

## Objectives

- Greedy Algorithms
  - Interval partitioning
  - Minimizing Lateness
- Exchange argument

Feb 4, 2013

CSCI211 - Sprenkle

1

## Review

- What is the template for a greedy solution?
- What problem did we solve optimally with a greedy algorithm?
- How did we prove optimality?

Feb 4, 2013

CSCI211 - Sprenkle

2

## Review: Greedy Algorithms

- Template
  1. Consider jobs (or whatever) in some order
    - Decision: What order is best?
  2. Take each job provided it's compatible with the ones already taken
- At each step, take as much as you can get
  - Feasible – satisfy problem's constraints
  - Locally optimal – best local choice among available feasible choices
  - Irrevocable – after decided, no going back

Feb 4, 2013

CSCI211 - Sprenkle

3

## Review: Greedy Stays Ahead Proofs

1. Define your solutions
  - Describe the form of your greedy solution (A) and of some other solution (possibly the optimal solution, O)
2. Find a measure
  - Find a measure by which greedy *stays ahead* of the optimal solution
    - Ex: Let  $a_1, \dots, a_k$  be the first  $k$  measures of greedy algorithm and  $o_1, \dots, o_m$  be the first  $m$  measures of other solution (sometimes  $m = k$ )
3. Prove greedy stays ahead
  - Show that greedy's partial solutions constructed are always just as good as the optimal solution's initial segments based on the measure
    - Ex: for all indices  $r \leq \min(k, m)$ , prove by induction that  $a_r \geq o_r$  or  $a_r \leq o_r$
  - Use the greedy algorithm to help you argue the inductive step
4. Prove optimality
  - Prove that since greedy stays ahead of the other solution with respect to the measure, then the greedy solution is optimal

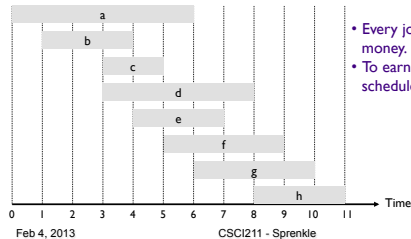
Feb 4, 2013

CSCI211 - Sprenkle

4

## Review: Interval Scheduling

- Job  $j$  starts at  $s_j$  and finishes at  $f_j$
- Two jobs are **compatible** if they don't overlap
- **Goal**: find maximum subset of mutually compatible jobs



- Every job is worth equal money.
- To earn the most money → schedule the most jobs

Feb 4, 2013

CSCI211 - Sprenkle

5

## Problem Assumptions

- All requests were known to scheduling algorithm
  - Online algorithms: make decisions without knowledge of future input
- Each job was worth the same amount
  - What if jobs had *different* values?
    - E.g., scaled with size
- Single resource requested
  - Rejected requests that didn't fit

Feb 4, 2013

CSCI211 - Sprenkle

6

## INTERVAL PARTITIONING

Feb 4, 2013

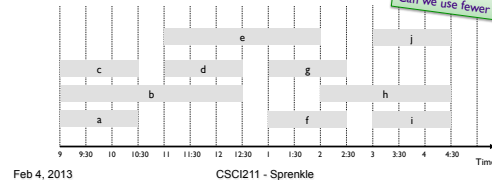
CSCI211 - Sprenkle

7

## Interval Partitioning

- Lecture  $j$  starts at  $s_j$  and finishes at  $f_j$
- Goal:** find minimum number of classrooms to schedule all lectures so that no two occur at the same time in the same room.
- Ex:** 10 lectures in 4 classrooms

What are our constraints?  
Can we use fewer rooms?



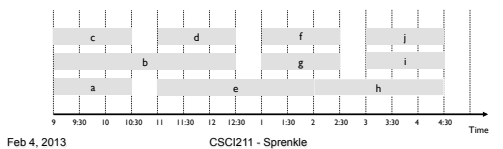
Feb 4, 2013

CSCI211 - Sprenkle

8

## Interval Partitioning

- Lecture  $j$  starts at  $s_j$  and finishes at  $f_j$
- Goal:** find minimum number of classrooms to schedule all lectures so that no two occur at the same time in the same room.
- Alternative schedule uses only 3 classrooms



Feb 4, 2013

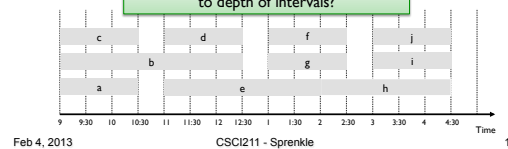
CSCI211 - Sprenkle

9

## Interval Partitioning: Lower Bound on Optimal Solution

- Def.** The depth of a set of open intervals is the maximum number that contain any given time.
- Key observation.** # of classrooms needed  $\geq$  depth.
- Ex:** Depth of schedule below = 3  $\Rightarrow$  schedule below is optimal.

Does there always exist a schedule equal to depth of intervals?



Feb 4, 2013

CSCI211 - Sprenkle

10

## Interval Partitioning Discussion

- Does there always exist a schedule equal to depth of intervals?
- Can we make decisions locally to get a global optimum?
  - Or are there long-range obstacles that require more resources?

Feb 4, 2013

CSCI211 - Sprenkle

11

## Interval Partitioning: Greedy Algorithm

- Consider lectures in increasing order of start time: assign lecture to any compatible classroom

```
Sort intervals by starting time so that  $s_1 \leq s_2 \leq \dots \leq s_n$ 
d = 0 ← number of allocated classrooms
for j = 1 to n
  if lecture j is compatible with some classroom k
    schedule lecture j in classroom k
  else
    allocate a new classroom d + 1
    schedule lecture j in classroom d + 1
    d = d + 1
```

Analyze algorithm

Feb 4, 2013

CSCI211 - Sprenkle

12

## Interval Partitioning: Greedy Algorithm

- Consider lectures in increasing order of start time: assign lecture to any compatible classroom

```

Sort intervals by starting time so that  $s_1 \leq s_2 \leq \dots \leq s_n$ 
 $d = 0$  ← number of allocated classrooms
for  $j = 1$  to  $n$ 
  if (lecture  $j$  is compatible with some classroom  $k$ )
    schedule lecture  $j$  in classroom  $k$ 
  else
    allocate a new classroom  $d + 1$ 
    schedule lecture  $j$  in classroom  $d + 1$ 
     $d = d + 1$ 

```

- Implementation:  $O(n \log n)$ 
  - For each classroom  $k$ , maintain the finish time of the last job added.
  - Keep the classrooms in a priority queue by last job finish time.

Feb 4, 2013

CSCI211 - Sprenkle

13

## Interval Partitioning: Greedy Analysis

- Observation.** Greedy algorithm never schedules two incompatible lectures in the same classroom
- Theorem.** Greedy algorithm is optimal
- Pf Intuition**
  - When do we add more classrooms?
  - When would we add the  $d+1$  classroom?

Feb 4, 2013

CSCI211 - Sprenkle

14

## Interval Partitioning: Greedy Analysis

- Observation.** Greedy algorithm never schedules two incompatible lectures in the same classroom
- Theorem.** Greedy algorithm is optimal
- Pf.**
  - Let  $d$  = number of classrooms that the greedy algorithm allocates
  - Classroom  $d$  is opened because we needed to schedule a job, say  $j$ , that is incompatible with all  $d-1$  other classrooms
  - Since we sorted by start time, all these incompatibilities are caused by lectures that start no later than  $s_j$
  - Thus, we have  $d$  lectures overlapping at time  $s_j + \epsilon$
  - $d$  is the depth of the set of lectures

Feb 4, 2013

CSCI211 - Sprenkle

15

Exchange argument

## SCHEDULING TO MINIMIZE MAX LATENESS

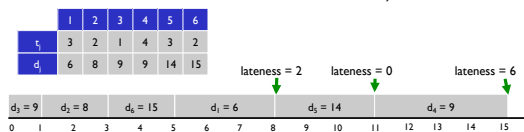
Feb 4, 2013

CSCI211 - Sprenkle

16

## Scheduling to Minimizing Max Lateness

- Single resource processes one job at a time
- Job  $j$  requires  $t_j$  units of processing time and is due at time  $d_j$  (its deadline)
- If  $j$  starts at time  $s_j$ , it finishes at time  $f_j = s_j + t_j$
- Lateness:**  $\ell_j = \max \{ 0, f_j - d_j \}$
- Goal:** schedule all jobs to **minimize maximum lateness**  $L = \max \ell_j$



Feb 4, 2013

CSCI211 - Sp

17

## Greedy Algorithms

- Greedy template.** Consider jobs in some order.
- What do we want to optimize?
- What order?
  - Intuition of order?
  - Counter examples for order being optimal?

Feb 4, 2013

CSCI211 - Sprenkle

18

## Minimizing Lateness: Greedy Algorithms

- **Greedy template.** Consider jobs in some order.
  - Shortest processing time first. Consider jobs in ascending order of processing time  $t_j$ .

Counter example

	1	2
$t_j$	1	10
$d_j$	100	10

- Smallest slack. Consider jobs in ascending order of slack  $d_j - t_j$ .

Counter example

	1	2
$t_j$	1	10
$d_j$	2	10

Feb 4, 2013

CSCI211 - Sprenkle

19

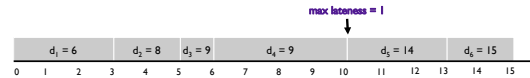
## Minimizing Lateness: Greedy Algorithm

- **Earliest deadline first.**

```

Sort n jobs by deadline so that  $d_1 \leq d_2 \leq \dots \leq d_n$ 
 $t = 0$ 
for  $j = 1$  to  $n$ 
  Assign job  $j$  to interval  $[t, t + t_j]$ 
   $s_j = t$ 
   $f_j = t + t_j$ 
   $t = t + t_j$ 
output intervals  $[s_j, f_j]$ 

```



What can we say about this algorithm/its results?

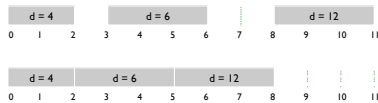
Feb 4, 2013

CSCI211 - Sprenkle

20

## Minimizing Lateness: No Idle Time

- **Observation.** There exists an optimal schedule with no **idle time**



- **Observation.** The greedy schedule has no idle time

Feb 4, 2013

CSCI211 - Sprenkle

21

## Proving Optimality

- **Goal:** Prove greedy algorithm produces optimal solution
- **Approach: Exchange argument**
  - Start with an optimal schedule Opt
  - Gradually modify Opt, preserving its optimality
  - Transform into a schedule identical to greedy's schedule

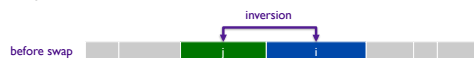
Feb 4, 2013

CSCI211 - Sprenkle

22

## Minimizing Lateness: Inversions

- **Def.** An **inversion** in schedule  $S$  is a pair of jobs  $i$  and  $j$  such that:  
 $d_i < d_j$  ( $i$ 's deadline is before  $j$ )  
 but  $j$  scheduled before  $i$



Can Greedy's solution have any inversions?

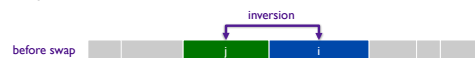
Feb 4, 2013

CSCI211 - Sprenkle

23

## Minimizing Lateness: Inversions

- **Def.** An **inversion** in schedule  $S$  is a pair of jobs  $i$  and  $j$  such that:  
 $d_i < d_j$  ( $i$ 's deadline is before  $j$ )  
 but  $j$  scheduled before  $i$



Greedy's schedule has no inversions!

Feb 4, 2013

CSCI211 - Sprenkle

24

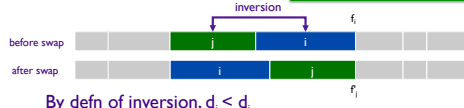
## Minimizing Lateness: Inversions

- **Claim.** Swapping two adjacent, inverted jobs reduces the number of inversions by one and does *not increase the max lateness*

How do we know inversions are adjacent?

- **Pf Setup.** Let  $\ell$  be the lateness before the swap, and let  $\ell'$  be it afterwards

What can we say about how  $i$ 's,  $j$ 's, and other jobs' lateness changes?



By defn of inversion,  $d_i < d_j$

Feb 4, 2013

CSCI211 - Sprenkle

25

## Minimizing Lateness: Inversions

- **Claim.** Swapping two adjacent jobs with the same deadline does not increase the max lateness

- **Pf.** Let  $\ell$  be the lateness before the swap, and let  $\ell'$  be it afterwards

➤ Lateness remains the same for all other jobs:

- $\ell'_k = \ell_k$  for all  $k \neq i, j$

➤ Lateness of  $i$  before is  $\ell_i = f_i - d_i = T_{i-1} + t_i + t_j - d_i$

➤ Lateness of  $j$  after is  $\ell'_j = f'_j - d_j = T_{i-1} + t_i + t_j - d_j$

- But  $d_i < d_j$

Put in terms of  $\ell_i$



Feb 4, 2013

CSCI211 - Sprenkle

26

## Minimizing Lateness: Inversions

- **Claim.** Swapping two adjacent, inverted jobs reduces the number of inversions by one and does *not increase the max lateness*.

- **Pf.** Let  $\ell$  be the lateness before the swap, and let  $\ell'$  be it afterwards

➤  $\ell'_k = \ell_k$  for all  $k \neq i, j$

➤  $\ell'_i \leq \ell_i$

➤ If job  $j$  is late:

$$\begin{aligned} \ell'_j &= f'_j - d_j && \text{(definition)} \\ &= f_i - d_j && \text{(j finishes at time } f_i\text{)} \\ &\leq f_i - d_i && (i < j) \\ &\leq \ell_i && \text{(definition)} \end{aligned}$$

Feb 4, 2013

CSCI211 - Sprenkle

27

## Greedy Exchange Proofs

1. Label your algorithm's solution and a general solution.
  - Example: let  $A = \{a_1, a_2, \dots, a_n\}$  be the solution generated by your algorithm, and let  $O = \{o_1, o_2, \dots, o_n\}$  be an optimal feasible solution.
2. Compare greedy with other solution.
  - Assume that the arbitrary/optimal solution is not the same as your greedy solution (since otherwise, you are done).
  - Typically, can isolate a simple example of this difference, such as:
    - ① There is an element  $e \in O$  that  $\notin A$  and an element  $f \in A$  that  $\notin O$
    - ② 2 consecutive elements in  $O$  are in a different order than in  $A$
 ➤ i.e., there is an *inversion*
3. Exchange.
  - Swap the elements in question in  $O$  (either ① swap one element out and another in or ② swap the order of the elements) and argue that solution is no worse than before.
  - Argue that if you continue swapping, you eliminate all differences between  $O$  and  $A$  in a *finite* # of steps without worsening the solution's quality.
  - Thus, the greedy solution produced is just as good as any optimal solution, and hence is optimal itself.

Feb 4, 2013

CSCI211 - Sprenkle

28

## Minimizing Lateness: Analysis of Greedy Algorithm

- **Theorem.** Greedy schedule  $S$  is optimal
- **Pf idea.** Convert Opt to Greedy
  - Does opt schedule have idle time?
  - What if opt schedule has no inversions?
  - What if opt schedule has inversions?

Feb 4, 2013

CSCI211 - Sprenkle

29

## Minimizing Lateness: Analysis of Greedy Algorithm

- **Theorem.** Greedy schedule  $S$  is optimal
- **Pf.** Define  $S^*$  to be an optimal schedule that has the fewest number of inversions, and let's see what happens
  - Can assume  $S^*$  has no idle time
  - If  $S^*$  has no inversions, then  $S = S^*$
  - If  $S^*$  has an inversion, let  $i-j$  be an adjacent inversion
    - Swapping  $i$  and  $j$  does not increase the maximum lateness and strictly decreases the number of inversions
    - This contradicts definition of  $S^*$

Feb 4, 2013

CSCI211 - Sprenkle

30

## Greedy Analysis Strategies

- **Greedy algorithm stays ahead.** Show that after each step of the greedy algorithm, its solution is at least as good as any other algorithm's.
- **Exchange argument.** Gradually transform any solution to the one found by the greedy algorithm without hurting its quality.
- **Structural.** Discover a simple "structural" bound asserting that every possible solution must have a certain value. Then show that your algorithm always achieves this bound.

Feb 4, 2013

CSCI211 - Sprenkle

31

## Assignments

- Exam 1 – due Friday
  - Open book, open notes, open lecture notes
  - I mention explicitly to analyze your algorithms' running times. I will not do that in the future.
- Wed: work period
  - Ask me questions
  - Office Hours: today: 1-3 p.m., Wed: 1-4:30 p.m.
    - By appointment

Feb 4, 2013

CSCI211 - Sprenkle

32