NPTEL MOOC,JAN-FEB 2015
Week 6, Module 3

DESIGN AND ANALYSIS OF ALGORITHMS

Greedy algorithms: Interval scheduling

MADHAVAN MUKUND, CHENNAI MATHEMATICAL INSTITUTE
http://www.cmi.ac.in/~madhavan

Greedy Algorithms

- Need to make a sequence of choices to achieve a global optimum
- At each stage, make the next choice based on some local criterion
  - Drastically reduces space to search for solutions
- Never go back and revise an earlier decision
- How to prove that local choices achieve global optimum?

Examples so far

Dijkstra's algorithm

- Local rule:
  Freeze the distance of nearest unburnt vertex
- Global optimum:
  Distance assigned to each vertex is shortest distance from source

Examples so far

Prim's algorithm

- Local rule:
  Add to the spanning tree the nearest vertex not yet in the tree
- Global optimum:
  Final spanning tree constructed is a minimum cost spanning tree

Examples so far

Kruskal's algorithm

- Local rule:
  Add to the current set of edges the next smallest edge that does not form a cycle
- Global optimum:
  Edges collected form a minimum cost spanning tree

Interval scheduling

- CMI has a special video classroom for delivering online lectures
- Different teachers want to book the classroom — the slot for each instructor i starts at s(i) and finishes at f(i)
- Slots may overlap, so not all bookings can be honoured
- Choose a subset of bookings to maximize the number of teachers who get to use the room

Interval scheduling …

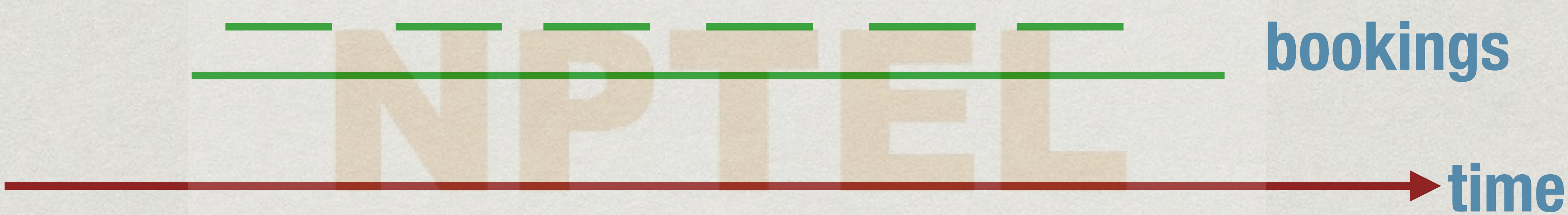
Greedy approach

- Pick the next booking to allot based on a local strategy
- Remove all bookings that overlap with this slot
- Argue that this sequence of bookings will maximize the number of teachers who get to use the room

Interval scheduling …

Greedy strategy 1

- Choose the booking whose start time is earliest
- Counterexample



Interval scheduling …

Greedy strategy 2

- Choose the booking whose interval is shortest
- Counterexample

Interval scheduling …

Greedy strategy 3

- Choose the booking that overlaps with minimum number of other bookings
- Counterexample

Interval scheduling …

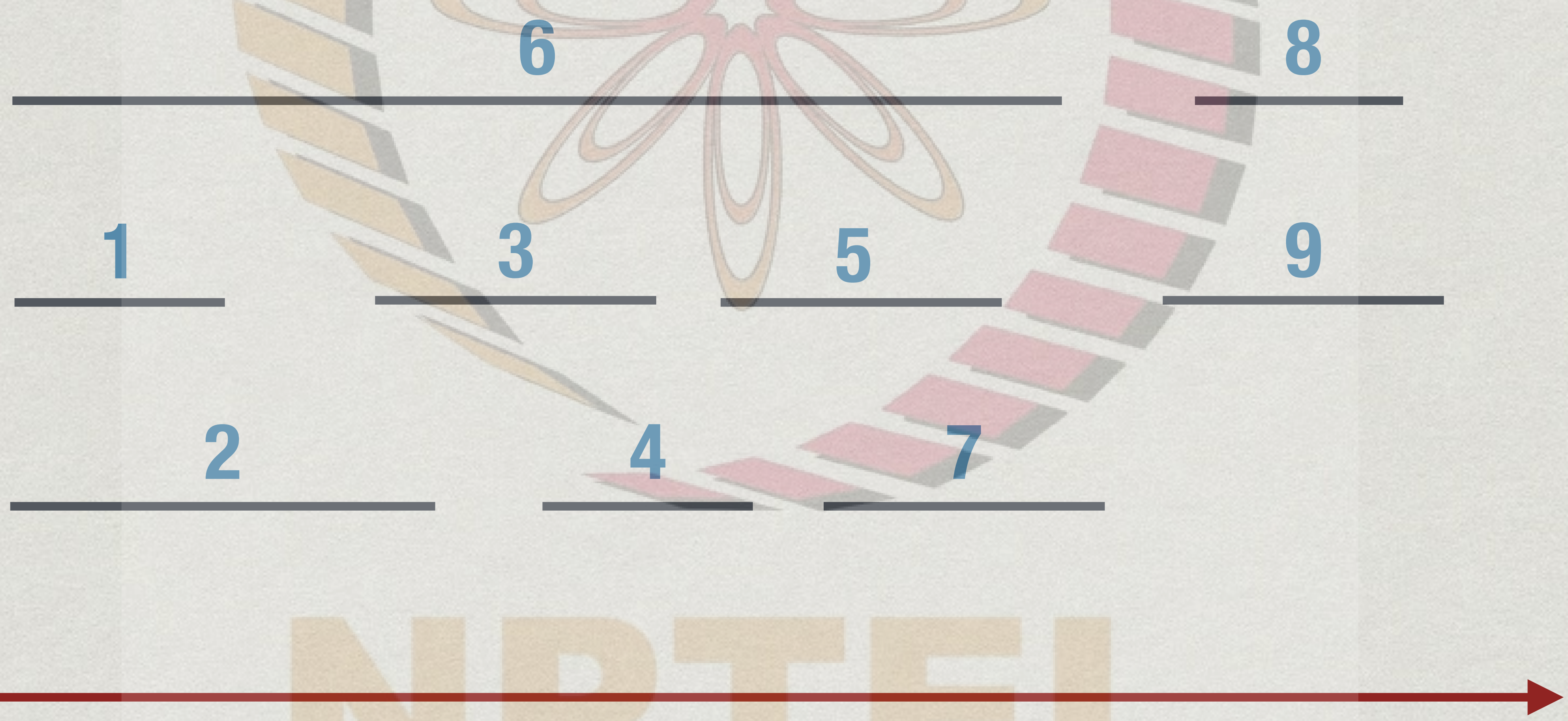
Greedy strategy 4

- Choose the booking that whose finish time is earliest
- Counterexample?
- Proof of correctness?

The algorithm

- B is the set of bookings
- A is the set of accepted bookings, initially empty
- While B is not empty
  - Pick b in B with smallest finishing time
  - Add b to A
  - Remove from B all bookings that overlap with b

The algorithm in action

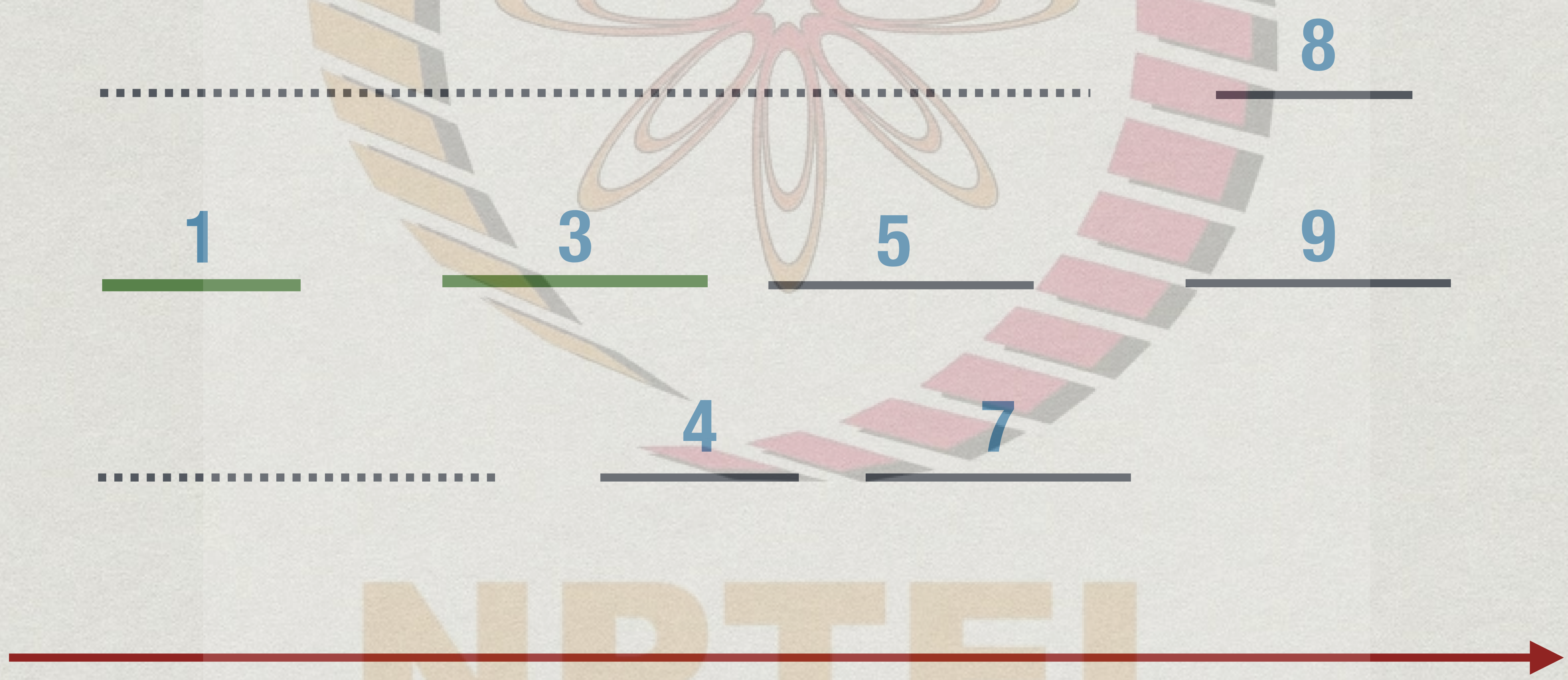


The algorithm in action

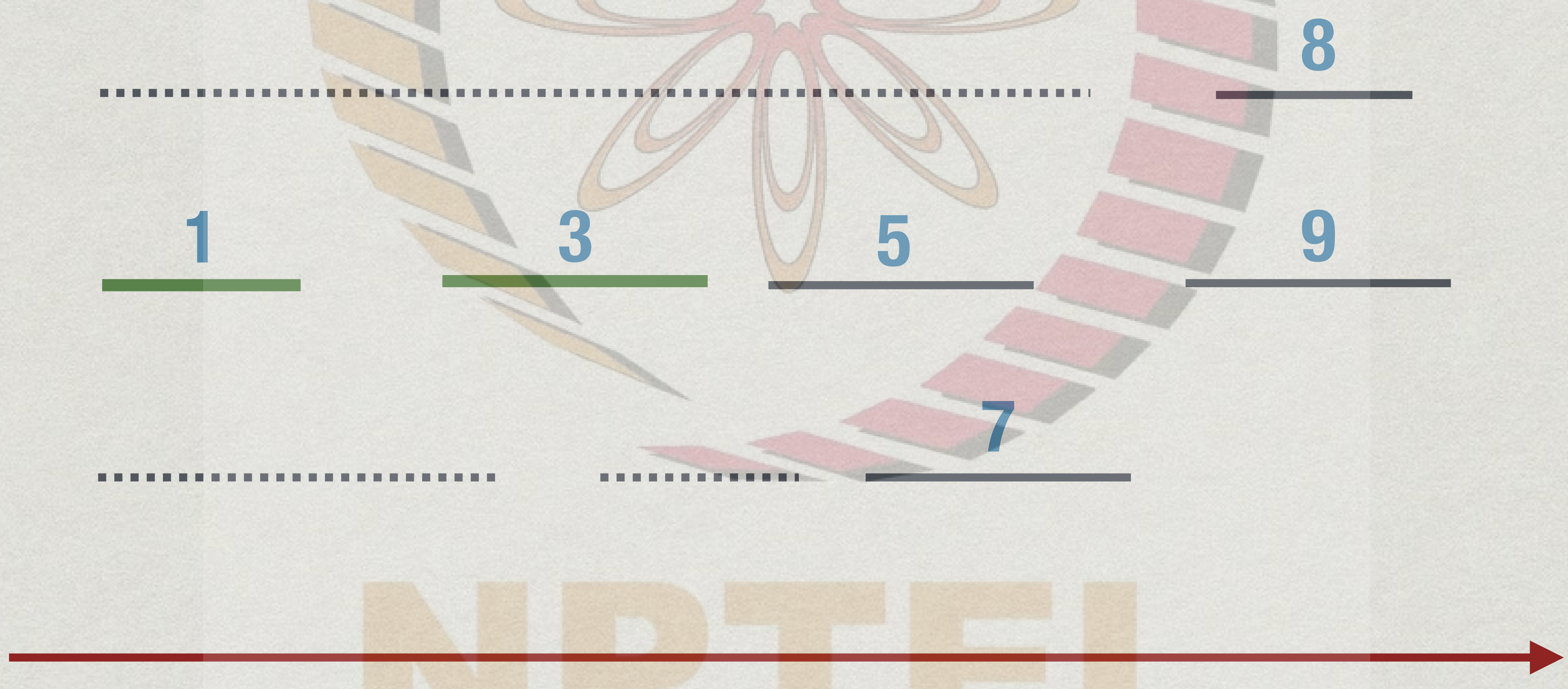
The algorithm in action

The algorithm in action

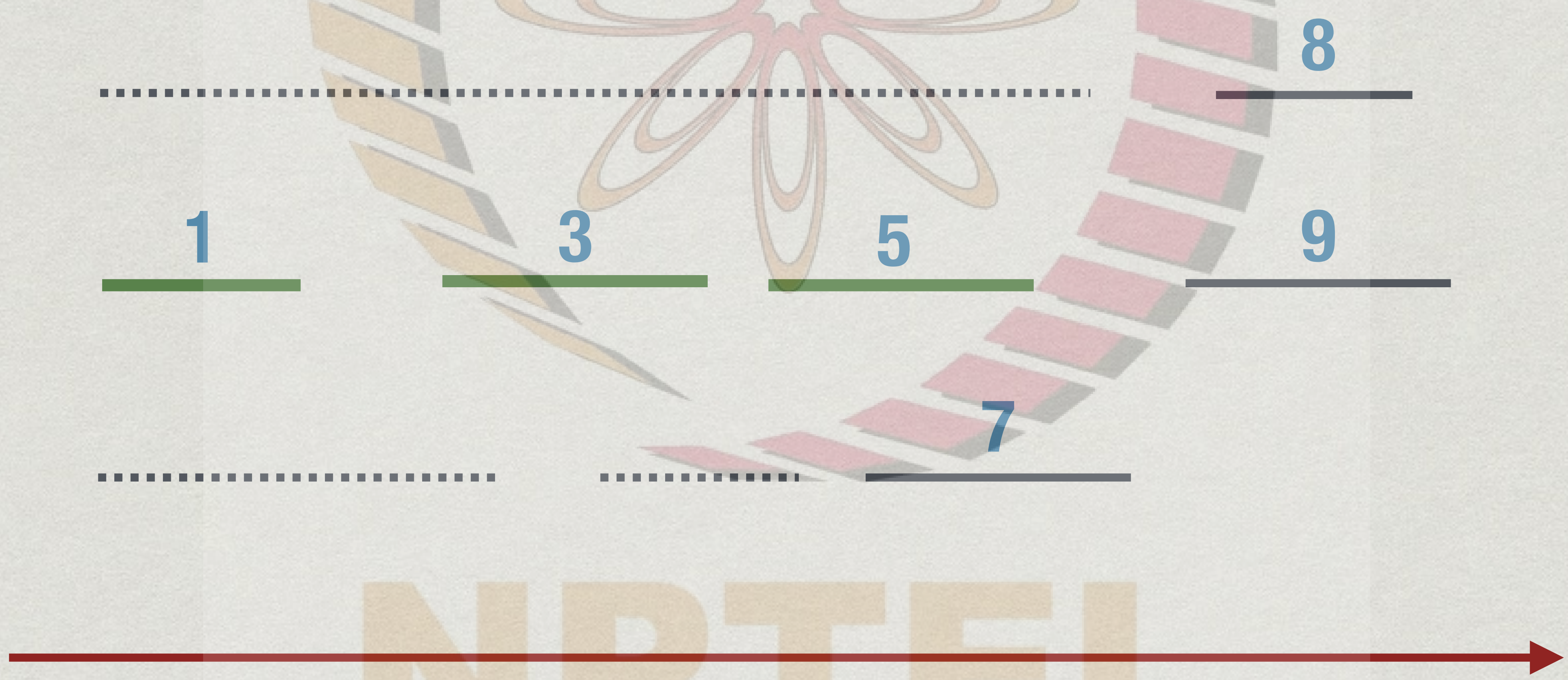
The algorithm in action



The algorithm in action



The algorithm in action



The algorithm in action

The algorithm in action

The algorithm in action

Correctness

- Our algorithm produces a solution A
- Let O be any optimal allocation of bookings
- A and O need not be identical
  - Can have multiple allocations of same size
- Instead, just show that $|A| = |O|$ — same size

Greedy allocation stays ahead

- Let $A = i_1, i_2, \ldots, i_k$
  - Assume sorted: $f(i_1) \leq s(i_2)$, $f(i_2) \leq s(i_3)$, …
- Let $O = j_1, j_2, \ldots, j_m$
  - Again, assume sorted: $f(j_1) \leq s(j_2)$, $f(j_2) \leq s(j_3)$, …
- To show that $k = m$

Greedy allocation stays ahead

Claim: For each $r \leq k$, $f(i_r) \leq f(j_r)$

- Our greedy solution "stays ahead" of O

Proof: By induction on r

- $r = 1$: our algorithm chooses booking i_1 with earliest overall finish time

Greedy allocation stays ahead

- $r > 1$: Assume, by induction that $f(i_{r-1}) \leq f(j_{r-1})$
- Then, it must be the case that $f(i_r) \leq f(j_r)$
- If not, algorithm would choose j_r rather than i_r

Greedy allocation is optimal

- Suppose $m > k$
- We know that $f(i_k) \leq f(j_k)$
- Consider booking j_{k+1} in O
  - Greedy algorithm terminates when B is empty
  - Since $f(i_k) \leq f(j_k) \leq s(j_{k+1})$, this booking is compatible with $A = i_1, i_2, \ldots, i_k$
  - After selecting i_k, B still contains j_{k+1}. Contradiction!

Implementation, complexity

- Initially, sort the n bookings by finish time, O(n log n)
  - Bookings are renumbered 1,2,…,n in this order
- Set up an array ST[1..n] so that ST[i] = s(i)
- Start with booking 1
- After choosing booking j, scan ST[j+1], ST[j+2], … and choose first k such that ST[k] > f(j)
- Second phase is O(n), so O(n log n) overall