CSE 421
Algorithms
Richard Anderson
Autumn 2019
Lecture 8 – Greedy Algorithms II

Announcements
• Today’s lecture
  – Kleinberg-Tardos, 4.2, 4.3
• Next week
  – Kleinberg-Tardos, 4.4, 4.5

Scheduling Intervals
• Given a set of intervals
  – What is the largest set of non-overlapping intervals
  – What is the minimum number of processors required to schedule all intervals
• Suppose the n intervals are “random”
  – What is the expected number of independent intervals
  – What is the expected depth

Generating a random set of intervals
• Method 1:
  – Each interval assigned random start position in [0.0, 1.0]
  – Each interval assigned a random length in [0.0, 1.0]
• Method 2:
  – Start with the array [1, 1, 2, 2, 3, 3, 4, 4, 5, 5]
  – Randomly permute it [2, 1, 4, 2, 3, 4, 5, 1, 3, 5]
  – Index of the first j is the start of interval j, and the index of the second j is the end of interval j

Greedy Algorithms
• Solve problems with the simplest possible algorithm
• The hard part: showing that something simple actually works
• Today’s problems (Sections 4.2, 4.3)
  – Homework Scheduling
  – Optimal Caching
  – Subsequence testing

Homework Scheduling
• Tasks to perform
• Deadlines on the tasks
• Freedom to schedule tasks in any order
• Can I get all my work turned in on time?
• If I can’t get everything in, I want to minimize the maximum lateness
Scheduling tasks

- Each task has a length $t_i$ and a deadline $d_i$
- All tasks are available at the start
- One task may be worked on at a time
- All tasks must be completed

- Goal: minimize maximum lateness
  - Lateness: $L_i = f_i - d_i$ if $f_i \geq d_i$

Example

<table>
<thead>
<tr>
<th>Time</th>
<th>Deadline</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_1$</td>
<td>2</td>
</tr>
<tr>
<td>$a_2$</td>
<td>3</td>
</tr>
<tr>
<td>$a_3$</td>
<td>4</td>
</tr>
</tbody>
</table>

Lateness 1

<table>
<thead>
<tr>
<th>Time</th>
<th>Deadline</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_3$</td>
<td>3</td>
</tr>
<tr>
<td>$a_2$</td>
<td>2</td>
</tr>
</tbody>
</table>

Lateness 3

Determine the minimum lateness

<table>
<thead>
<tr>
<th>Time</th>
<th>Deadline</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_1$</td>
<td>2</td>
</tr>
<tr>
<td>$a_2$</td>
<td>3</td>
</tr>
<tr>
<td>$a_3$</td>
<td>4</td>
</tr>
<tr>
<td>$a_4$</td>
<td>5</td>
</tr>
<tr>
<td>$a_5$</td>
<td>6</td>
</tr>
</tbody>
</table>

Greedy Algorithm

- Earliest deadline first
- Order jobs by deadline
- This algorithm is optimal

Analysis

- Suppose the jobs are ordered by deadlines, $d_1 \leq d_2 \leq \ldots \leq d_n$
- A schedule has an inversion if job $j$ is scheduled before $i$ where $j > i$
- The schedule $A$ computed by the greedy algorithm has no inversions.
- Let $O$ be the optimal schedule, we want to show that $A$ has the same maximum lateness as $O$

List the inversions

<table>
<thead>
<tr>
<th>Time</th>
<th>Deadline</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_1$</td>
<td>3</td>
</tr>
<tr>
<td>$a_2$</td>
<td>4</td>
</tr>
<tr>
<td>$a_3$</td>
<td>2</td>
</tr>
<tr>
<td>$a_4$</td>
<td>5</td>
</tr>
<tr>
<td>$a_5$</td>
<td>6</td>
</tr>
</tbody>
</table>

| $a_1$ | $a_2$ | $a_3$ | $a_4$ | $a_5$ |
Lemma: There is an optimal schedule with no idle time

- It doesn’t hurt to start your homework early!

- Note on proof techniques
  - This type of can be important for keeping proofs clean
  - It allows us to make a simplifying assumption for the remainder of the proof

Interchange argument

- Suppose there is a pair of jobs i and j, with \( d_i \leq d_j \) and j scheduled immediately before i. Interchanging i and j does not increase the maximum lateness.

Proof by Bubble Sort

Determine maximum lateness

Real Proof

- There is an optimal schedule with no inversions and no idle time.
- Let O be an optimal schedule k inversions, we construct a new optimal schedule with k-1 inversions
- Repeat until we have an optimal schedule with 0 inversions
- This is the solution found by the earliest deadline first algorithm

Result

- Earliest Deadline First algorithm constructs a schedule that minimizes the maximum lateness
Homework Scheduling

• How is the model unrealistic?

Extensions

• What if the objective is to minimize the sum of the lateness?
  – EDF does not work
• If the tasks have release times and deadlines, and are non-preemptable, the problem is NP-complete
• What about the case with release times and deadlines where tasks are preemptable?

Optimal Caching

• Caching problem:
  – Maintain collection of items in local memory
  – Minimize number of items fetched

Caching example

A, B, C, D, A, E, B, A, D, A, C, B, D, A

Optimal Caching

• If you know the sequence of requests, what is the optimal replacement pattern?
• Note – it is rare to know what the requests are in advance – but we still might want to do this:
  – Some specific applications, the sequence is known
  • Register allocation in code generation
  – Competitive analysis, compare performance on an online algorithm with an optimal offline algorithm

Farthest in the future algorithm

• Discard element used farthest in the future
A, B, C, A, D, C, B, C, A, D
Correctness Proof

- Sketch
- Start with Optimal Solution $O$
- Convert to Farthest in the Future Solution $F$-
- Look at the first place where they differ
- Convert $O$ to evict $F$-
  - There are some technicalities here to ensure the caches have the same configuration . . .

Subsequence Testing

- Is $a_1a_2...a_m$ a subsequence of $b_1b_2...b_n$?
  - e.g. is $T,R,E,E$ a subsequence of $S,T,U,A,R,T,R,E,G,E,S$

Next week

- 25

- 26

- 27