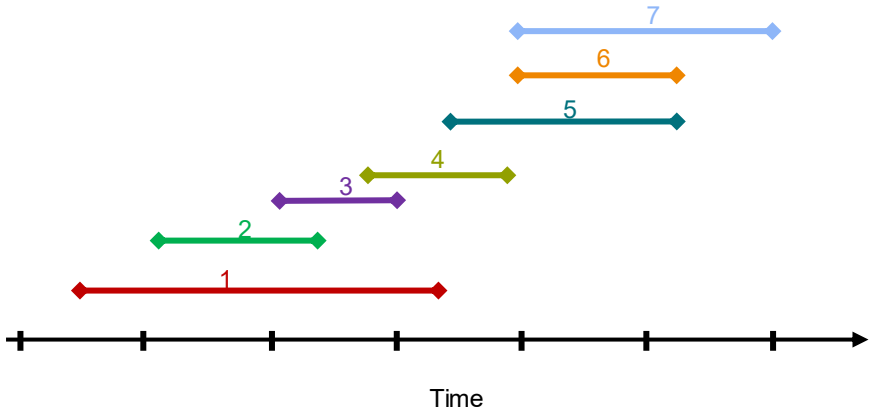
- Not optimal in this example

Shortest time: consider acts in ascending order of $f(i) - s(i)$

- Not optimal in this example

Fewest conflicts: sort in ascending order of $c(i)$, the number of acts overlapping with act i

- Optimal in this example



"Natural Order"

Start time: consider acts in ascending order of $s(i)$

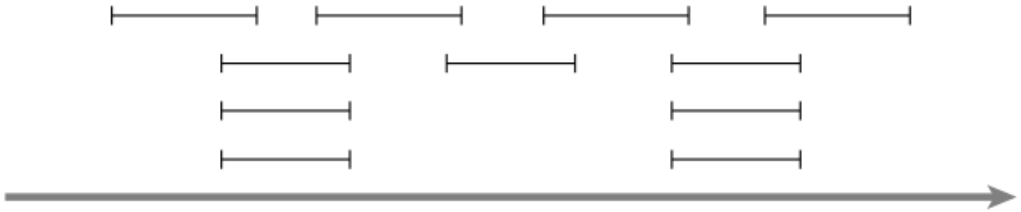
- Not optimal in this example

Shortest time: consider acts in ascending order of $f(i) - s(i)$

- Not optimal in this example

Fewest conflicts: sort in ascending order of $c(i)$, the number of acts overlapping with act i

- Optimal in this example
- But not in this one



"Natural Order"

Start time: consider acts in ascending order of $s(i)$

- Not optimal in this example

Shortest time: consider acts in ascending order of $f(i) - s(i)$

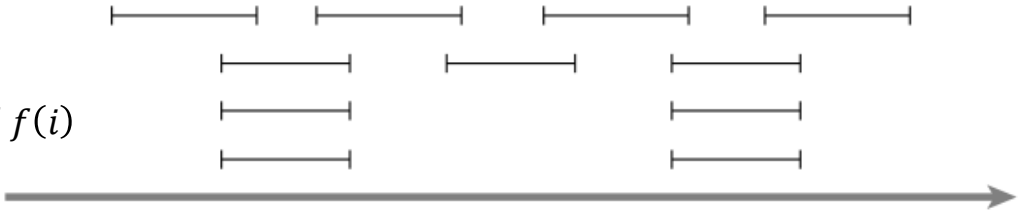
- Not optimal in this example

Fewest conflicts: sort in ascending order of $c(i)$, the number of acts overlapping with act i

- Optimal in this example
- But not in this one

Finish time: consider acts in ascending order of $f(i)$

- We will show this is always optimal



Analysis

Let A be the set of acts returned by the algorithm when acts are sorted by finish time.
What do we need to prove?

- A is compatible (obvious property of algorithm)
- A is optimal

We will prove A is optimal by a "greedy stays ahead" argument

Greedy Stays Ahead

Let $A = i_1, \ldots, i_k$ be the intervals selected by our algorithm

Let $O = j_1, \ldots, j_m$ be the intervals of an optimal solution O

- There may be many optimal solutions
- Assume both are sorted by finish time

Greedy Stays Ahead

Let $A = i_1, \ldots, i_k$ be the intervals selected by our algorithm

Let $O = j_1, \ldots, j_m$ be the intervals of an optimal solution O

- There may be many optimal solutions
- Assume both are sorted by finish time

Could it be the case that $m > k$?

- If A is an optimal solution, $m = k$

Greedy Stays Ahead

Let $A = i_1, \ldots, i_k$ be the intervals selected by our algorithm

Let $O = j_1, \ldots, j_m$ be the intervals of an optimal solution O

- There may be many optimal solutions
- Assume both are sorted by finish time

Could it be the case that $m > k$?

- If A is an optimal solution, $m = k$

Observation: $f(i_1) \leq f(j_1)$.

- The first show in A finishes no later than the first show in O

Greedy Stays Ahead

Let $A = i_1, \ldots, i_k$ be the intervals selected by our algorithm

Let $O = j_1, \ldots, j_m$ be the intervals of an optimal solution O

- There may be many optimal solutions
- Assume both are sorted by finish time

Could it be the case that $m > k$?

- If A is an optimal solution, $m = k$

Observation: $f(i_1) \leq f(j_1)$.

- The first show in A finishes no later than the first show in O

Claim: $f(i_r) \leq f(j_r)$ for all $r = 1, 2, 3, \ldots$

Greedy Stays Ahead

Claim: $f(i_r) \leq f(j_r)$ for all $r = 1, 2, 3, \ldots$

- Proof by induction on r
- Base case ($r = 1$): i_1 is the first choice of the greedy algorithm, which has the earliest overall finish time, so $f(i_1) \leq f(j_1)$

Greedy Stays Ahead

Claim: $f(i_r) \leq f(j_r)$ for all $r = 1, 2, 3, \ldots$

- Proof by induction on r
- Base case ($r = 1$): i_1 is the first choice of the greedy algorithm, which has the earliest overall finish time, so $f(i_1) \leq f(j_1)$

We still need to show the inductive step

- Prove that if $f(i_{r-1}) \leq f(j_{r-1})$, then $f(i_r) \leq f(j_r)$

Inductive Step

Assume for induction that $f(i_{r-1}) \leq f(j_{r-1})$ for $r \geq 2$

- j_r is compatible with j_{r-1}, so $f(j_{r-1}) \leq s(j_r)$
- By inductive hypothesis, $f(i_{r-1}) \leq f(j_{r-1})$
- Thus, $f(i_{r-1}) \leq s(j_r)$ and interval j_r is in the set of available intervals when try to select i_r
- Since we greedily select the earliest finish time, $f(i_r) \leq f(j_r)$

Exercise II

Q: Recall that k is the number of intervals in the greedy solution and m is the number of intervals in an optimal solution. What have we just proven?

a) $f(i_r) \leq f(j_r)$ for all $r = 1, 2, \dots, m$

b) $f(i_r) \leq f(j_r)$ for all $r = 1, 2, \dots, k$

c) The greedy algorithm is optimal

d) None of the above

Exercise II

Q: Recall that k is the number of intervals in the greedy solution and m is the number of intervals in an optimal solution. What have we just proven?

a) $f(i_r) \leq f(j_r)$ for all $r = 1, 2, \dots, m$

b) $f(i_r) \leq f(j_r)$ for all $r = 1, 2, \dots, k$

c) The greedy algorithm is optimal

d) None of the above

Optimality

Can it be the case that $k < m$?

- For contradiction, suppose $k < m$
- Let's consider interval j_{k+1}
- We know that $f(i_k) \leq f(j_k) \leq s(j_{k+1})$
- So, interval j_{k+1} is compatible with greedy intervals A
- However, the greedy algorithm did not add j_{k+1} to A $(\rightarrow\leftarrow)$

Therefore, $k = m$

Running Time

$R \leftarrow$ set of all shows sorted by some property
$A \leftarrow \emptyset$
while R is not empty **do**
    take first show i from R
    add i to A
    delete i and all overlapping acts from R

Can we make the loop better than n^2?

Running Time

$R \leftarrow$ set of all shows sorted by some property
$A \leftarrow \emptyset$; end $\leftarrow 0$
for i from 1 to n **do**
    if $s(i) \geq$ end **then**
    add i to A
    end = $f(i)$

$\Theta(n \log n)$

- Running time dominated by sort
- Reduced loop to linear time swapping from while to for

Next Time

Greedy: make a single "greedy" choice at a time and don't look back

Focusing on proof techniques for correctness

- "Greedy Stays Ahead" (today)
- Exchange arguments (next time)