Greedy Algorithms

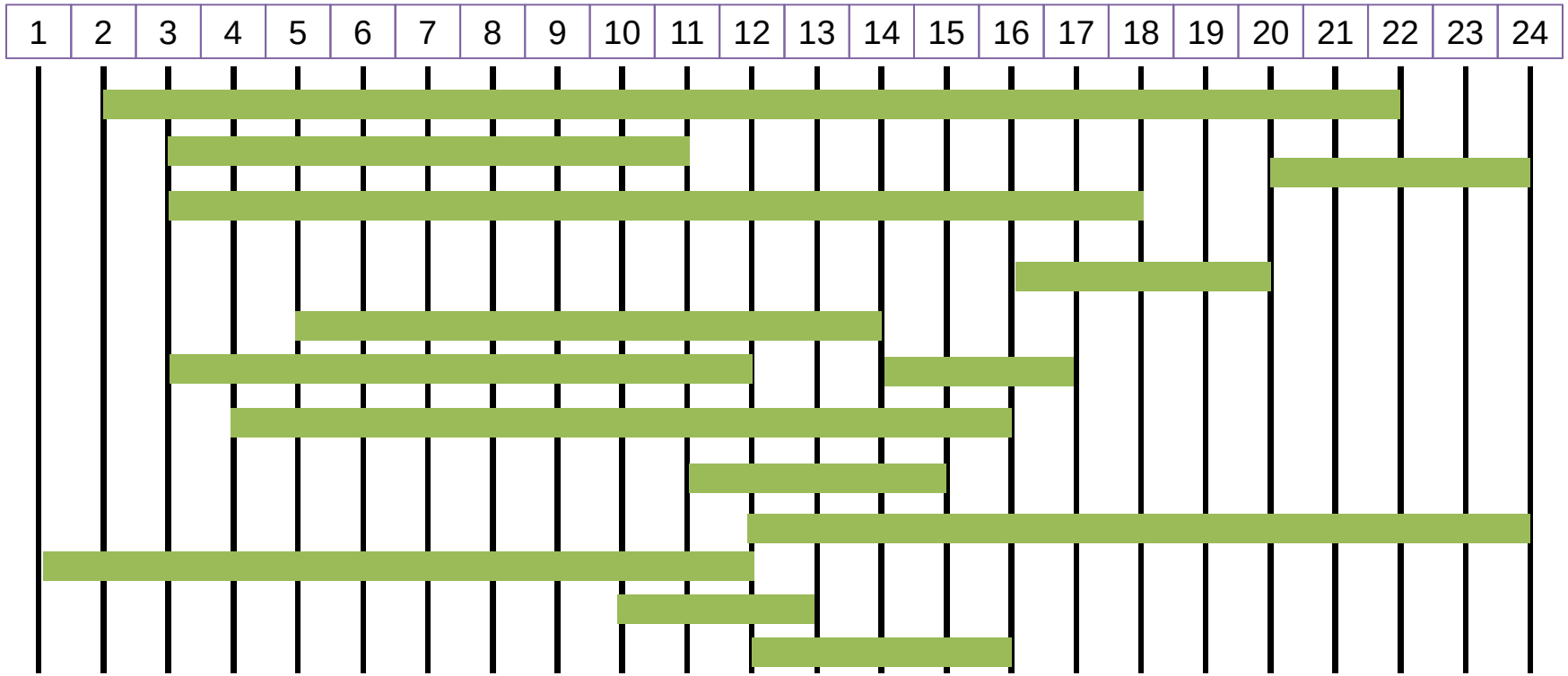
CSE 101: Design and Analysis of Algorithms

Lecture 9

CSE 101: Design and analysis of algorithms

- Greedy algorithms
  - Reading: Kleinberg and Tardos, sections 4.1, 4.2, and 4.3
- Homework 4 is due Oct 30, 11:59 PM

Event scheduling

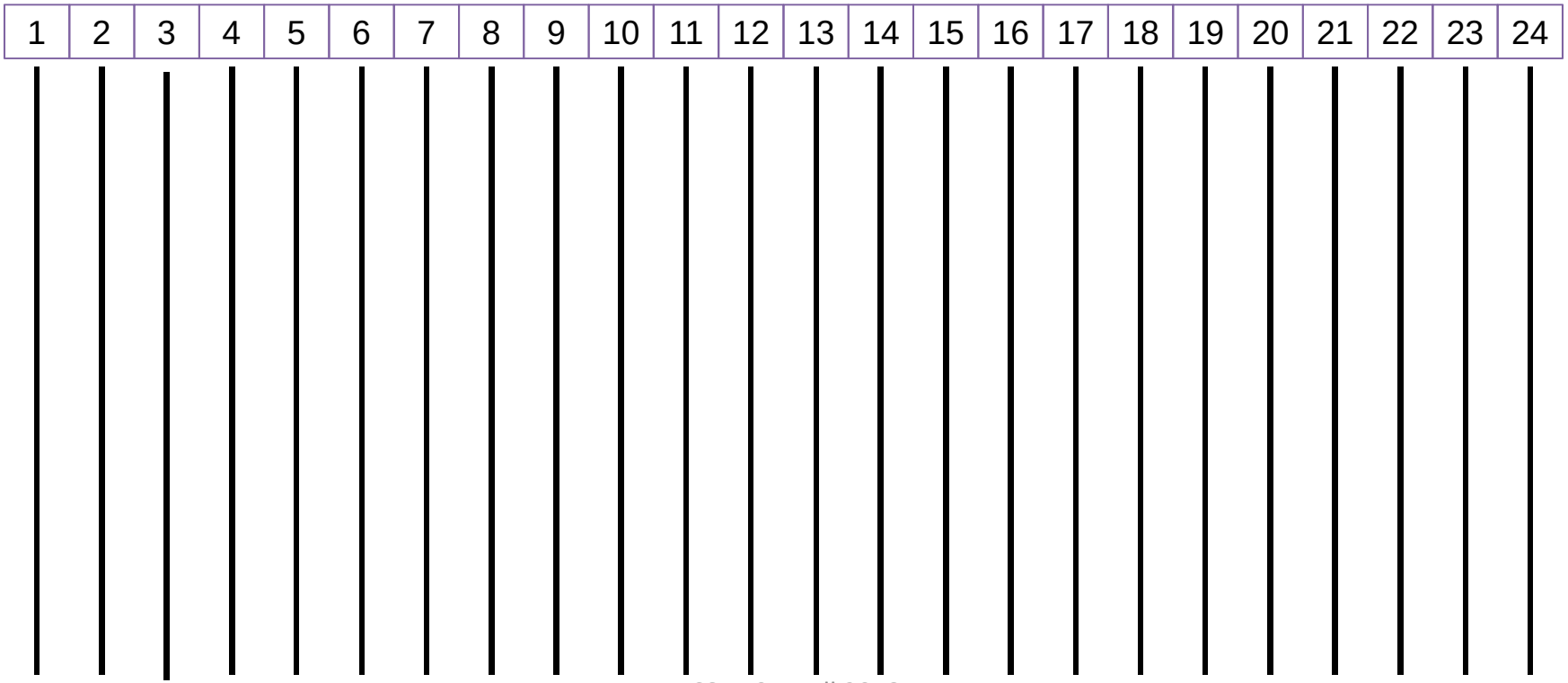


Based on slides courtesy of Miles Jones

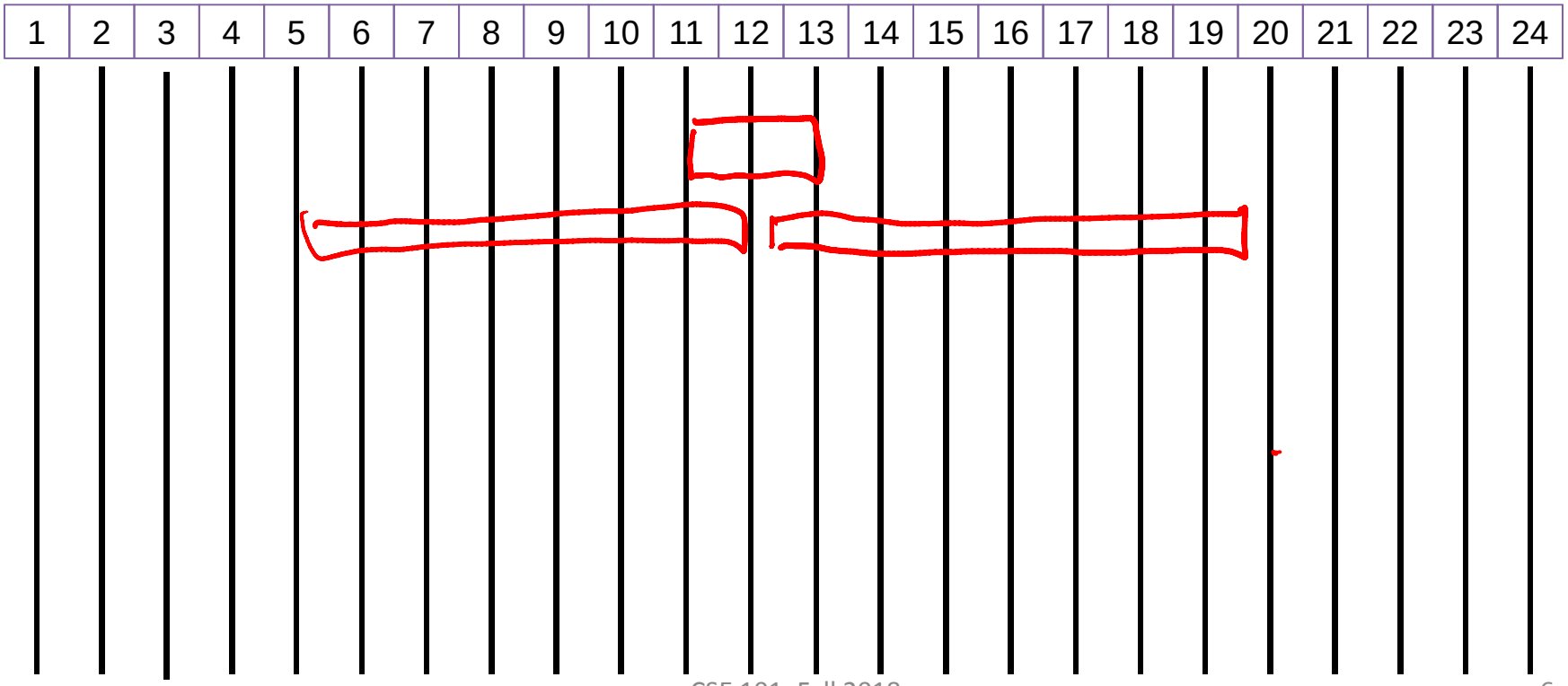
Event scheduling

- Your goal is to schedule the most events possible that day such that no two events overlap
- Exponential is too slow. Let's try some greedy strategies:
  - Shortest duration
  - Earliest start time
  - Fewest conflicts (take away most conflicts)
  - Earliest end time

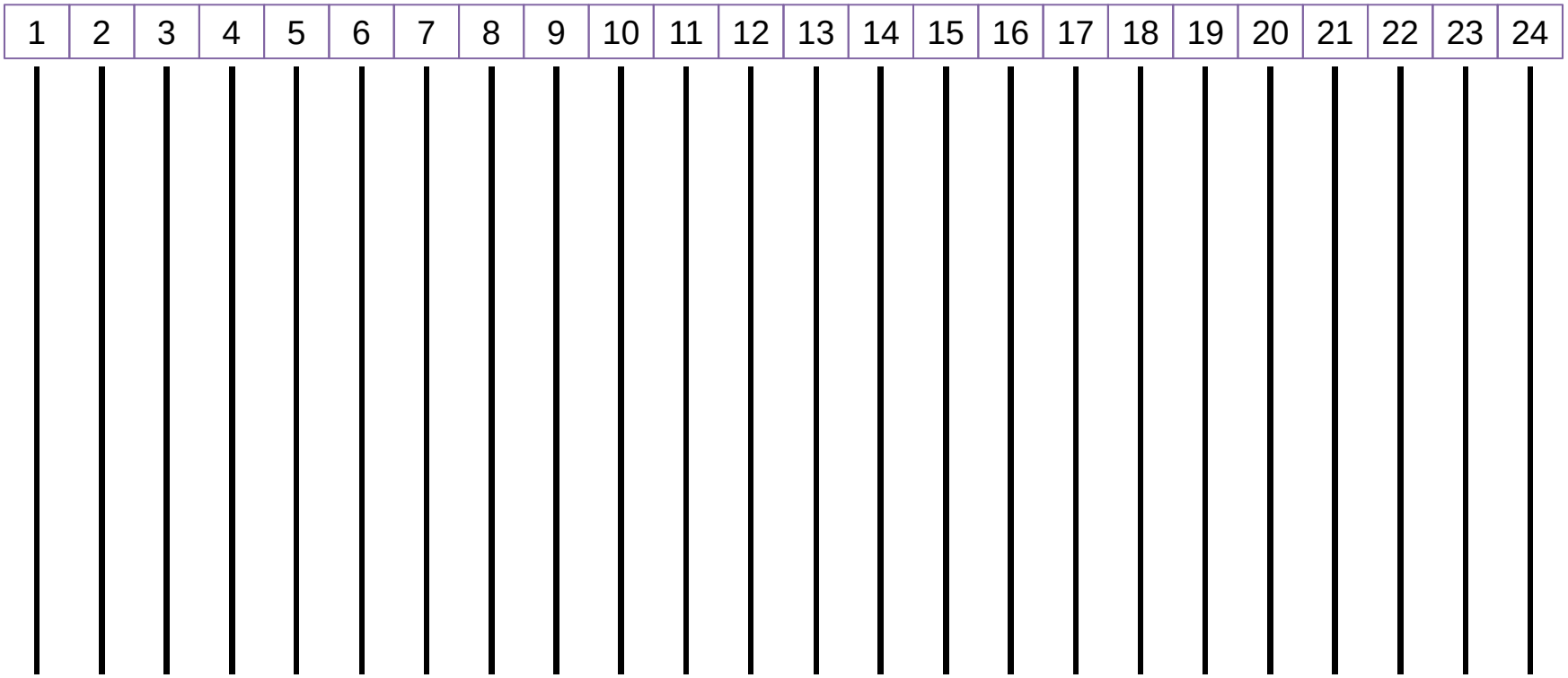
Shortest duration



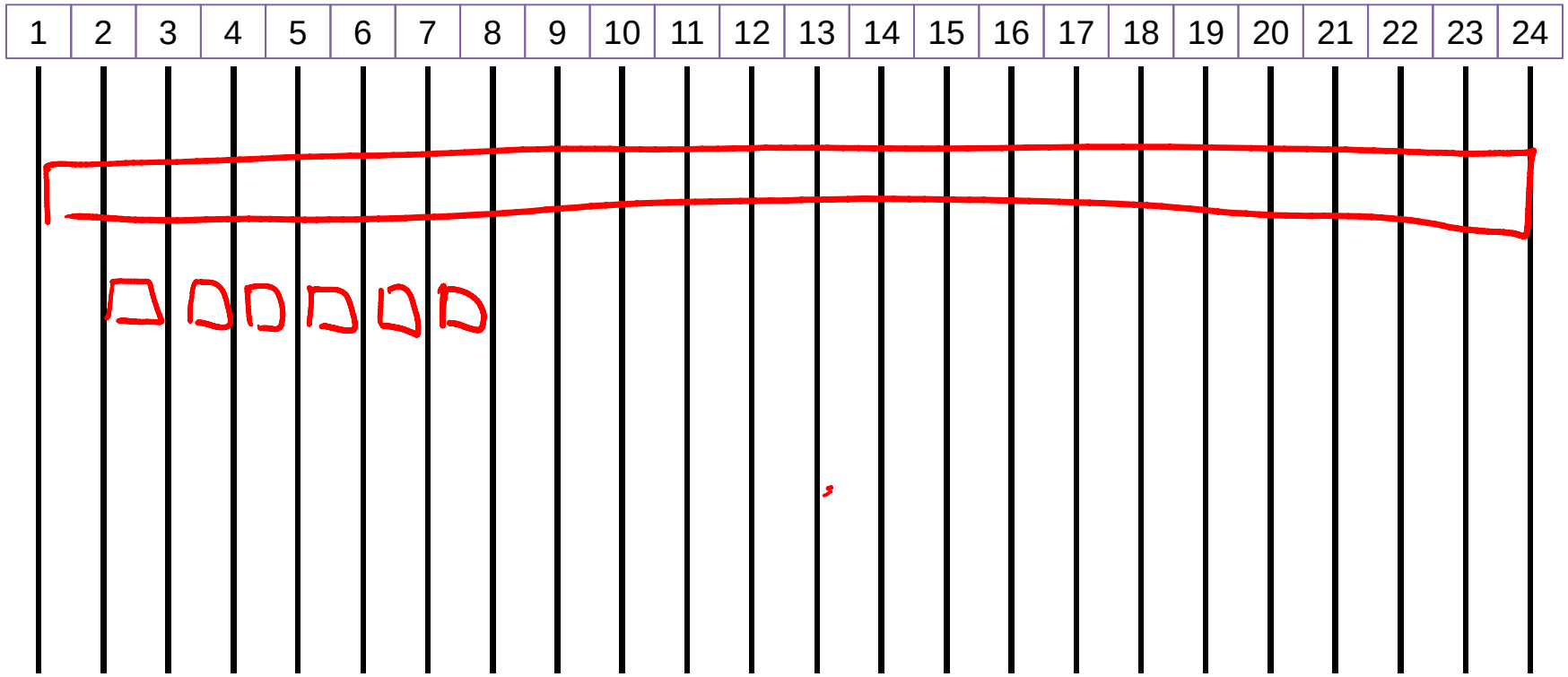
Shortest duration



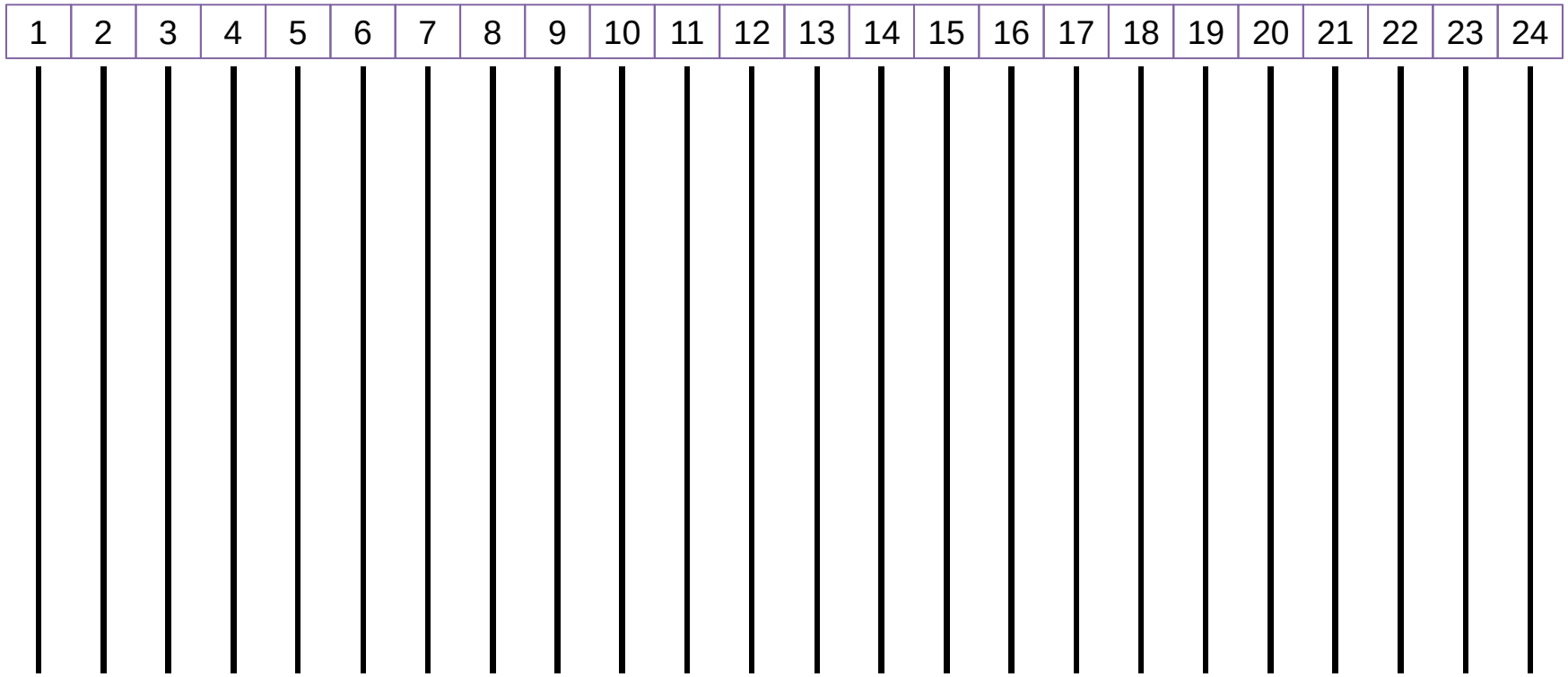
Earliest start time



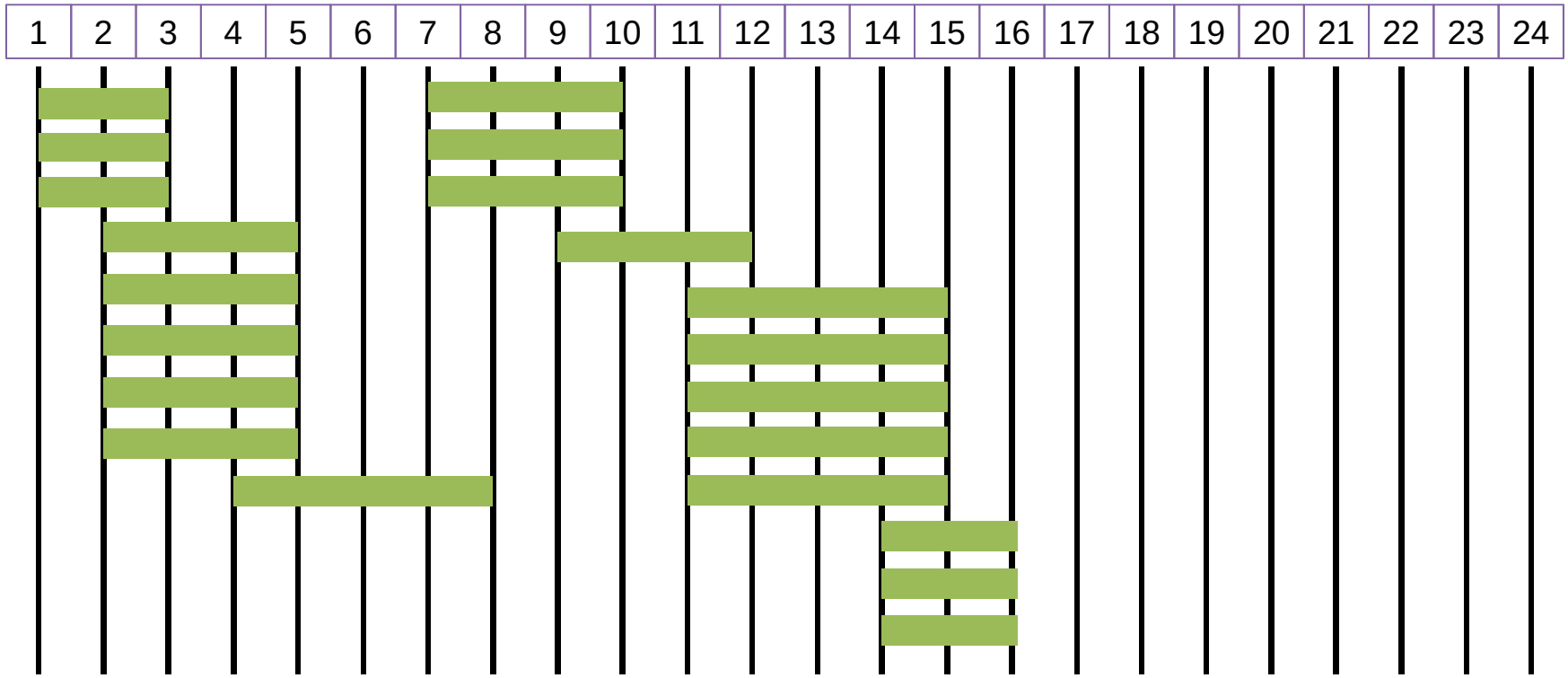
Earliest start time



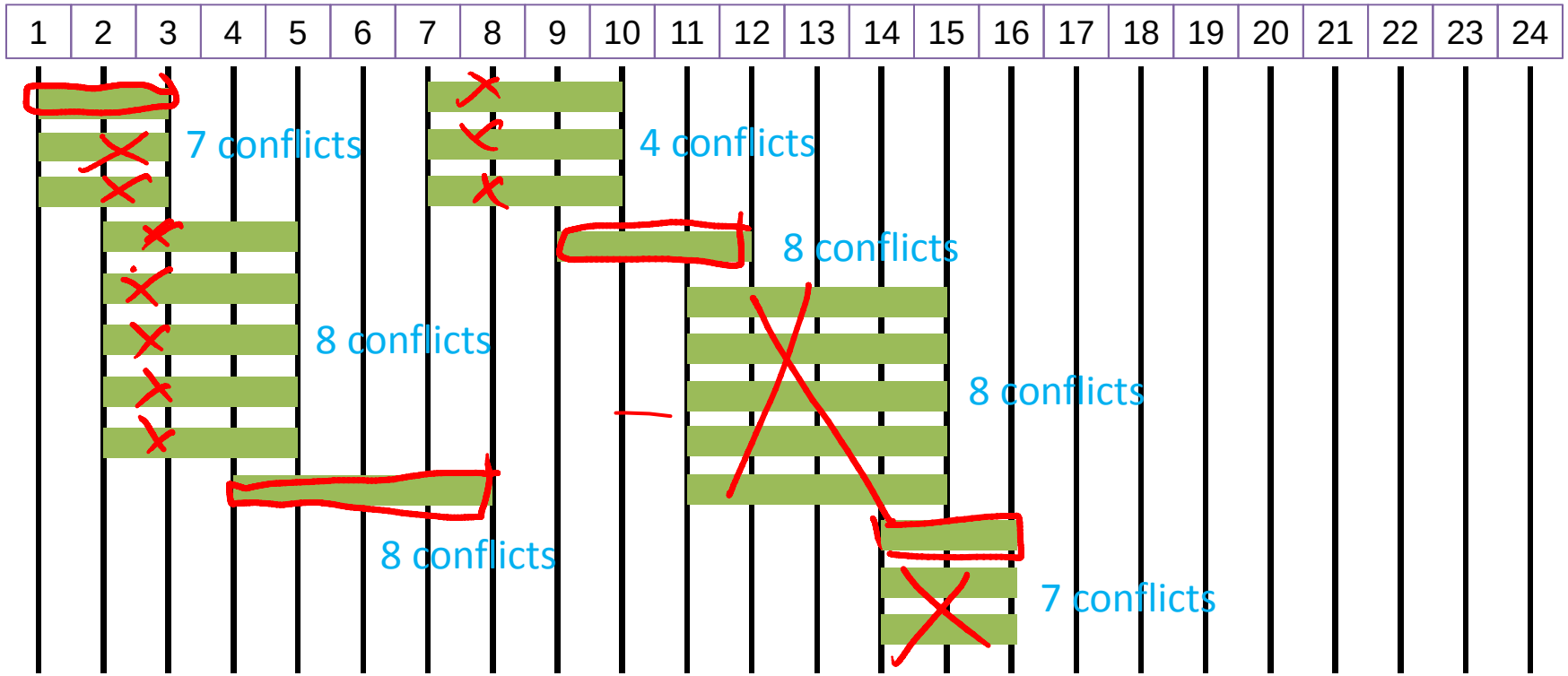
Fewest conflicts



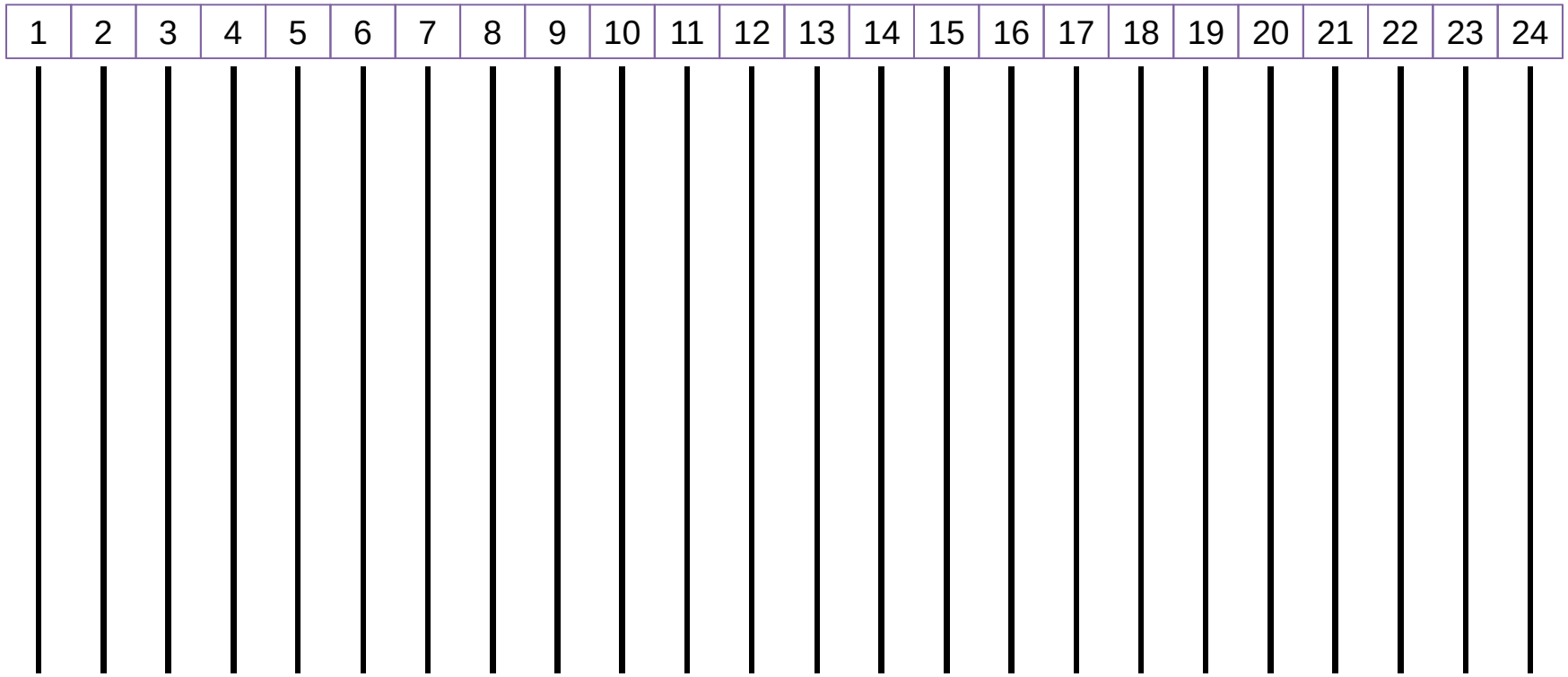
Counterexample for fewest conflicts



Counterexample for fewest conflicts



Earliest end time



Event scheduling

- Your goal is to schedule the most events possible that day such that no two events overlap
- Exponential is too slow. Let's try some greedy strategies:
  - ~~Shortest duration~~
  - ~~Earliest start time~~
  - ~~Fewest conflicts~~
  - Earliest end time (we cannot find a counterexample!)

Problem specification

- Design an algorithm that uses the greedy choice of picking the next available event with the earliest end time
  - Instance: n events E_1, …, E_n each with a start time s and end time f; $E_i = (s_i, f_i)$
  - Solution format: list of events
  - Constraints: events cannot overlap
  - Objective: maximize the number of events

Event scheduling implementation

Initialize a queue S
Sort the intervals by end time
Put the first event E_1 in S
Set $F = f_1$ *F* is finish time of last event put in *S*
For $i = 2 \ldots n$:
  If $s_i \geq F$: Else, $s_i < F$, E_i conflicts with last event put in *S*
    enqueue(E_i, S)
    $F = f_i$
Return S

Proving optimization algorithms correct

- Remember what we need to show
  - Let I be any instance of our problem, GS the greedy algorithm's solution, and OS any other solution
  - If minimization: Cost (OS) ≥ Cost(GS)
  - If maximization: Value(GS) ≥ Value (OS)

Proving optimality

- What does it mean that the greedy algorithm solves an optimization problem?
- I: problem instance
- GS: greedy solution to I
- OS: other solution to I
- For the greedy solution to be optimal, this means that

  Value(OS) $\leq$ Value (GS) for maximization (or Cost(GS) $\leq$ Cost (OS) for minimization)
- We need to show
  - For every instance I, let GS be the greedy algorithm's solution to I. Let OS be any other solution for I. Then,

    Value(OS) $\leq$Value (GS) (or Cost(GS) $\leq$ Cost (OS) for minimization)
- Tricky part: OS is arbitrary solution, not one that makes sense. We do not know much about it.

Techniques to prove optimality

- We will look at a number of general methods to prove optimality
  - Greedy modify the solution (also referred to as greedy exchange): most general
  - Greedy stays ahead: more intuitive
  - Greedy achieves the bound: also comes up in approximation, linear programming, network flow
- Which one to use is up to you, but modify the solution applies almost universally. Others can be easier, but only work in special cases.

Greedy modify the solution

- General structure of modify the solution
  1. Let g be the first greedy choice the algorithm makes
  2. Let OS be a solution achieved by not choosing g
  3. Show how to transform OS into some solution OS' that chooses g, and that is at least as good as OS
     - Must show that OS' is a valid solution and that OS' is better than or equal to OS
  4. Use 1-3 in an inductive argument
     - Must show that OS' is a valid solution and that OS' is better than or equal to OS
     - Base Case: the greedy strategy works for an instance of size 1
     - Assume the greedy strategy works for all instances of size $< n$
     - Let OS be any solution for instance $I, |I| = n$. Then, there is another solution OS' such that $|OS| \leq |OS'|$ and OS' includes the first greedy choice g

       $|OS| \leq |OS'| = |\{g\} \cup OS(I')| \leq |\{g\} \cup GS(I')| = |GS(I)|$

Earliest end time, greedy modify the solution

- Correctness:
  - Let $E = \{E_1, \ldots E_n\}$ be the set of all events with the start time s_i and finish time f_i of E_i
  - Greedy modify the solution: Say E_1 is the event with the earliest finish time (E_1 is the first greedy choice)
  - Let OS be a legal schedule that does not include E_1
  - **Claim**: there is a schedule OS' that does include E_1 such that $|OS'| \geq |OS|$
  - Proof: (greedy modify the solution)
  - Let the events in OS be $J_1, \ldots, J_k$, ordered by start and finish times ($J_1 \neq E_1$)
  - Define OS' from OS

$E_1(f) \leq J_1(f)$

Earliest end time, greedy modify the solution

- What we have to start
- *OS*

First greedy decision

Earlier finish time than J_1

- Our to do list

Define OS'

- OS' must have no overlaps
- OS' must include E_1
- $|OS'| \geq |OS|$

Define OS'

- *OS*

First greedy decision

OS' = OS ∪ $\{E_1\}$– {events in OS that conflict with E_1}

OS' is valid (there are no conflicts, by design)
What to show: $|OS'| \geq |OS|$

Define OS'

- *OS*

First greedy decision

$E_1(f) \leq J_1(f)$

OS' = $OS \cup \{E_1\} - \{J_1\}$, if J_1 intersects with E_1
OS' = $OS \cup E_1$, otherwise

OS' has no overlaps

- OS' = $OS \cup \{E_1\} - \{J_1\}$, if J_1 intersects with E_1
- OS' = $OS \cup E_1$, otherwise

Only new place for overlaps: E_1 with J_2, so we need to show $E_1(f) \leq J_2(s)$

OS' has no overlaps

- OS' = $OS \cup \{E_1\} - \{J_1\}$, if J_1 intersects with E_1
- OS' = $OS \cup E_1$, otherwise

$E_1(f) \leq J_1(f)$

Only new place for overlaps: E_1 with J_2, so we need to show $E_1(f) \leq J_2(s)$

$E_1(f) \leq J_1(f) \leq J_2(s)$

OS' is at least as good as OS

- OS' = $OS \cup \{E_1\} - \{J_1\}$, if J_1 intersects with E_1
- OS' = $OS \cup E_1$, otherwise

Either of the above
$|OS'| = |OS|+1-1=|OS|$, or
$|OS'| = |OS|+1 > |OS|$

$|OS'| \geq |OS|$

Induction

- There is an optimal solution that always picks the greedy choice
  - Proof by strong induction on n, the number of events
  - Base case: $n = 0$ or $n = 1$. The greedy (actually, any) choice works.
  - Inductive hypothesis (strong)
  - Assume that the greedy algorithm is optimal for any k events for $0 \leq k \leq n - 1$ (i.e., if $|I| < n$, then for any solution $OS(I)$, $|OS(I)| \leq |GS(I)|$)
  - Goal: greedy is optimal for any n events

Induction

- Inductive Step: Let OS be any solution of the set of events $I = \{E_1, \dots, E_n\}$. Then, there exists a solution OS' such that $|OS| \leq |OS'|$ and OS' includes the first greedy choice E_1.
- Let I' be the set of events that do not conflict with E_1
- Then, $OS' = \{E_1\} \cup S(I')$, where $S(I')$ is some solution of I'
- Since $|I'| < n$, by the inductive hypothesis, $|S(I')| \leq |GS(I')|$
- And, by definition, $GS(I) = \{E_1\} \cup GS(I')$
- Conclusion

  $$|OS| \leq |OS'| = |\{E_1\} \cup S(I')| \leq |\{E_1\} \cup GS(I')| = |GS(I)|$$

  – The greedy solution GS is optimal for every set of events

Induction

- Inductive Step: Let OS be any solution of the set of events $I = \{E_1, \ldots, E_n\}$. Then, there exists a solution OS' such that $|OS| \leq |OS'|$ and OS' includes the first greedy choice E_1.
- Let I' be the set of events that do not conflict with E_1
- Then, $OS' = \{E_1\} \cup S(I')$, where $S(I')$ is some solution of I'
- Since $|I'| < n$, by the inductive hypothesis, $|S(I')| \leq |GS(I')|$
- And, by definition, $GS(I) = \{E_1\} \cup GS(I')$
- Conclusion

  $$|OS| \leq |OS'| = |\{E_1\} \cup S(I')| \leq |\{E_1\} \cup GS(I')| = |GS(I)|$$

  – The greedy solution GS is optimal for every set of events

General greedy modify the solution template

- Lemma: Let g be the first greedy decision. Let OS be any legal solution that does not pick g. Then, there is a solution OS' that does pick g and OS' is at least as good as OS.

General greedy modify the solution template, proof of lemma

- Lemma: Let g be the first greedy decision. Let OS be any legal solution that does not pick g. Then, there is a solution OS' that does pick g and OS' is at least as good as OS.
  1. State what we know: Definition of g. OS meets constraints.
  2. Define OS' from OS, g This requires creativity
  3. Prove that OS' meets constraints (use 1, 2)
  4. Compare value/cost of OS' to OS (use 2, sometimes 1)

General greedy modify the solution template, induction

- Lemma: Let g be the first greedy decision. Let OS be any legal solution that does not pick g. Then, there is a solution OS' that does pick g and OS' is at least as good as OS.
- Prove by strong induction on instance size that *GS* is optimal
- Induction step
  1. Let g be first greedy decision. Let I' be "rest of problem given g''
  2. GS = g + GS(I')
  3. OS is any legal solution
  4. OS' is defined from OS by the modify the solution argument (if OS does not include g)
  5. OS' = g + some solution on I'
  6. Induction: GS(I') at least as good as some solution on I'
  7. GS is at least as good as OS', which is at least as good as OS

Event scheduling implementation

Initialize a queue S
Sort the intervals by end time
Put the first event E_1 in S
Set $F = f_1$
For $i = 2 \ldots n$:
  If $s_i \geq F$:
    enqueue(E_i, S)
    $F = f_i$
Return S

Another method:
greedy achieves the bound

Greedy achieves the bound

- This is a proof technique that does not work in all cases
- The way it works is to argue that when the greedy solution reaches its peak cost, it reveals a bound
- Then, show this bound is also a lower bound on the cost of any other solution
- So we are showing: Cost(GS) ≤ Bound ≤ Cost (OS)
- Allows the two inequalities to be separated

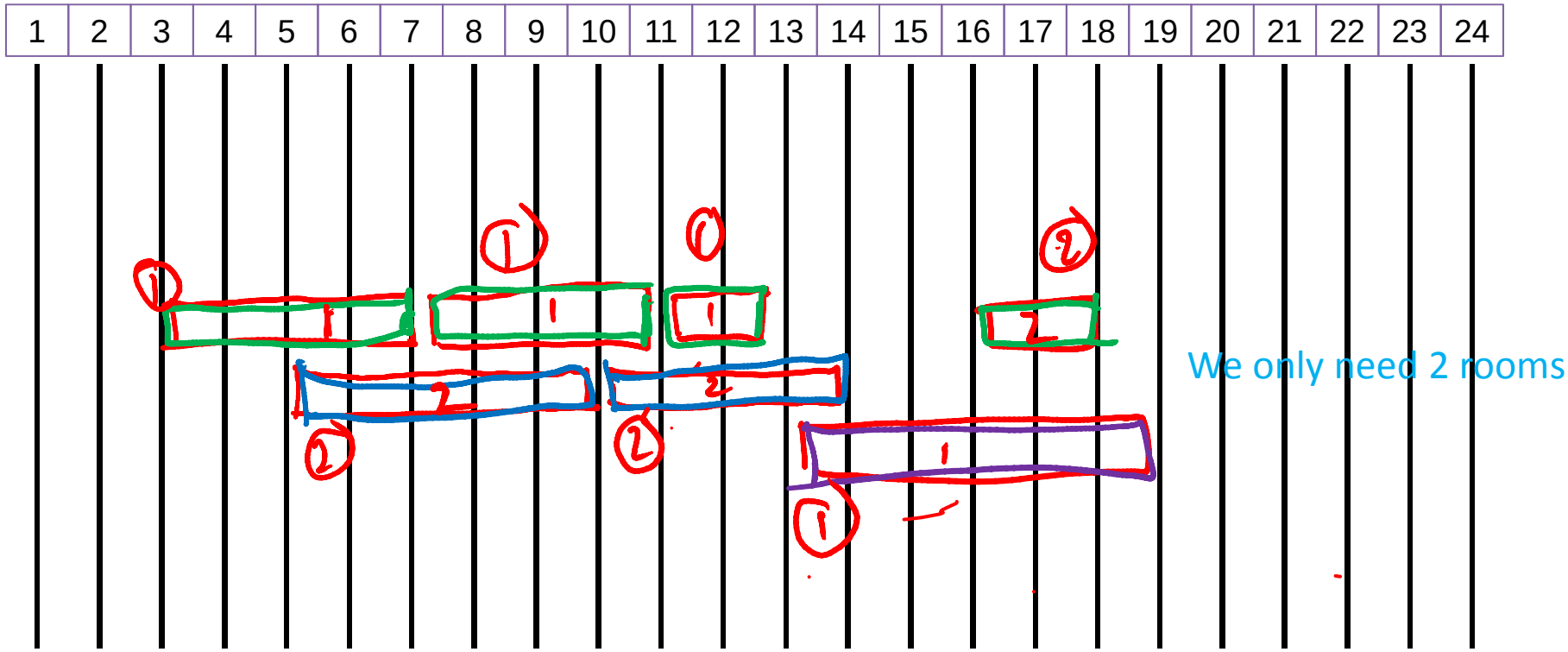
Event scheduling with multiple rooms

- Suppose you have a conference to plan with *n* events and you have an unlimited supply of rooms. How can you assign events to rooms in such a way as to minimize the number of rooms?
- Greedy choice:

Event scheduling with multiple rooms

- Suppose you have a conference to plan with *n* events and you have an unlimited supply of rooms. How can you assign events to rooms in such a way as to minimize the number of rooms?
- Greedy choice:
  - Run previous algorithm and put results in room 1
  - Repeat on remainder and put results in room 2
  - Repeat until all events have a room

Counterexample



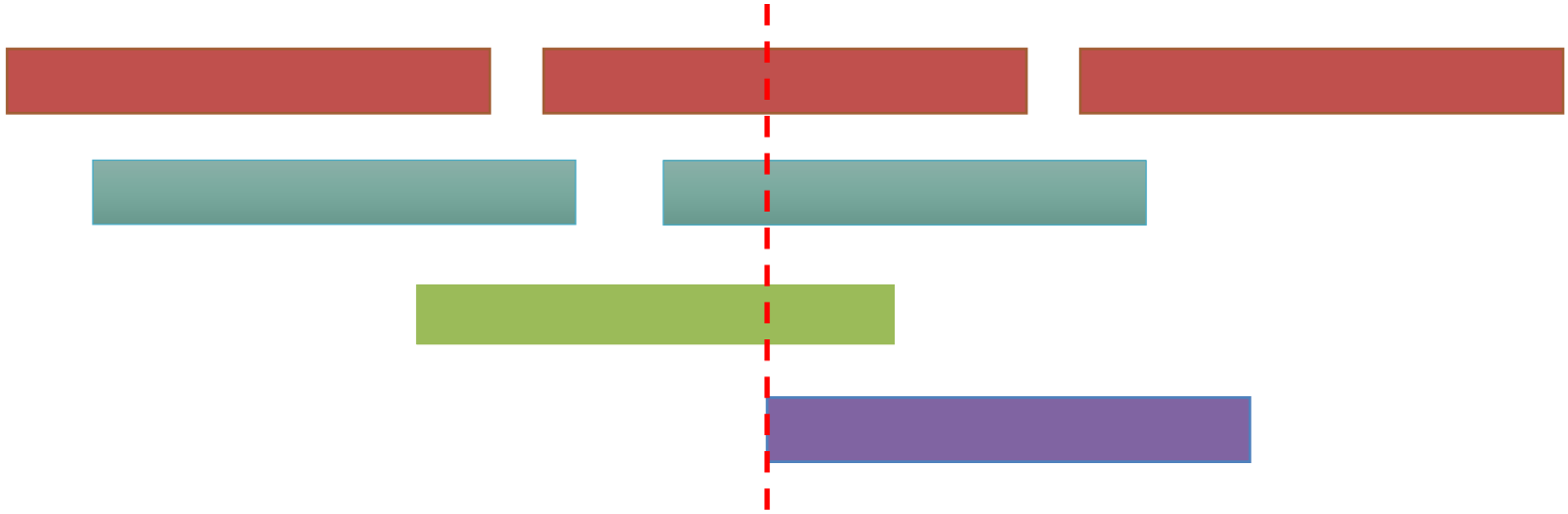
Event scheduling with multiple rooms

- Suppose you have a conference to plan with *n* events and you have an unlimited supply of rooms. How can you assign events to rooms in such a way as to minimize the number of rooms?
- Greedy choice:
  - Number the rooms from 1 to *n*
  - Sort the events by earliest start time
  - Put the first event in room 1
  - For events 2, …, *n*, put each event in the smallest numbered room that is available

When does GS reach peak cost?

Room

- 1
- 2
- 3
- 4



Why did we have to use four rooms? What happened at the time we reached "peak cost"?

Defining the lower bound

- Let t be a certain time during the conference
- Let $B(t)$ be the set of all events E such that $t \in E$
- Let R be the number of rooms you need for a valid schedule
- Then, $R \geq |B(t)|$ for all t
- Proof
  - For any time t, let $E_{i_1}, \ldots, E_{i_{|B(t)|}}$ be all the events in $B(t)$
  - Then, since they all are happening at time t, they all have to be in different rooms, so $R \geq |B(t)|$
- Let $L = \max_t |B(t)|$. Then, L is the lower bound on the number of rooms needed.

Greedy achieves the bound

- Let k be the number of rooms picked by the greedy algorithm. Then, at some point t, $|B(t)| \geq k$ (i.e., there are at least k events happening at time t).
- Proof
  - Let t be the starting time of the first event to be scheduled in room k
  - Then, by the greedy choice, room k was the least number room available at that time
  - This means at time t there was an event happening in room 1, room 2, …, room k-1. And, an additional event happening in room k
  - Therefore, $|B(t)| \geq k$ at some point t

Conclusion: greedy is optimal

- The greedy algorithm uses the minimum number of rooms
  - Let GS be the greedy solution, *k* = Cost(GS) the number of rooms used in the greedy solution
  - Let *k* be the number of rooms the greedy algorithm uses and let *R* be any valid schedule of rooms. There exists a *t* such that at all time, *k* events are happening simultaneously. So *R* uses at least *k* rooms. So, *R* uses at least as many rooms as the greedy solution. Therefore, the greedy solution is optimal.

Conclusion: greedy is optimal

- Let GS be the greedy solution, k = Cost(GS) the number of rooms used in the greedy solution
- Let OS be any other schedule, R= Cost(OS) the number of rooms used in OS
- By the bounding lemma, $\mathrm{R} \geq L = \max_t |B(t)|$
- By the achieves the bound lemma, k = |B(t)| ≤ L for some t
- Putting the two together, Cost(GS) =k ≤ R= Cost(OS)
- Thus, the greedy solution is optimal

Next lecture

- Greedy algorithms
  - Reading: Kleinberg and Tardos, sections 4.1, 4.2, and 4.3