minimize the maximum lateness

Assume that there is only one processor

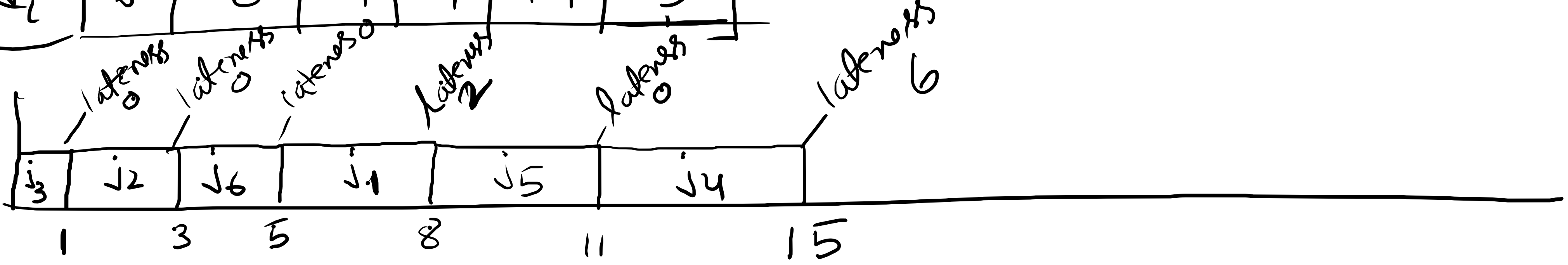
Input: A set J of n jobs $j_1, j_2, \ldots, j_n$ where each job j_i has a processing time t_i and a deadline d_i

output + objective: schedule the jobs in one processor such that the maximum amount time that any single job is past its deadline is minimized.

Exm

	j_1	j_2	j_3	j_4	j_5	j_6
t_i	3	2	1	4	3	2
d_i	6	8	9	9	14	15

$j_3 \to j_2 \to j_6 \to j_1 \to j_5 \to j_4$



objective: Find a schedule that minimizes the lateness

Greedy template:

- consider the jobs in some order Which order ??
- Assign the jobs in this order to the resource.

rule 1: shortest processing time ✗ H.W. counterexample

rule 2: shortest slack time ✗

$d_i - t_i$

rule 3: Earliest deadline first ✓

Algorithm

Earliest deadline first $(n, (t_1, d_1), (t_2, d_2), \ldots, (t_n, d_n))$

- sort the jobs by their deadlines.
- $d_1 \leq d_2 \leq \cdots \leq d_n$ be the order
- $t = 0$
- for $i = 1$ to n

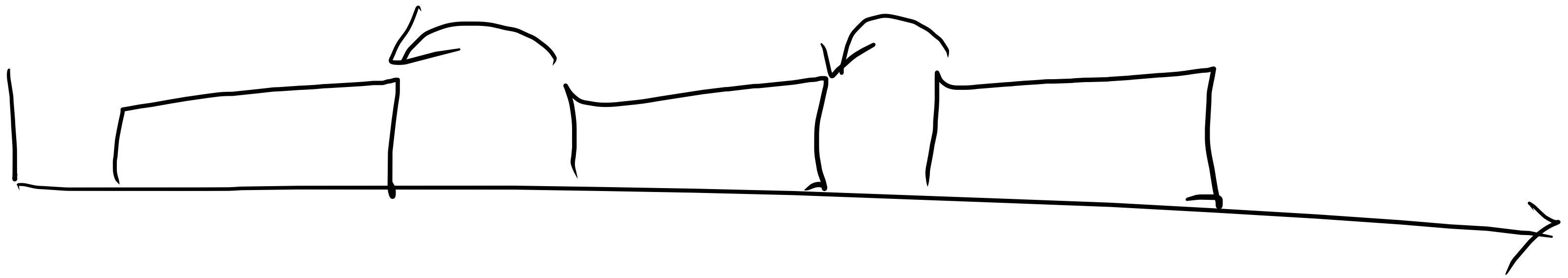
  $s_i = t$, $f_i = t + t_i$

  $t = t + t_i$
- output intervals. $[s_1, f_1], [s_2, f_2], \ldots, [s_n, f_n]$

running time $O(n \log n)$

correctness

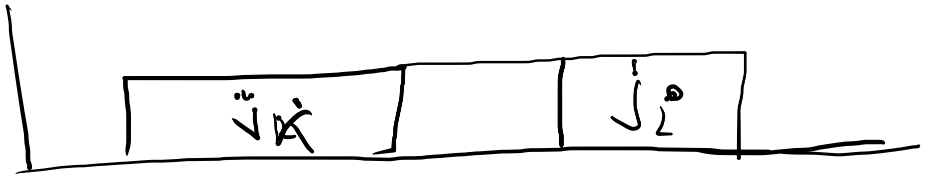
observation: There exists an optimum schedule with no idle time.



Observation about our greedy schedule

EDF algo returns a schedule with no idle time

Definition inversion



An inversion is a pair of jobs j_i and j_k such that $d_i < d_k$ but j_k is scheduled before j_i

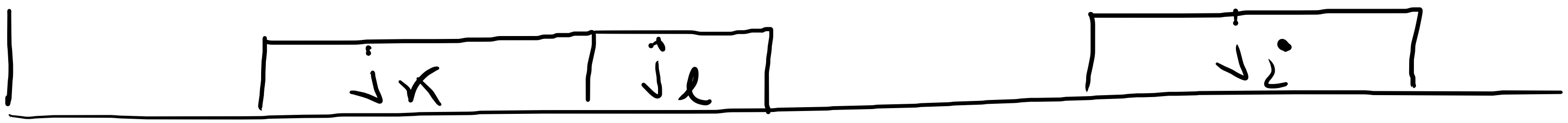
However $d_k > d_i$

observation about greedy

EDF does not have any inversion.

Claim If an idle-free schedule has an inversion then it is an adjacent inversion.

Proof j_i and j_k be a closest inversion but not adjacent.



$d_i < d_k$

There are two cases

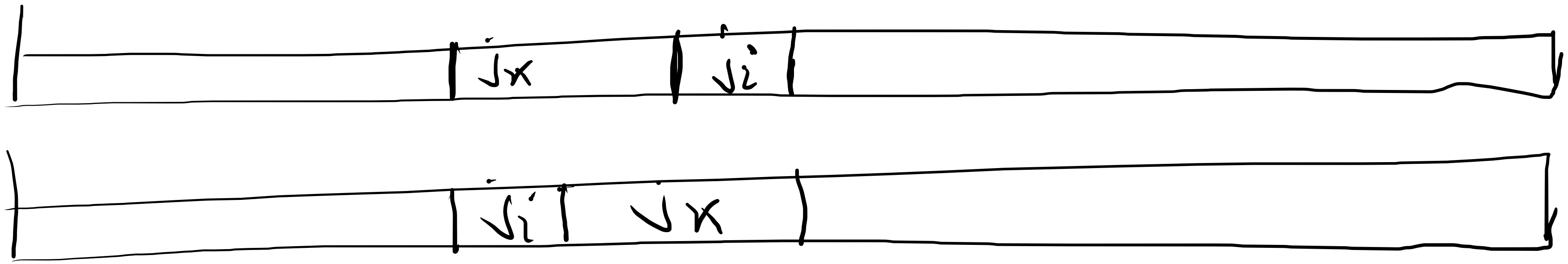
- $d_k > d_l \Rightarrow$ j_k and j_l are inverted jobs.

- $d_k < d_l \Rightarrow d_i < d_k < d_l$

  j_l and j_i are inverted jobs. $\Rightarrow \circ \Leftarrow$

claim If we invert two adjacent inverted jobs j_i and j_K then it reduces the number of inversions by 1 and does not increase the maximum lateness.

Proof



- $L \leftarrow$ ~~be~~ the lateness before exchange
- $L' \leftarrow$ " " after exchange.
- For all other jobs other tan j_i and j_K their lateness remain same.
- For the i-th job j_i.

$$l_i^{new} \leq l_i^{old}$$ as it scheduled befor now.

we want to show, $l_K^{new} \leq l_i^{old}$.

- If the job j_k is late.

$$l_k^{new} = f_k^{new} - d_k \quad \text{def}^n \text{ of lateness}$$

$$= f_i^{old} - d_k \quad \text{as } j_k \text{ finishes now at time } j_i \text{ in old schedule.}$$

$$\leq f_i^{old} - d_i^{old} \quad d_i \leq d_k \text{ by def}^n \text{ of inversion}$$

$$= l_i^{old}$$

lateness of j_k can not be more than the lateness of l_i in the old schedule.

$$\text{current lateness} = \max\{l_1^{new}, l_2^{new}, \ldots, l_k^{new}, l_i^{new}, \ldots, l_n^{new}\}$$

$$\leq \max\{l_i^{old}, l_i^{old}, \ldots, l_i^{old}, l_i^{old}, \ldots, l_n^{old}\}$$

- The algorithm produces a schedule with no inversion and no idle-time
- There is an optimum schedule with no inversion and no idle time.
- All schedule with no inversions and idle-free have the same lateness.

$\Rightarrow$ Gereedy EDF is optimum.