

# COMPSCI 311: Introduction to Algorithms

## Lecture 6: Greedy Algorithms – Exchange Arguments

Marius Minea

University of Massachusetts Amherst

slides credit: Dan Sheldon, Akshay Krishnamurthy, Andrew McGregor

13 February 2019

## Algorithm Design—Greedy

Greedy: make a single “greedy” choice at a time, don’t look back.

Greedy	
Formulate problem	
Design algorithm	
Prove correctness	✓
Analyze running time	
Specific algorithms	Dijkstra, MST

Focus is on proof techniques

- ▶ Last time: “greedy stays ahead” (inductive proof)
- ▶ This time: exchange argument

## Scheduling to Minimize Lateness

- ▶ You have a very busy month:  $n$  assignments are due, with different deadlines

Assignments:

```

1: |---|           (len=1, due=2)
2: |---|---|      (len=2, due=5)
3: |---|---|---|  (len=3, due=6)
4: |---|---|      (len=2, due=7)
    
```

Deadlines:

```

           d1           d2 d3 d4
|---|---|---|---|---|---|---|---|---|
0  1  2  3  4  5  6  7  8  9
    
```

- ▶ How should you schedule your time to “minimize lateness”?

## Scheduling to Minimize Lateness

Let’s formalize the problem. The input is:

- ▶  $t_j$  = length (in days) to complete assignment  $j$  (or “job”  $j$ )
- ▶  $d_j$  = deadline for assignment  $j$

What does a schedule look like?

- ▶  $s_j$  = start time for assignment  $j$  (selected by algorithm)
- ▶  $f_j = s_j + t_j$  finish time

How to evaluate a schedule?

- ▶ Lateness of assignment  $j$  is  $l_j = \begin{cases} 0 & \text{if } f_j \leq d_j \\ f_j - d_j & \text{if } f_j > d_j \end{cases}$
- ▶ Maximum lateness  $L = \max_j l_j$

**Goal:** schedule so maximum lateness is as small as possible

## Clicker Question 1

True or false: an algorithm to minimize maximum lateness will also find a schedule that is not late, if one exists.

- A. False, because the lateness function is not linear
- B. False, because it minimizes the maximum lateness, whereas we want all jobs to have lateness zero
- C. True

## Possible Greedy Approaches

- ▶ **Note:** scheduling work back-to-back (no idle time) can’t hurt  $\Rightarrow$  schedule determined just by order of assignments

```

1: |---|           (len=1, due=2)
2: |---|---|      (len=2, due=5)
3: |---|---|---|  (len=3, due=6)
4: |---|---|      (len=2, due=7)
    
```

- ▶ What order should we choose?
  - ▶ *Shortest Length:* ascending order of  $t_j$ .
  - ▶ *Smallest Slack:* ascending order of  $d_j - t_j$ .
  - ▶ *Earliest Deadline:* ascending order of  $d_j$ .
- ▶ *Earliest deadline first* is better in example. Let’s prove it is always optimal.

## Clicker Question 2

If two jobs have the same deadline, the earliest deadline first algorithm should schedule

- The longest job first, because then its lateness will be minimized
- The shortest job first, because that has a higher chance of finishing before the deadline
- Does not matter

## Identical Maximum Lateness

**Claim:** If in a schedule we swap two jobs with the same deadline, we get the same maximum lateness. True?

Not necessarily, if the jobs are not in earliest deadline first order! (Example)

**Claim:** If in an EDF schedule, we swap two jobs with the same deadline, we get the same maximum lateness.

**Proof:** Since the schedules are EDF, all jobs with the same deadline are scheduled in a consecutive block.

Then among those, the last one has the maximum lateness.

That finishing time does not change by swapping schedules within the block.

**Corollary** All EDF schedules have the same maximum lateness.

## Exchange Argument (False Start)

Assume jobs ordered by deadline  $d_1 \leq d_2 \leq \dots \leq d_n$ , so the greedy ordering is simply

$$A = 1, 2, \dots, n$$

**Claim:**  $A$  is optimal

**Proof attempt:** Suppose for contradiction that  $A$  is not optimal. Then, there is an optimal solution  $O$  with  $O \neq A$

- ▶ Since  $O \neq A$ , there must be two jobs  $i$  and  $j$  that are out of order in  $O$  (e.g.  $O = 1, 3, 2, 4$ )
- ▶ Let's swap  $i$  and  $j$  and show we get a *better* solution  $O'$
- ▶  $\implies O$  is *not* optimal. Contradiction, so  $A$  must be optimal.

**Problem?**  $O'$  may still be optimal. [Example?](#)

Can't do proof by contradiction in this way.

## Exchange Argument (Correct)

Suppose  $O$  optimal and  $O \neq A$ . Then we can modify  $O$  to get a new solution  $O'$  that is:

- No worse than  $O$
- Closer to  $A$  is some measurable way

$$O(\text{optimal}) \rightarrow O'(\text{optimal}) \rightarrow O''(\text{optimal}) \rightarrow \dots \rightarrow A(\text{optimal})$$

**High-level idea:** gradually transform  $O$  into  $A$  without hurting solution, thus preserving optimality.

**Concretely:** show 1 and 2 above.

## Exchange Argument for Scheduling to Minimize Lateness

Recall  $A = 1, 2, \dots, n$ .

For  $S \neq A$ , say there is an **inversion** if  $i$  comes before  $j$  but  $j < i$  (thus  $d_j \leq d_i$ )

**Claim:** if  $S$  has an inversion,  $S$  has a **consecutive inversion**—one where  $i$  comes immediately before  $j$ . Why?

**Main result:** Let  $O \neq A$  be an optimal schedule.

Then  $O$  has a consecutive inversion  $i, j$ .

We can swap  $i$  and  $j$  to get a new schedule  $O'$  such that:

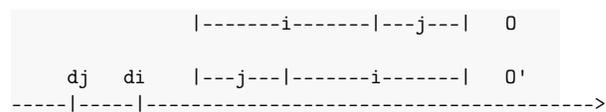
- $O'$  has one less inversion than  $O$
- Maximum lateness of  $O'$  is at most maximum lateness of  $O$

**Proof:**

- Obvious
- [Next slide\(s\)](#)

## Proof (Lateness does not increase)

Swapping a consecutive inversion ( $i$  precedes  $j$ ;  $d_j \leq d_i$ )



Consider the lateness  $\ell'_k$  of each job  $k$  in  $O'$ :

- ▶ If  $k \notin \{i, j\}$ , then lateness is unchanged:  $\ell'_k = \ell_k$
- ▶ Job  $j$  finishes earlier in  $O'$  than  $O$ :  $\ell'_j \leq \ell_j$
- ▶ Finish time of  $i$  in  $O'$  = finish time of  $j$  in  $O$ . Therefore

$$\ell'_i = f'_i - d_i = f_j - d_i \leq f_j - d_j = \ell_j$$

**Conclusion:**  $\max_k \ell'_k \leq \max_k \ell_k$ . Therefore  $O'$  is still optimal.

## Wrap-Up

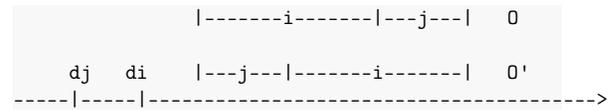
For any optimal  $O \neq A$  we showed that we could transform  $O$  to  $O'$  such that:

1.  $O'$  is still optimal
2.  $O'$  has one less inversion than  $A$

$$O(\text{optimal}) \rightarrow O'(\text{optimal}) \rightarrow O''(\text{optimal}) \rightarrow \dots \rightarrow A(\text{optimal})$$

Since there are at most  $\binom{n}{2}$  inversions, by repeating the process a finite number of times we see that  $A$  is optimal.

## Clicker Question 3



Consider the *total* lateness  $\ell'_i + \ell'_j$  in the new schedule. Which fact about total lateness follows from our argument?

- A. It is no more than  $2\ell_i$
- B. It is no more than  $2\ell_j$
- C. It is no more than  $\ell_i + \ell_j$
- D. None of the above

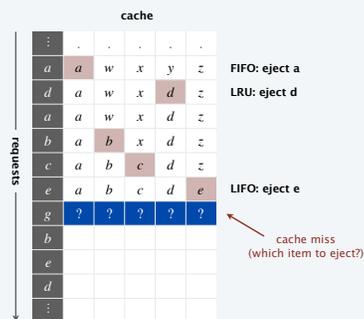
C would imply EDF is also optimal for minimizing total lateness. It is not. There is no known polynomial time algorithm for that.

### Optimal offline caching: greedy algorithms

**LIFO/FIFO.** Evict item brought in least (most) recently.

**LRU.** Evict item whose most recent access was earliest.

**LFU.** Evict item that was least frequently requested.

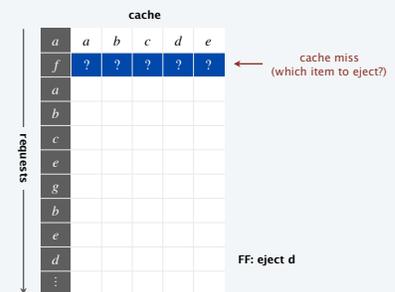


39

slide credit: Kevin Wayne / Pearson

### Optimal offline caching: farthest-in-future (clairvoyant algorithm)

**Farthest-in-future.** Evict item in the cache that is not requested until farthest in the future.



**Theorem.** [Bélády 1966] FF is optimal eviction schedule.

**Pf.** Algorithm and theorem are intuitive; proof is subtle.

40

slide credit: Kevin Wayne / Pearson

## Wrap-Up: Greedy Algorithms

Greedy: make a single "greedy" choice at a time, don't look back.

Greedy	
Formulate problem	
Design algorithm	
Prove correctness	✓
Analyze running time	
Specific algorithms	Dijkstra, MST

Proof techniques

- ▶ Last time: "greedy stays ahead" (inductive proof)
- ▶ This time: exchange argument

Need to formulate precisely; careful not to get arguments wrong!