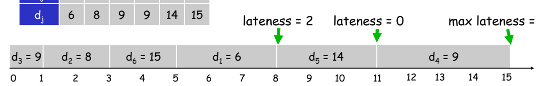
Objectives

- Wrap up minimizing max lateness

Scheduling to Minimizing Lateness

- Single resource processes one job at a time
- Job j requires t_j units of processing time and is due at time d_j (its deadline)
- If j starts at time s_j, it finishes at time $f_j = s_j + t_j$
- Lateness: $\ell_j = \max\{0, f_j - d_j\}$
- **Goal**: schedule all jobs to *minimize* **maximum lateness** $L = \max \ell_j$

	1	2	3	4	5	6
t_j	3	2	1	4	3	2
d_j	6	8	9	9	14	15

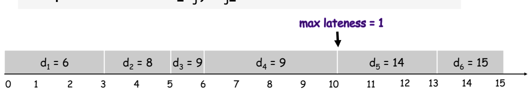


Note: not a sum total

Minimizing Lateness: Greedy Algorithm

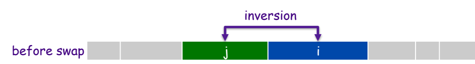
- Earliest deadline first.

```
Sort n jobs by deadline so that d1 ≤ d2 ≤ … ≤ dn
t = 0
for j = 1 to n
   Assign job j to interval [t, t + tj]
   sj = t
   fj = t + tj
   t = t + tj
output intervals [sj, fj]
```



Minimizing Lateness: Inversions

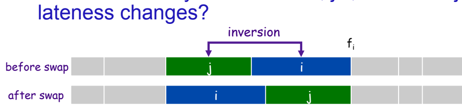
- Def. An ***inversion*** in schedule S is a pair of jobs i and j such that: $d_i < d_j$ but j scheduled before i



Greedy's schedule has no inversions!

Minimizing Lateness: Inversions

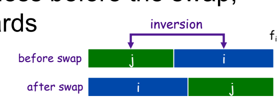
- Claim. Swapping two adjacent, inverted jobs reduces the number of inversions by one and does *not increase the max lateness*
  - How do we know inversions are adjacent?
- Pf Setup. Let ℓ be the lateness before the swap, and let ℓ' be it afterwards
  - What can we say about how i's, j's, and other jobs' lateness changes?



By def of inversion, $d_i < d_j$

Minimizing Lateness: Inversions

- Claim. Swapping two adjacent, inverted jobs reduces the number of inversions by one and does *not increase the max lateness*.
- Pf. Let ℓ be the lateness before the swap, and let ℓ' be it afterwards
  - $\ell'_k = \ell_k$ for all $k \neq i, j$
  - Know: $d_i < d_j$
  - $\ell'_i \leq \ell_i$
  - If job j is late:



$$
\begin{aligned}
\ell'_j &= f'_j - d_j && \text{(definition)}\\
&= f_i - d_j && (j \text{ finishes at time } f_i)\\
&\leq f_i - d_i && (i < j)\\
&\leq \ell_i && \text{(definition)}
\end{aligned}
$$

Minimizing Lateness: Analysis of Greedy Algorithm

- Theorem. Greedy schedule S is optimal
- Pf idea. Convert Opt to Greedy
  - Does opt schedule have idle time?
  - What if opt schedule has no inversions?
  - What if opt schedule has inversions?

Minimizing Lateness: Analysis of Greedy Algorithm

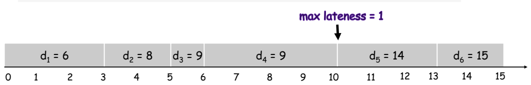
- Theorem. Greedy schedule S is optimal
- Pf. Define S* to be an optimal schedule that has the fewest number of inversions, and let's see what happens
  - Can assume S* has no idle time
  - If S* has no inversions, then S = S*
  - If S* has an inversion, let i-j be an adjacent inversion
    - Swapping i and j does not increase the maximum lateness and strictly decreases the number of inversions
    - This contradicts definition of S* ▪

Analyzing Running Time

- Earliest deadline first.

```
Sort n jobs by deadline so that d1 ≤ d2 ≤ … ≤ dn
t = 0
for j = 1 to n
   Assign job j to interval [t, t + tj]
   sj = t
   fj = t + tj
   t = t + tj
output intervals [sj, fj]
```

O(n logn)



What is the runtime of this algorithm?

Greedy Exchange Proofs

1. Label your algorithm's solution and a general solution.
   - For example, let A = $\{a_1, a_2, ..., a_k\}$ be the solution generated by your algorithm, and let O = $\{o_1, o_2, ..., o_m\}$ be an arbitrary (or optimal) feasible solution.
2. Compare greedy with other solution.
   - Assume that your arbitrary/optimal solution is not the same as your greedy solution (since otherwise, you are done).
   - Typically, you can isolate a simple example of this difference, such as one of the following:
     - There is an element of O that is not in A and an element of A that is not in O
     - There are 2 consecutive elements in O in a different order than they are in A (i.e., there is an *inversion*).
3. Exchange.
   - Swap the elements in question in O (either swap one element out and another in for the first case, or swap the order of the elements in the second case), and argue that you have a solution that is no worse than before.
   - Then argue that if you continue swapping, you eliminate all differences between O and A in a *finite* # of steps without worsening the solution's quality.
   - Thus, the greedy solution produced is just as good as any optimal solution, and hence is optimal itself.

Greedy Analysis Strategies

- Greedy algorithm stays ahead. Show that after each step of the greedy algorithm, its solution is at least as good as any other algorithm's.
- Exchange argument. Gradually transform any solution to the one found by the greedy algorithm without hurting its quality.
- Structural. Discover a simple "structural" bound asserting that every possible solution must have a certain value. Then show that your algorithm always achieves this bound.

Assignments

- Read Chapter 4
  - Wiki due next Wednesday
- Friday: Exam 1 Due