Announcements

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

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)
  - Graph Coloring
  - Homework Scheduling
  - Optimal Caching

Greedy Graph Coloring

Theorem: An undirected graph with maximum degree \( K \) can be colored with \( K + 1 \) colors

Coloring Algorithm, Version 1

Let \( k \) be the largest vertex degree
Choose \( k \) colors

for each vertex \( v \)
  Color\([v]\) = uncolored

for each vertex \( v \)
  Let \( c \) be a color not used in \( N[v] \)
  Color\([v]\) = \( c \)

Coloring Algorithm, Version 2

for each vertex \( v \)
  Color\([v]\) = uncolored

for each vertex \( v \)
  Let \( c \) be the smallest color not used in \( N[v] \)
  Color\([v]\) = \( c \)
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 $f_i >= d_i$

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>
</tbody>
</table>

Lateness 1: 1
Lateness 3: 3

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>

Greedy Algorithm

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

Analysis

- Suppose the jobs are ordered by deadlines, $d_1 <= d_2 <= \ldots <= d_n$
- 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></td>
<td></td>
</tr>
<tr>
<td>a_1</td>
<td>a_2</td>
</tr>
<tr>
<td>a_3</td>
<td>a_4</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 <= d_j, and j scheduled immediately before i. Interchanging i and j does not increase the maximum lateness.

Proof by Bubble Sort

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 . . .

Later this week