CSE 421
Algorithms

Richard Anderson
Lecture 7
Greedy Algorithms: Homework Scheduling and Optimal Caching

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

Homework Scheduling

- Tasks to perform
- Deadlines on the tasks
- Freedom to schedule tasks in any order

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

Time	Deadline
2	2
3	4

2	3	Lateness 1
3	2	Lateness 3

Determine the minimum lateness

Time	Deadline
2	6
3	4
4	5
5	12

Greedy Algorithm

- Earliest deadline first
- Order jobs by deadline

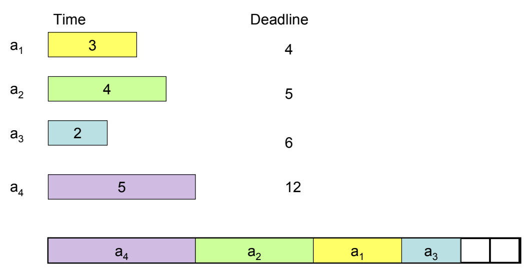
- This algorithm is optimal

Analysis

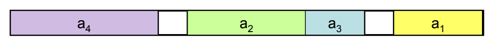
- Suppose the jobs are ordered by deadlines, $d_1 \le d_2 \le \ldots \le d_n$
- A schedule has an *inversion* if job j is scheduled before i where j > i

- 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



Lemma: There is an optimal schedule with no idle time



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

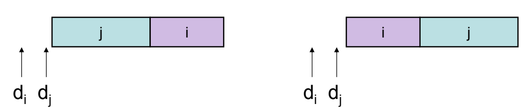
- Note on proof techniques
  - This type of can be important for keeping proofs clean
  - It allows us to make a simplifying assumption for the remainder of the proof

Lemma

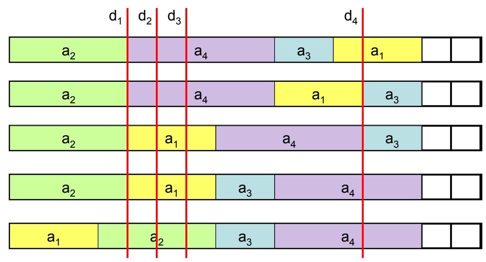
- If there is an inversion i, j, there is a pair of adjacent jobs i', j' which form an inversion

Interchange argument

- Suppose there is a pair of jobs i and j, with $i < j$, $d_i \le d_j$, 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

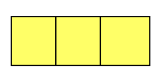
Extensions

- What if the objective is to minimize the sum of the lateness?
  - EDF does not seem to 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



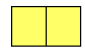
A, B, C, D, A, E, B, A, D, A, C, B, D, A

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
  - 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 . . .