4.2 Scheduling to Minimize Lateness

SUBSECTION 4.2 OF KT's BOOK

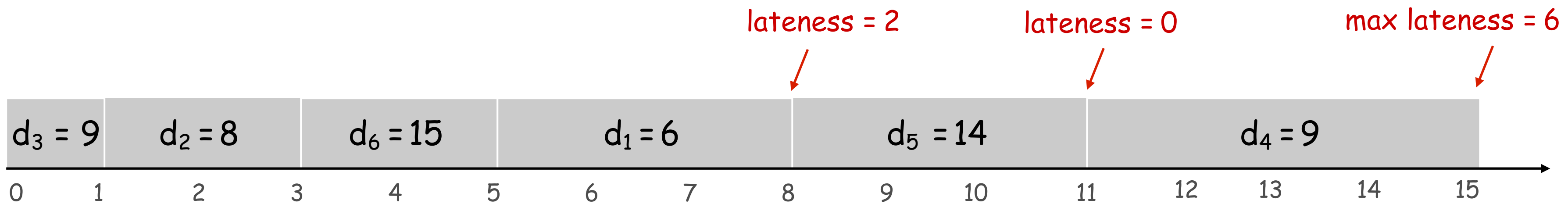
Scheduling to Minimizing Lateness

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.
- Solution: If j starts at time s_j, it finishes at time $f_j = s_j + t_j$.
- Lateness: $l_j = \max\{0, f_j - d_j\}$.
- ***Goal:*** *schedule all jobs to minimize maximum lateness* $L = \max l_j$.
- **Note: input** elements are **in blue, solution** elements are **in red, cost** elements are in **violet**

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.*

- [**G1**: 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

G1 solution: Job 1; Job 2 --> Latency = 1

Optimal Solution: Job 2; Job 1 --> Latency = 0

Minimizing Lateness: Greedy Algorithms

Greedy template. Consider jobs in some order.

- [G2 Smallest slack] Consider jobs in **ascending** order of slack $d_j - t_j$.

G2 Solution: Job 2; Job 1. Latency = 10

Optimal: Job 1; Job 2. Latency = 1

	1	2
t_j	1	10
d_j	2	10

counterexample

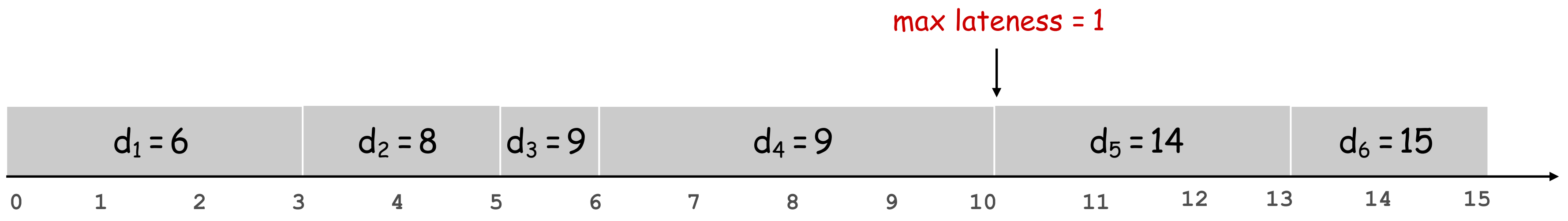
Minimizing Lateness: Greedy Algorithm

Greedy algorithm. Earliest deadline d first

Input: { (t_1,d_1), ..., (t_j,d_j),....(t_n,d_n) }

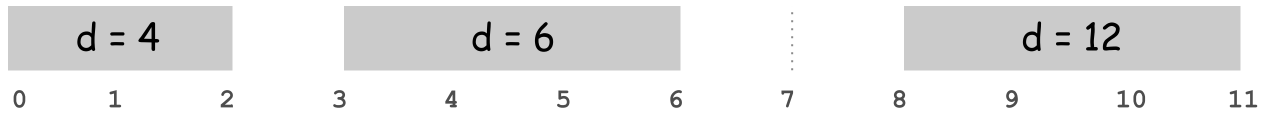
```
Sort n jobs by deadline so that $d_1 \le d_2 \le \dots \le d_n$

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 [$s_j$, $f_j$]
```

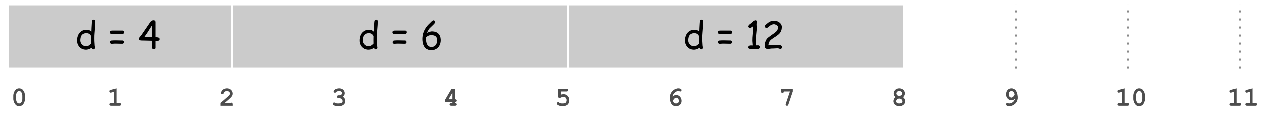
	1	2	3	4	5	6
t_j	3	2	1	4	3	2
d_j	6	8	9	9	14	15



Minimizing Lateness: No Idle Time

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

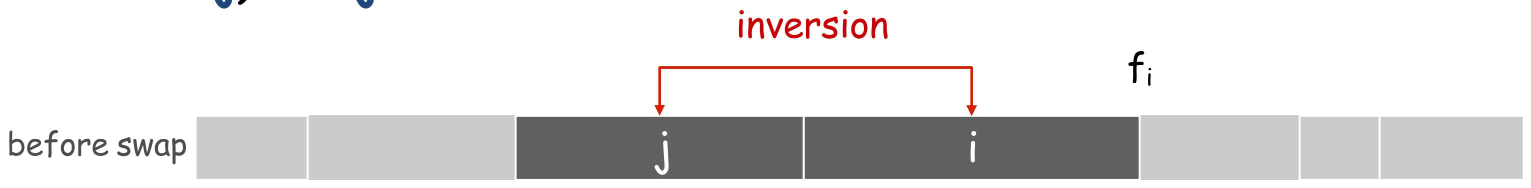




Observation. *The greedy schedule has no idle time.*

Minimizing Lateness: Inversions

Def. Given a **schedule** S, an **inversion** is a pair of jobs i and j such that: i< j (i.e. di <= dj) but j scheduled before i.



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

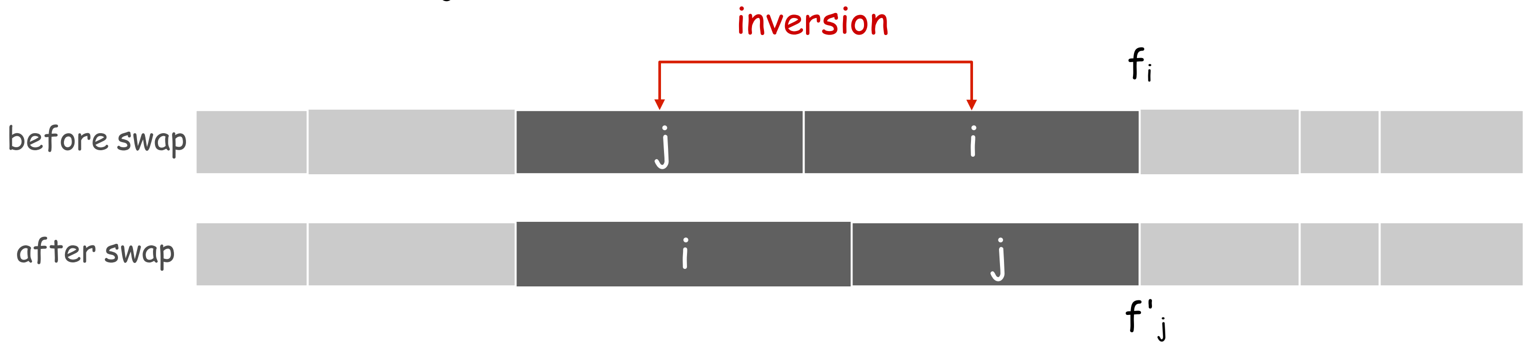
Observation. *Greedy schedule has no inversions.*

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

a ≤ b ≤ c .... c' : c". ... <= f : f' :
If b > f' then for some consecutive c', c" it must holds c' > c"

Minimizing Lateness: Inversions

Def. Given a schedule S, an **inversion** is a pair of jobs **i** and j such that:
i < j (w.r.t. deadline d) but **j is** scheduled <u>before</u> **i**



LEMMA (Exchange Arg.). Swapping two consecutive, inverted jobs **reduces** the number of inversions by **one** and does not increase the max lateness (the sum is commutative!).

- **Pf.** Let L be the lateness before the swap, and let L ' be it afterwards.

- $l'_k = l_k$ for all $k \neq i, j$
- $l'_i \leq l_i$
- If job j is late: →

$l'_j = f'_j - d_j$	(definition)
$= f_i - d_j$	(j finishes at time f_i)
$\leq f_i - d_i$	($i < j$)
$= l'_i$	(definition)

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 <u>does not increase</u> the maximum lateness and strictly <u>decreases</u> the **number of inversions**
  - this contradicts definition° of S* ▪

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.

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.

Exchange argument. Gradually transform any solution to the one found by the greedy algorithm without hurting its quality.

...

EXCERCISE 1 – PAGE 183

EXERCISE I: Prove that the Greedy Algorithm based on the **earliest finish time** is **optimal.**

SOLUTION: Let $A = \{i_1, i_2, \dots i_k\}$ denote set of jobs selected by ***Greedy;***

Let $J = \{j_1, j_2, \dots J_m\}$ denote set of jobs in an optimal solution. We know .....

Lemma 1 (*Greedy Stays Ahead*). For any $r = 1, \dots, k$ it holds $f(i_r) <= f(j_r)$

- Now, suppose (by contradiction) that optimal solution is such that **$m >= k+1$**. So,

$$J = \{ j_1, j_2, \dots, j_k, j_{k+1} \dots, j_m \}$$

- Apply **Lemma 1** on intervals i_k and j_k: → $f(j_k) >= f(i_k)$. (*)

- From (*), we get that the Greedy would have inserted j_{k+1} too! Since it is <u>compatible</u> with i_k as well! Contradiction with the assumption $|A| = k$!

EXCERCISE 2 AT PAGE 185

-BUYING ITEMS OF INCREASING COSTS

DO AS HOMEWORK!

HINTS: DON'T LOOK AT THE BOOK, TRY GREEDY SOLUTIONS, and PROVE :

- INVERSIONS in the good greedy ORDERING imply CONTRADICTIONS
- How prove CONTRADICTIONS? exchange argument