CS 381 – FALL 2019

Week 5.2, Wed, Sept 18

Homework 3 Due: September 23rd , 2019 @ 11:59PM on Gradescope
Midterm 1: September 25 (evening)

Homework 3 Released

- Due: September 23 (11:59PM) on Gradescope
- Shorter than Homework 2
- **Goal:** Practice Greedy Algorithms before Midterm

Midterm 1

- Practice Midterm Released Soon

- Topics:
  - Induction
  - Big-O
  - Divide and Conquer
    - Sorting, Counting Inversions, Maximum Subarray, Skyline Problem, Karatsuba Multiplication
  - Recurrences
    - Deriving a Recurrence
    - Unrolling
    - Recursion Trees
    - Master Theorem
  - Greedy Algorithms

- No Dynamic Programming (until Midterm 2)

Midterm 1: Logistics

- 90 minutes (8:00-9:30PM)
  - Tuesday/Thursday PSOs (SMTH 108)
  - Friday PSO (MTHW 210)
- 1 Page of Notes (Single-Sided)
- Standard paper (or A4) is acceptable
- Bring number 2 pencil (for scanned exam)
- Closed book, no calculators, no smartphones, no smartwatches, no laptops etc…

4.2 Scheduling to Minimize Lateness

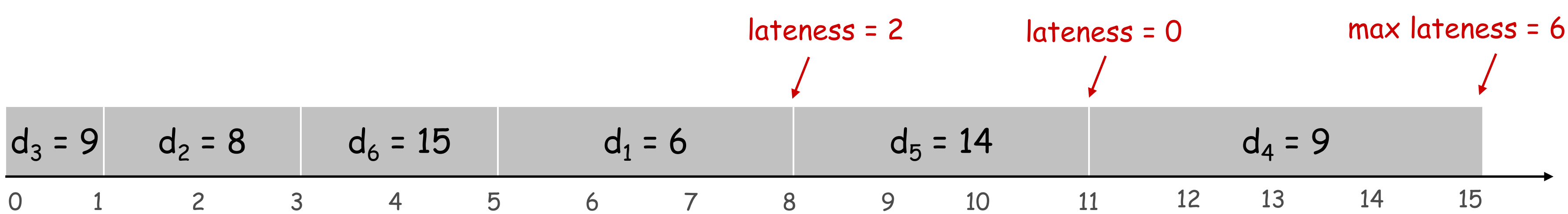
Scheduling to Minimizing Lateness

Minimizing lateness problem.

- Single resource processes one job at a time.
- Job j requires t_j units of processing time and is due at time d_j.
- 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 $L = \max \ell_j$.

Ex:

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



Minimizing Lateness: Greedy Algorithms

Greedy template. Consider jobs in some order.

- [Shortest processing time first] Consider jobs in ascending order of processing time t_j.

- [Earliest deadline first] Consider jobs in ascending order of deadline d_j.

- [Smallest slack] Consider jobs in ascending order of slack $d_j - t_j$.

Minimizing Lateness: Greedy Algorithms

Greedy template. Consider jobs in some order.

- [Shortest processing time first] Consider jobs in ascending order of processing time t_j.

	1	2
t_j	1	10
d_j	100	10

counterexample

- [Smallest slack] Consider jobs in ascending order of slack $d_j - t_j$.

	1	2
t_j	1	10
d_j	2	10

counterexample

Clicker Question

Greedy template. Consider jobs in some order.

- [Shortest processing time first] Consider jobs in ascending order of processing time t_j.
- [Smallest slack] Consider jobs in ascending order of slack $d_j - t_j$.

	1	2
t_j	9	5
d_j	2	10

Which greedy algorithms outputs the optimal schedule (minimizes the maximum lateness)?

A. Smallest Slack

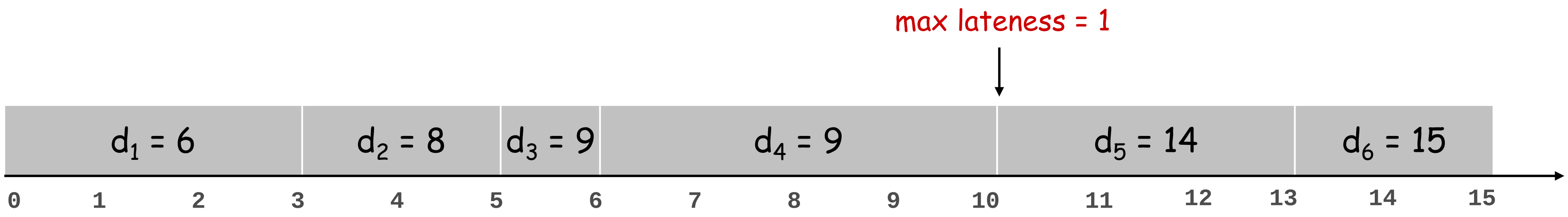
B. Shortest Processing Time First

C. Both

D. Neither

Minimizing Lateness: Greedy Algorithm

Greedy algorithm. Earliest deadline first.

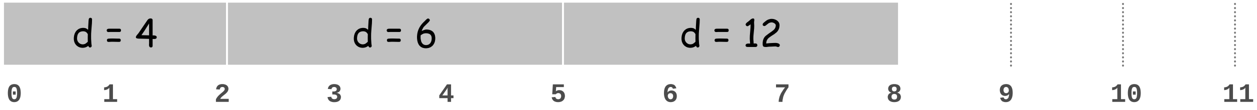



```
Sort n jobs by deadline so that d_1 ≤ d_2 ≤ … ≤ d_n

t ← 0
for j = 1 to n
   Assign job j to interval [t, t + t_j]
   s_j ← t, f_j ← t + t_j
   t ← t + t_j
output intervals [s_j, f_j]
```

Minimizing Lateness: No Idle Time

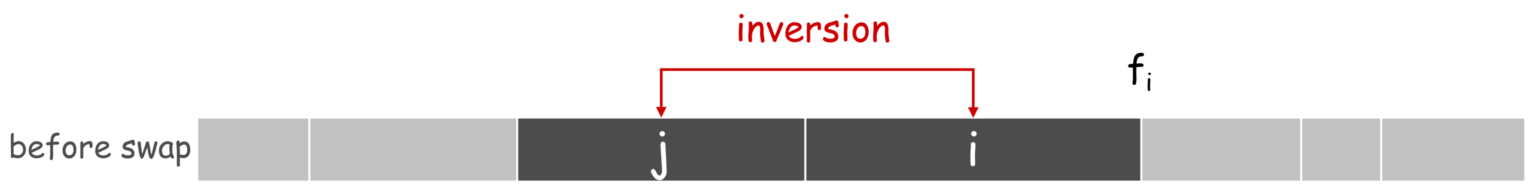
Observation. There exists an optimal schedule with no idle time.



Observation. The greedy schedule has no idle time.

Minimizing Lateness: Inversions

Def. Given a schedule S, an inversion is a pair of jobs i and j such that: i < j (i.e., $d_i < d_j$) but j scheduled before i.



[as before, we assume jobs are numbered so that $d_1 \le d_2 \le \ldots \le d_n$]

Observation. Greedy schedule has no inversions.

Observation. If a schedule (with no idle time) has an inversion, it has one with a pair of inverted jobs scheduled consecutively.

Proof: If (i,j) be inversion *minimizing number of intermediate jobs*. Suppose for contradiction that some job k was scheduled between jobs i and j.

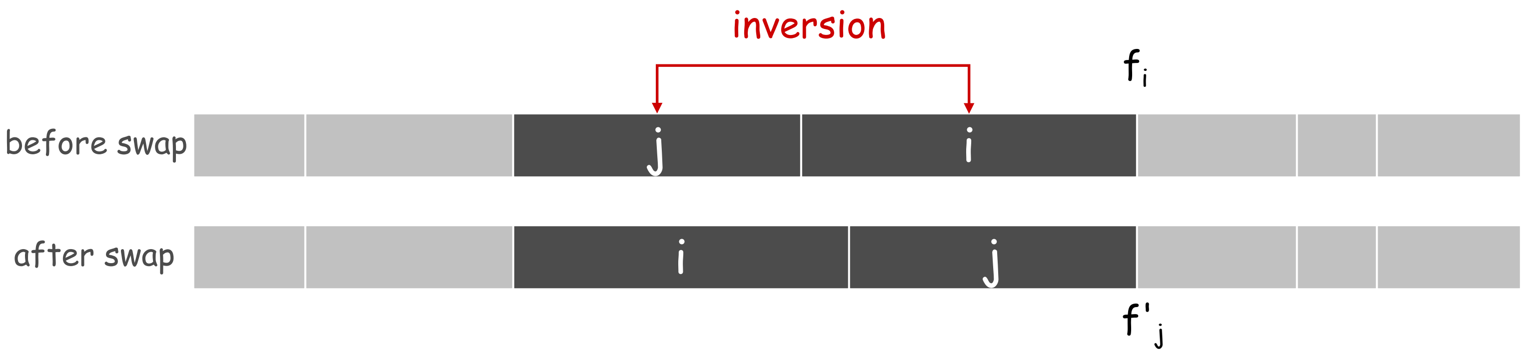
Case 1: $d_k \le d_i$ → (j,k) is inversion

Case 2: $d_k > d_i$ → (i,k) is an inversion

Contradicts minimality of (i,j)

Minimizing Lateness: Inversions

Def. Given a schedule S, an inversion is a pair of jobs i and j such that: i < j but j scheduled before i.



Claim. Swapping two consecutive, 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 $k \neq i, j$
- $\ell'_i \leq \ell_i$
- 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.

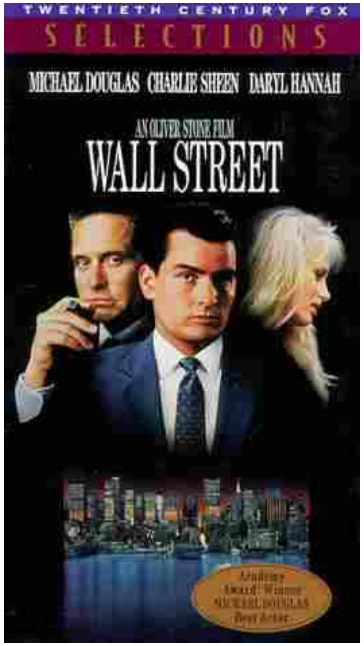
Pf. Define S* to be an optimal schedule that has the fewest number of inversions, and let's see what happens.

- 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* ▪

Coin Changing

Greed is good. Greed is right. Greed works. Greed clarifies, cuts through, and captures the essence of the evolutionary spirit.

- Gordon Gecko (Michael Douglas)



Coin Changing

Goal. Given currency denominations: 1, 5, 10, 25, 100, devise a method to pay amount to customer using fewest number of coins.

Ex: 34¢.

Cashier's algorithm. At each iteration, add coin of the largest value that does not take us past the amount to be paid.

Ex: $2.89.

Goal. Given currency denominations: 1, 5, 10, 25, 100, devise a method to pay amount to customer using fewest number of coins.

Which (if any) of the following claims is not true?

A. The optimal solution uses at most 2 dimes

B. The optimal solution uses at most 1 nickel

C. The optimal solution uses at most 3 quarters

D. The optimal solution uses at most 4 pennies

E. **All of the claims above are true**

Coin-Changing: Greedy Algorithm

Cashier's algorithm. At each iteration, add coin of the largest value that does not take us past the amount to be paid.

```
Sort coins denominations by value: c₁ < c₂ < … < cₙ.

  coins selected
S ← φ
while (x ≠ 0) {
   let k be largest integer such that cₖ ≤ x
   if (k = 0)
      return "no solution found"
   x ← x - cₖ
   S ← S ∪ {k}
}
return S
```

Q. Is cashier's algorithm optimal?

Coin-Changing: Analysis of Greedy Algorithm

Theorem. Greed is optimal for U.S. coinage: 1, 5, 10, 25, 100.

Pf. (by induction on x)

- Consider optimal way to change $c_k \leq x < c_{k+1}$: greedy takes coin k.
- We claim that any optimal solution must also take coin k.
  - if not, it needs enough coins of type $c_1, \ldots, c_{k-1}$ to add up to x
  - table below indicates no optimal solution can do this
- Problem reduces to coin-changing $x - c_k$ cents, which, by induction, is optimally solved by greedy algorithm. ▪

k	c_k	All optimal solutions must satisfy	Max value of coins 1, 2, …, k-1 in any OPT
1	1	$P \leq 4$	-
2	5	$N \leq 1$	4
3	10	$N + D \leq 2$	4 + 5 = 9
4	25	$Q \leq 3$	20 + 4 = 24
5	100	no limit	75 + 24 = 99

Coin-Changing: Analysis of Greedy Algorithm

Observation. Greedy algorithm is sub-optimal for US postal denominations: 1, 10, 21, 34, 70, 100, 350, 1225, 1500.

Counterexample. 140¢.

- Greedy: 100, 34, 1, 1, 1, 1, 1, 1.
- Optimal: 70, 70.