CSE 421
Algorithms
Richard Anderson
Winter 2019
Lecture 7

Announcements

• Reading
  – For today, sections 4.1, 4.2, 4.4
  – For friday, sections 4.5, 4.7, 4.8
• Homework 3 is available
  – Random Interval Graphs

Highlight from last lecture: Topological Sort Algorithm

While there exists a vertex v with in-degree 0
  Output vertex v
  Delete the vertex v and all outgoing edges

Greedy Algorithms

• Solve problems with the simplest possible algorithm
• The hard part: showing that something simple actually works
• Pseudo-definition
  – An algorithm is Greedy if it builds its solution by adding elements one at a time using a simple rule

Scheduling Theory

• Tasks
  – Processing requirements, release times, deadlines
• Processors
• Precedence constraints
• Objective function
  – Jobs scheduled, lateness, total execution time
Interval Scheduling

- Tasks occur at fixed times
- Single processor
- Maximize number of tasks completed
- Tasks \( \{1, 2, \ldots, N\} \)
- Start and finish times, \( s(i), f(i) \)

What is the largest solution?

Greedy Algorithm for Scheduling

Let \( T \) be the set of tasks, construct a set of independent tasks \( I \), \( A \) is the rule determining the greedy algorithm

\[
I = \{\}
\]
While \( T \) is not empty

- Select a task \( t \) from \( T \) by a rule \( A \)
- Add \( t \) to \( I \)
- Remove \( t \) and all tasks incompatible with \( t \) from \( T \)

Simulate the greedy algorithm for each of these heuristics

- Schedule earliest starting task
- Schedule shortest available task
- Schedule task with fewest conflicting tasks

Greedy solution based on earliest finishing time

Example 1

Example 2

Example 3

Theorem: Earliest Finish Algorithm is Optimal

- Key idea: Earliest Finish Algorithm stays ahead
- Let \( A = \{i_1, \ldots, i_k\} \) be the set of tasks found by EFA in increasing order of finish times
- Let \( B = \{j_1, \ldots, j_m\} \) be the set of tasks found by a different algorithm in increasing order of finish times
- Show that for \( r \leq \min(k, m) \), \( f(i_r) \leq f(j_r) \)
Stay ahead lemma

• A always stays ahead of B, \( f(i) \leq f(j) \)
• Induction argument
  - \( f(i_1) \leq f(j_1) \)
  - If \( f(i_{r-1}) \leq f(j_{r-1}) \) then \( f(i_r) \leq f(j_r) \)

Completing the proof

• Let \( A = \{i_1, \ldots, i_k\} \) be the set of tasks found by EFA in increasing order of finish times
• Let \( O = \{j_1, \ldots, j_m\} \) be the set of tasks found by an optimal algorithm in increasing order of finish times
• If \( k < m \), then the Earliest Finish Algorithm stopped before it ran out of tasks

Scheduling all intervals

• Minimize number of processors to schedule all intervals

How many processors are needed for this example?

Prove that you cannot schedule this set of intervals with two processors

Depth: maximum number of intervals active
Algorithm

- Sort by start times
- Suppose maximum depth is d, create d slots
- Schedule items in increasing order, assign each item to an open slot

- Correctness proof: When we reach an item, we always have an open slot

What happens on “Random” sets of intervals

- Given n random intervals
  - What is the expected number independent intervals
  - What is the expected depth

What is a random set of intervals

- Method 1:
  - Each interval assigned random start position in [0, 1]
  - Each interval assigned a random end length in [0, 1]
- 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

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 = $f_i - d_i$ if $f_i >= d_i$

Example

<table>
<thead>
<tr>
<th>Time</th>
<th>Deadline</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>

Determine the minimum lateness

<table>
<thead>
<tr>
<th>Time</th>
<th>Deadline</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>12</td>
</tr>
</tbody>
</table>