CSE 421
Introduction to Algorithms

Richard Anderson

Winter 2024

Lecture 6 – Greedy Algorithms II

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?

n	m-rank	w-rank
500	5.10	98.05
500	7.52	66.95
500	8.57	58.18
500	6.32	75.87
500	5.25	90.73
500	6.55	77.95
1000	6.80	146.93
1000	6.50	154.71
1000	7.14	133.53
1000	7.44	128.96
1000	7.36	137.85
1000	7.04	140.40
2000	7.83	257.79
2000	7.50	263.78
2000	11.42	175.17
2000	7.16	274.76
2000	7.54	261.60
2000	8.29	246.62

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
  - Generate random interval [a,b]:
    - x = randomDouble(0, 1.0); y = randomDouble(x, 1.0)
    - a = min(x,y); b = max(x,y)

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

	Time	Deadline
a_1	2	2
a_2	3	4

2	3	Lateness 1
3	2	Lateness 3

Determine the minimum lateness

	Time	Deadline
a_1	2	6
a_2	3	4
a_3	4	5
a_4	5	12

Greedy Algorithm

- Earliest deadline first
- Order jobs by deadline

- This algorithm is optimal

Analysis

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

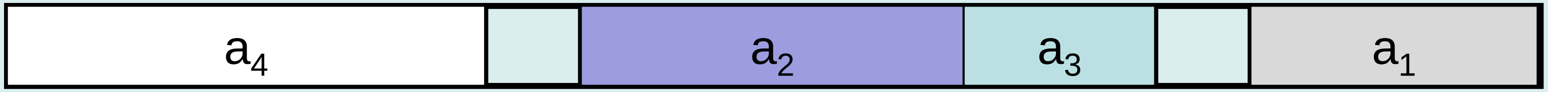
- The schedule A computed by the greedy algorithm has no inversions.
- Let O be the optimal schedule, we want to show that A has the same maximum lateness as O

List the inversions

	Time	Deadline
a_1	3	4
a_2	4	5
a_3	2	6
a_4	5	12

a_4	a_2	a_1	a_3		

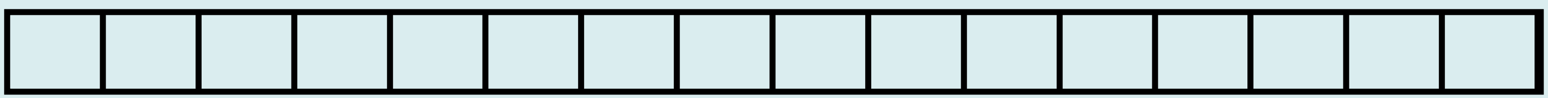
Lemma: There is an optimal schedule with no idle time



- 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

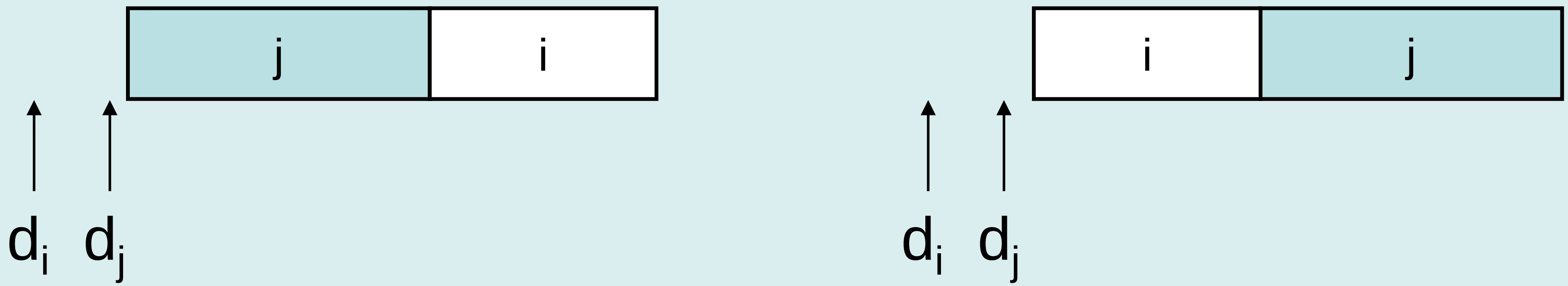
Lemma

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

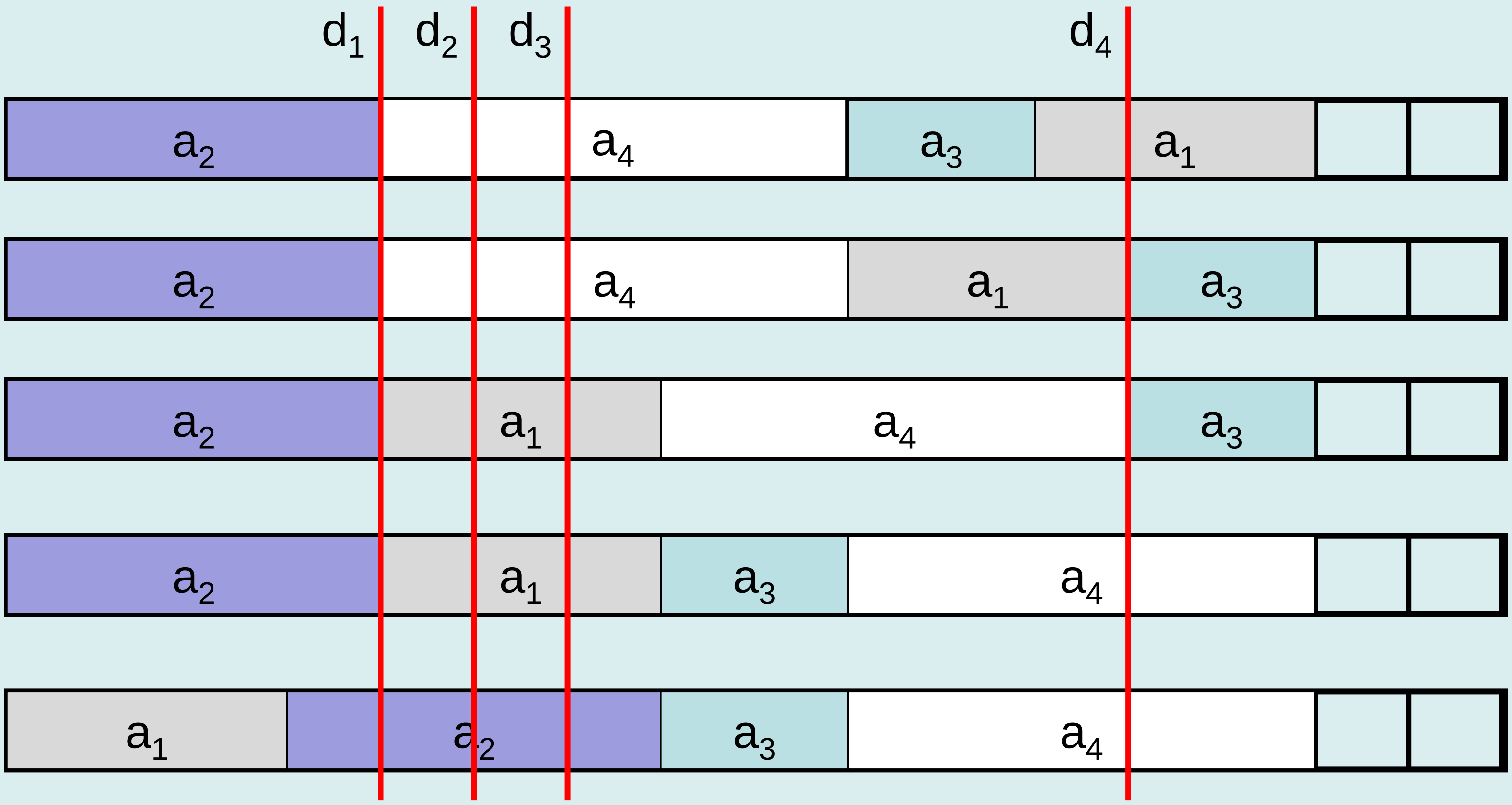


Interchange argument

- 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?

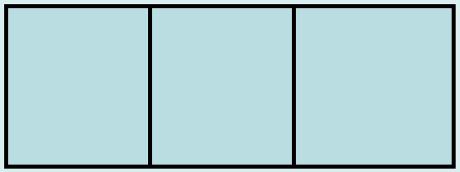
Extensions

- 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



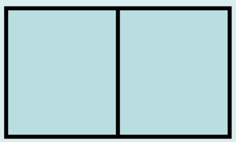
A, B, C, D, A, E, B, A, D, A, C, B, D, A

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 $a_1 a_2 \ldots a_m$ a subsequence of $b_1 b_2 \ldots b_n$?
  - 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,C,B,A,B,A

A	B	C	D	C	B	A

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

Friday