Announcements

- Today’s lecture
  - Kleinberg-Tardos, 4.2, 4.3
- Friday
  - Kleinberg-Tardos, 4.4, 4.5

Stable Matching Results

- Averages of 5 runs
- Much better for M than W
- Why is it better for M?
- What is the growth of m-rank and w-rank as a function of n?

<table>
<thead>
<tr>
<th>n</th>
<th>m-rank</th>
<th>w-rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>500</td>
<td>5.10</td>
<td>88.01</td>
</tr>
<tr>
<td>500</td>
<td>7.02</td>
<td>89.11</td>
</tr>
<tr>
<td>500</td>
<td>8.07</td>
<td>89.36</td>
</tr>
<tr>
<td>500</td>
<td>8.12</td>
<td>87.87</td>
</tr>
<tr>
<td>500</td>
<td>7.01</td>
<td>90.22</td>
</tr>
<tr>
<td>500</td>
<td>7.87</td>
<td>90.04</td>
</tr>
<tr>
<td>1000</td>
<td>6.83</td>
<td>141.91</td>
</tr>
<tr>
<td>1000</td>
<td>6.93</td>
<td>151.71</td>
</tr>
<tr>
<td>1000</td>
<td>7.03</td>
<td>153.33</td>
</tr>
<tr>
<td>1000</td>
<td>7.12</td>
<td>144.96</td>
</tr>
<tr>
<td>1000</td>
<td>7.22</td>
<td>135.81</td>
</tr>
<tr>
<td>1000</td>
<td>7.04</td>
<td>139.68</td>
</tr>
<tr>
<td>2000</td>
<td>7.83</td>
<td>257.79</td>
</tr>
<tr>
<td>2000</td>
<td>7.93</td>
<td>261.79</td>
</tr>
<tr>
<td>2000</td>
<td>11.22</td>
<td>176.17</td>
</tr>
<tr>
<td>2000</td>
<td>11.78</td>
<td>275.20</td>
</tr>
<tr>
<td>2000</td>
<td>11.56</td>
<td>275.02</td>
</tr>
<tr>
<td>2000</td>
<td>9.29</td>
<td>269.69</td>
</tr>
</tbody>
</table>

Approximation Algorithms

- Compare solution of approximation algorithm with the optimal algorithm
  - Earliest deadline first
  - Earliest starttime first
  - Shortest interval first
  - Fewest conflicts first

Scheduling Intervals

- Given a set of intervals
  - What is the largest set of non-overlapping intervals
  - Compare heuristics with optimal
- Suppose the n intervals are “random”
  - What is the expected number of independent intervals
- Determine \([x,y]\) by
  - \(x = \text{randomDouble}(0, 1.0)\)
  - \(y = \text{randomDouble}(x, 1.0)\)

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
  \[ L_i = f_i - d_i \text{ if } f_i \geq d_i \]

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>
</tbody>
</table>

Lateness 1

<table>
<thead>
<tr>
<th>Time</th>
<th>Deadline</th>
</tr>
</thead>
<tbody>
<tr>
<td>a_3</td>
<td>2</td>
</tr>
<tr>
<td>a_4</td>
<td>3</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>
</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>3</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>12</td>
</tr>
</tbody>
</table>

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

Lemma

- If there is an inversion i, j, there is a pair of adjacent jobs i', j' which form an inversion

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

Correctness Proof

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

Subsequence Testing

- Is $a_1 a_2 ... a_m$ a subsequence of $b_1 b_2 ... b_n$?
  - e.g. is A,B,C,D,C,B,A a subsequence of
    A,A,C,B,A,B,C,B,D,B,D,C,B,A,B,A

Friday