CS 381 – FALL 2019

Week 4.3, Friday, Sept 13

Homework 2 Due: September 16th, 2019 @ 11:59PM on Gradescope
September 16: Guest Lecture (Prof. Hambrusch)
Office Hours: ~~Monday (9/16) at 2:30PM~~ Friday (today) at 4:30 PM

Homework 2 Reminders

- Must include collaborator/resource statement
  - No credit for solutions that don't include CR statement
- Must type solutions (expectation to use math notation $\sqrt{n}$ vs n^(1/2))
  - Only allowed to scan hand drawn graphs/diagrams
  - No credit if the entire solution is a scan of your handwritten homework
- Remember to select the appropriate pages for each problem on Gradescope

Majority Element Problem

Input: Array A[1…n] of numbers (not sorted)

Output:

x ---- if x=A[i] for more than n/2 array elements

"N/A" ---- if no majority element exists

Example 1:

Input: A = [1 7 2 9 7 2 7]

Output: "N/A"

Example 2:

Input: A = [1 7 2 **7** 7 2 7]

Output: 7

Observation: If A does contain a majority element x then x=Median(A)

Correction: Sorted Majority problem

Suppose A is an array of size n containing increasingly sorted entries. We can determine whether A has a majority element in what time (check best bound)

~~A. O(1)~~ Impossible

B. O(log n) Binary Search

C. O($\log^2$ n)

D. O(n)

E. O(n log n)

Majority Element Problem

Input: Array A[1…n] of numbers (not sorted)

Output:

x ---- if x=A[i] for more than n/2 array elements

"N/A" ---- if no majority element exists

Solution 1: Sort array, find median x and binary search $O(n \log n)$

- i := unique location s.t. $A[i] \neq x$ but $A[i+1] = x$ (first occurrence of x)
- j:= unique location s.t. $A[j] \neq x$ but $A[j-1] = x$ (last occurrence of x)
- **Return**

$$\begin{cases} \mathrm{N/A} & j-i-1 \leq \frac{n}{2} \\ \boldsymbol{x} = A\left[\left\lceil \frac{n}{2} \right\rceil\right] & j-i-1 > \frac{n}{2} \end{cases}$$

Example:

Input: A = [1 2 7 7 7 7 8]

Output: 7

i+1 Median j-1

Majority Element Problem

Input: Array A[1…n] of numbers (not sorted)

Output:

x ---- if x=A[i] for more than n/2 array elements

"N/A" ---- if no majority element exists

Solution 2: Find Median and Scan Array to Count Matches

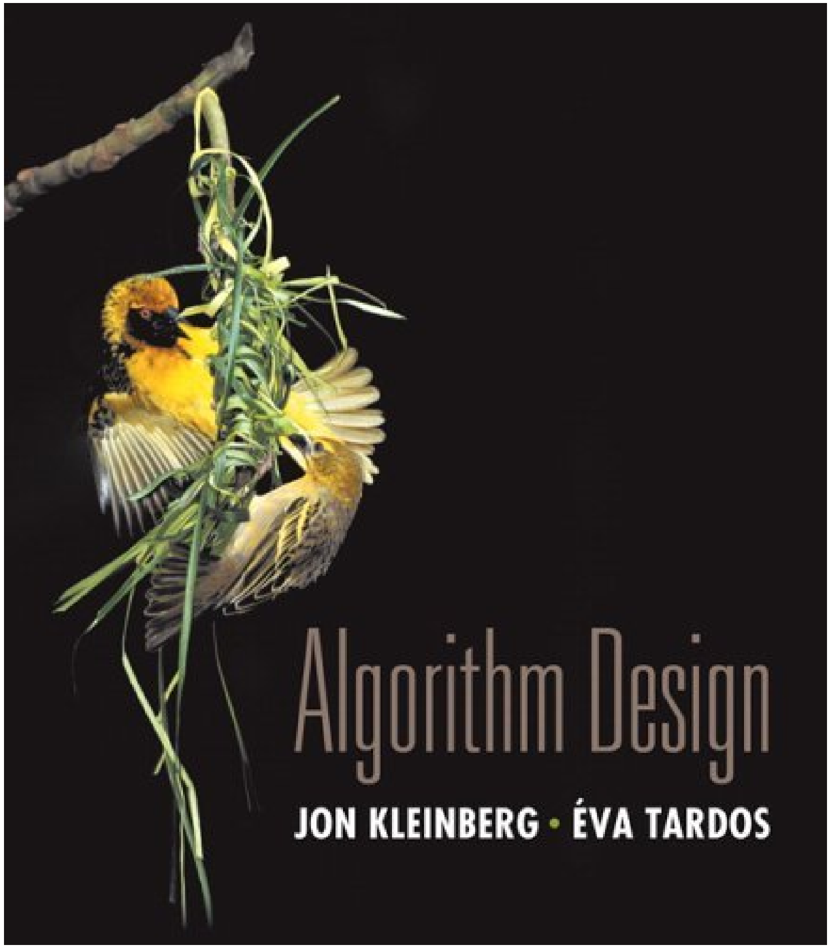
- X= Median(A)
- Count = 0
- For i = 1 to n
  - If X=A[i] then Count= Count + 1;
- **Return**

$$\begin{cases} \text{N/A} & count \leq \frac{n}{2} \\ x = A\left[\left[\frac{n}{2}\right]\right] & count > \frac{n}{2} \end{cases}$$

median

Running time: O(n) + O(n) = O(n)

Scan array

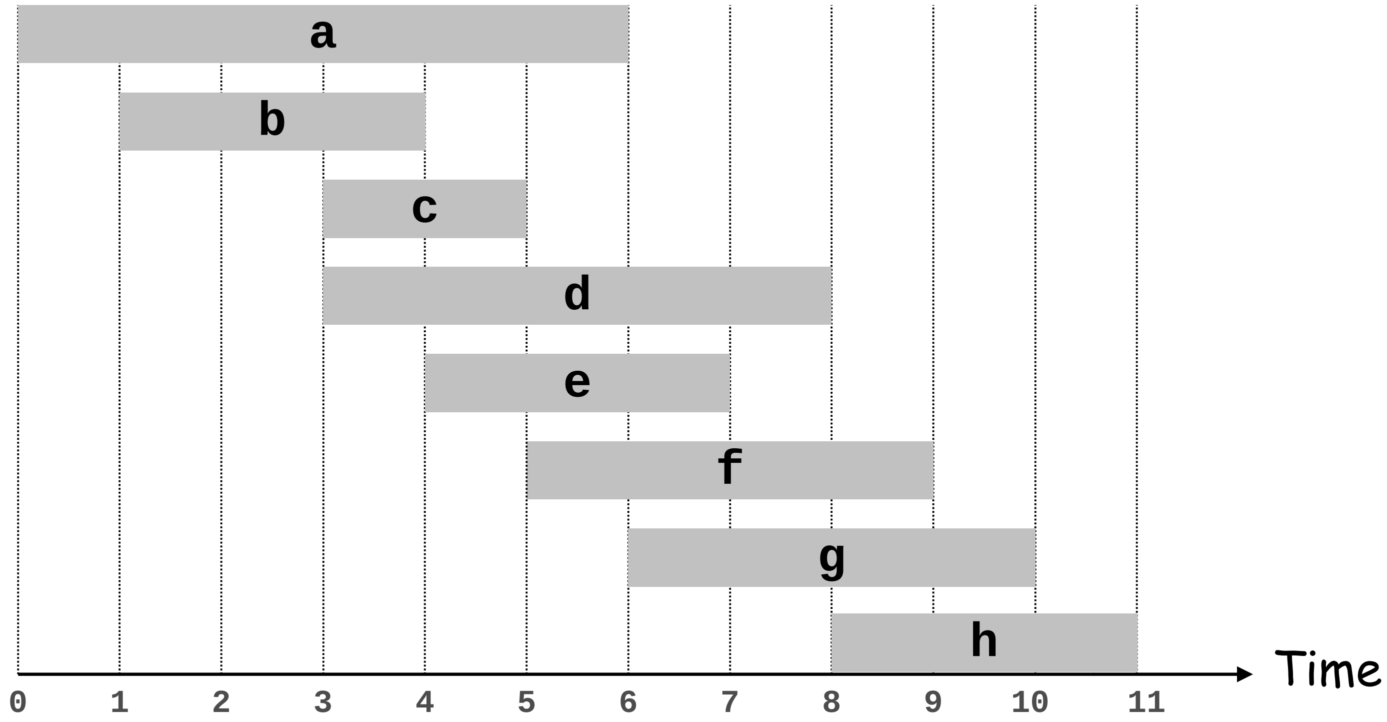


Greedy Algorithms

4.1 Interval Scheduling

Interval scheduling.

- Job j starts at s_j and finishes at f_j.
- Two jobs compatible if they don't overlap.
- Goal: find maximum subset of mutually compatible jobs.



Interval Scheduling: Greedy Algorithms

Greedy template. Consider jobs in some natural order.
Take each job provided it's compatible with the ones already taken.

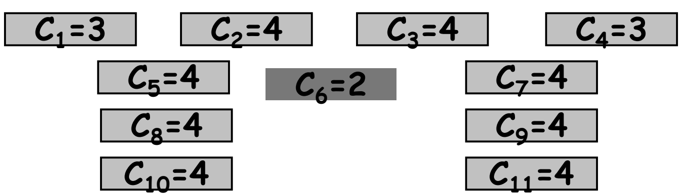
- [Earliest start time] Consider jobs in ascending order of s_j.
- [Earliest finish time] Consider jobs in ascending order of f_j.
- [Shortest interval] Consider jobs in ascending order of $f_j - s_j$.
- [Fewest conflicts] For each job j, count the number of conflicting jobs c_j. Schedule in ascending order of c_j.

Interval Scheduling: Greedy Algorithms

Greedy template. Consider jobs in some natural order.
Take each job provided it's compatible with the ones already taken.

counterexample for earliest start time

counterexample for shortest interval



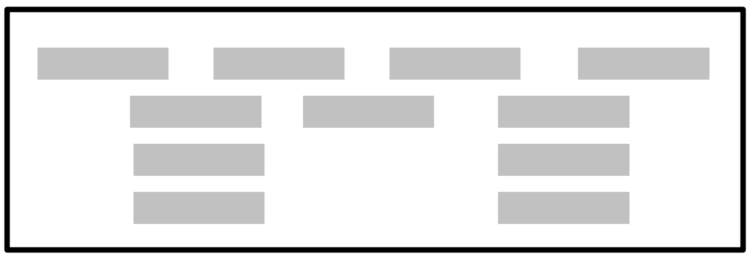
counterexample for fewest conflicts

Clicker Question

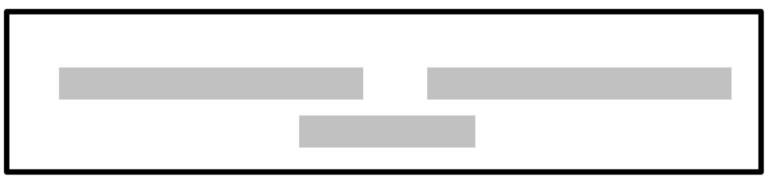
[Fewest conflicts] For each job j, count the number of conflicting jobs c_j. Schedule in ascending order of c_j.

Fewest conflicts <u>fails</u> to produce the optimal solution on which of the following inputs?

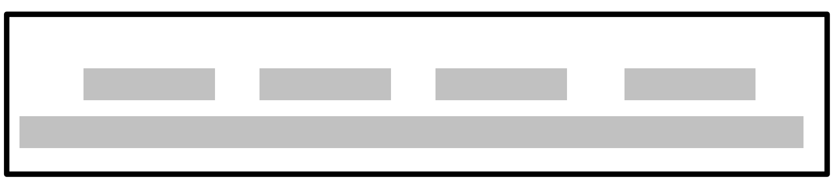
1.



2.



3.



A. Inputs 1 and 2

B. Inputs 2 and 3

C. Succeeds on all inputs

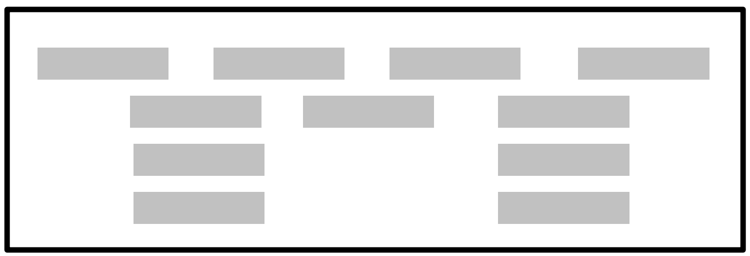
D. Fails on all inputs

E. Input 1 only

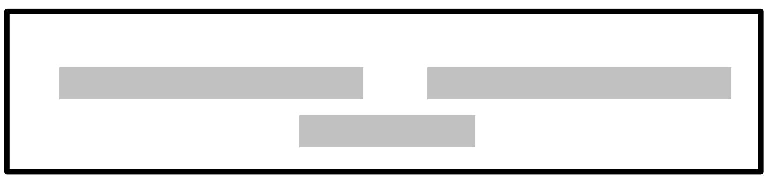
Clicker Question

[Fewest conflicts] For each job j, count the number of conflicting jobs c_j. Schedule in ascending order of c_j.

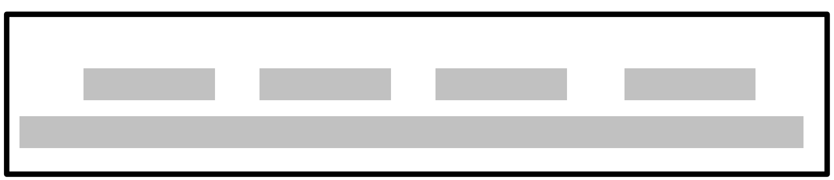
1.



2.



3.



Fewest conflicts <u>fails</u> to produce the optimal solution on which of the following inputs?

A. Inputs 1 and 2

B. Inputs 2 and 3

C. Succeeds on all inputs

D. Fails on all inputs

E. Input 1 only

Interval Scheduling: Greedy Algorithm

Greedy algorithm. Consider jobs in increasing order of finish time. Take each job provided it's compatible with the ones already taken.

```
Sort jobs by finish times so that f1 ≤ f2 ≤ ... ≤ fn.

 set of jobs selected
A ← φ
for j = 1 to n {
   if (job j compatible with A)
      A ← A ∪ {j}
}
return A
```

play

Implementation. O(n log n).

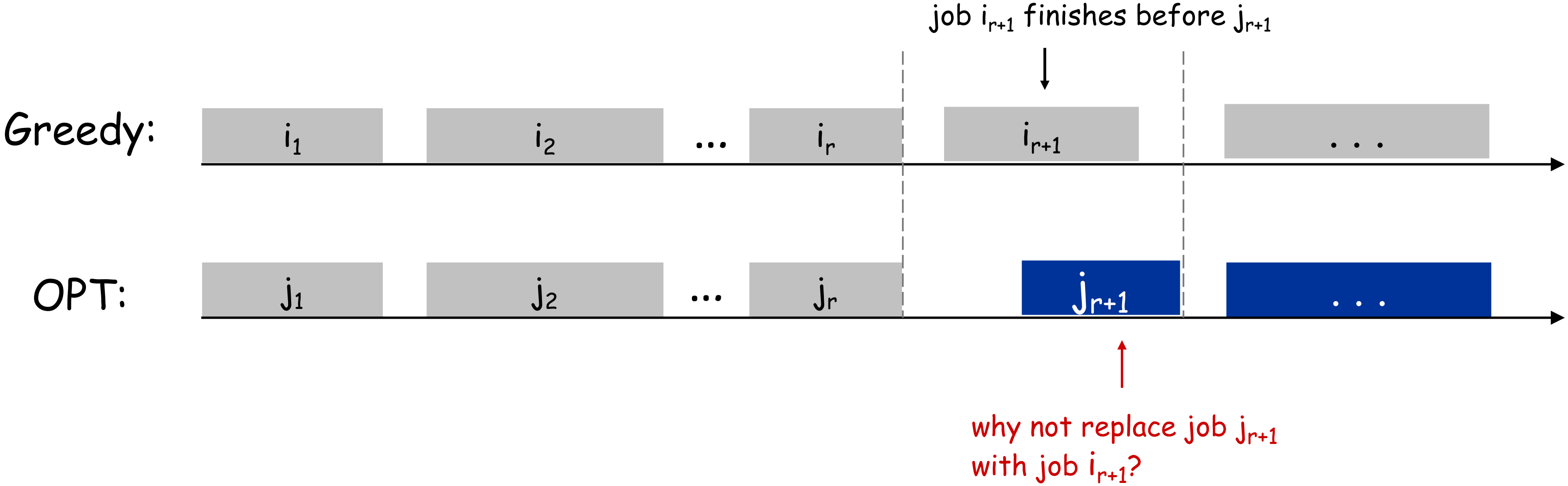
- Remember job j* that was added last to A.
- Job j is compatible with A if $s_j \geq f_{j^*}$.

Interval Scheduling: Analysis

Theorem. Greedy algorithm is optimal.

Pf. (by contradiction)

- Assume greedy is not optimal, and let's see what happens.
- Let $i_1, i_2, \ldots i_k$ denote set of jobs selected by greedy.
- Let $j_1, j_2, \ldots j_m$ denote set of jobs in the optimal solution with $i_1 = j_1, i_2 = j_2, \ldots, i_r = j_r$ for the largest possible value of r.

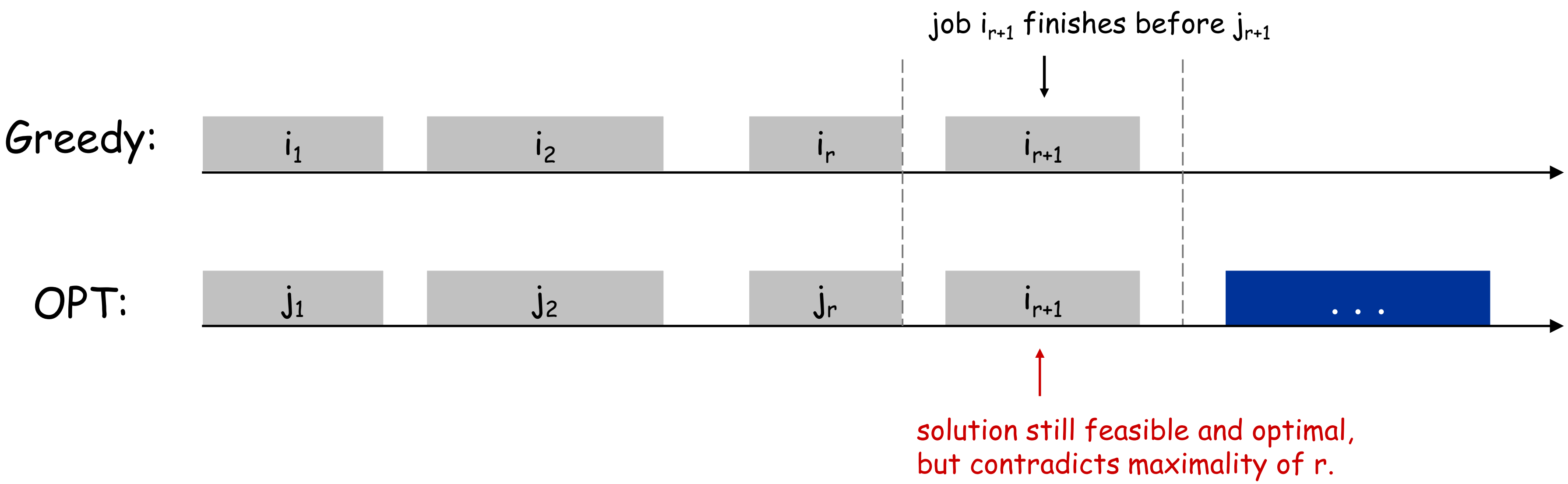


Interval Scheduling: Analysis

Theorem. Greedy algorithm is optimal.

Pf. (by contradiction)

- Assume greedy is not optimal, and let's see what happens.
- Let $i_1, i_2, \ldots i_k$ denote set of jobs selected by greedy.
- Let $j_1, j_2, \ldots j_m$ denote set of jobs in the optimal solution with $i_1 = j_1, i_2 = j_2, \ldots, i_r = j_r$ for the largest possible value of r.



4.2 Scheduling to Minimize Lateness

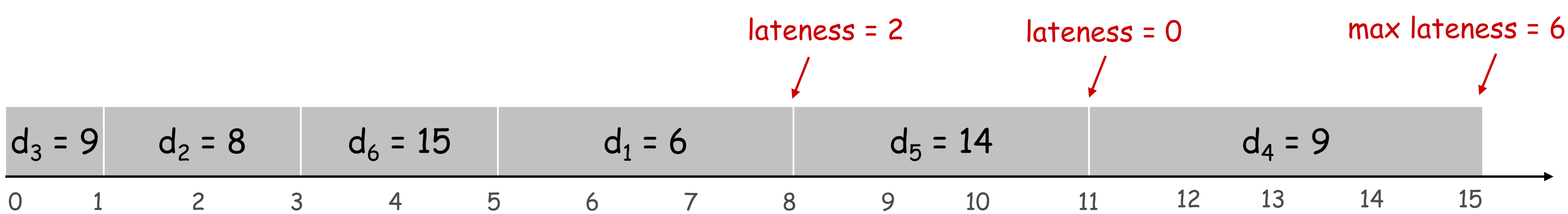
Scheduling to Minimizing Lateness

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

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