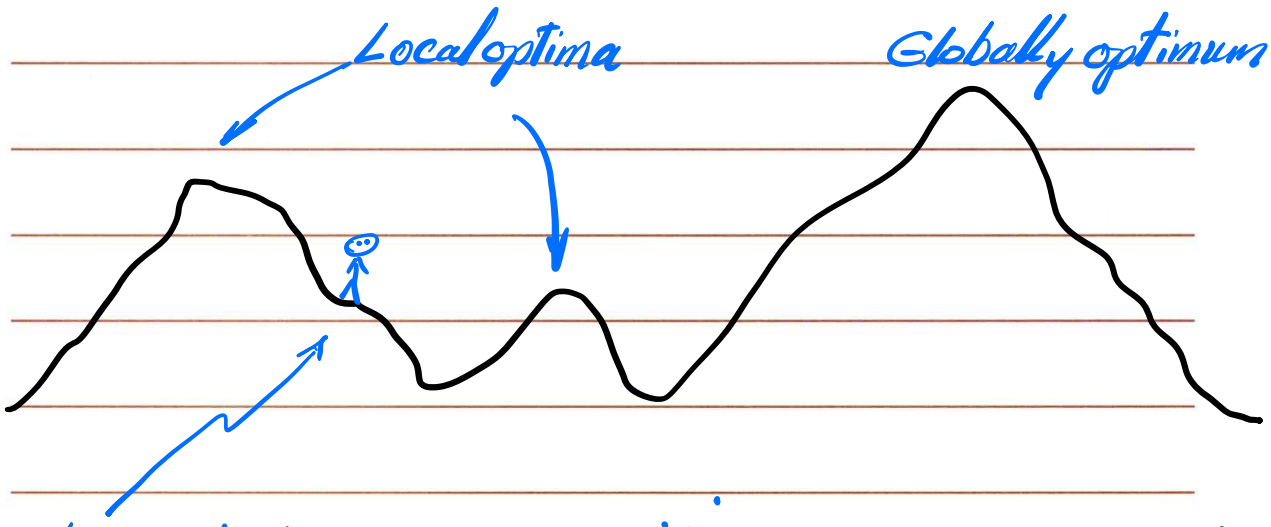
Greedy

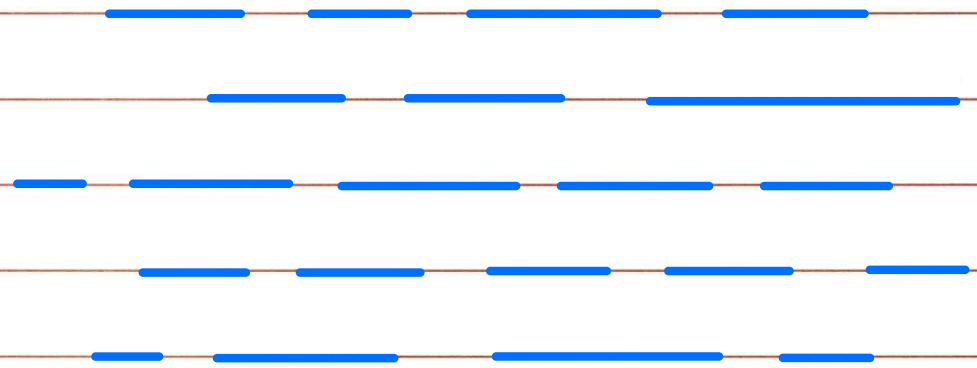


Hiker starting at this position and trying to find the highest peak will find a local optimum solution.

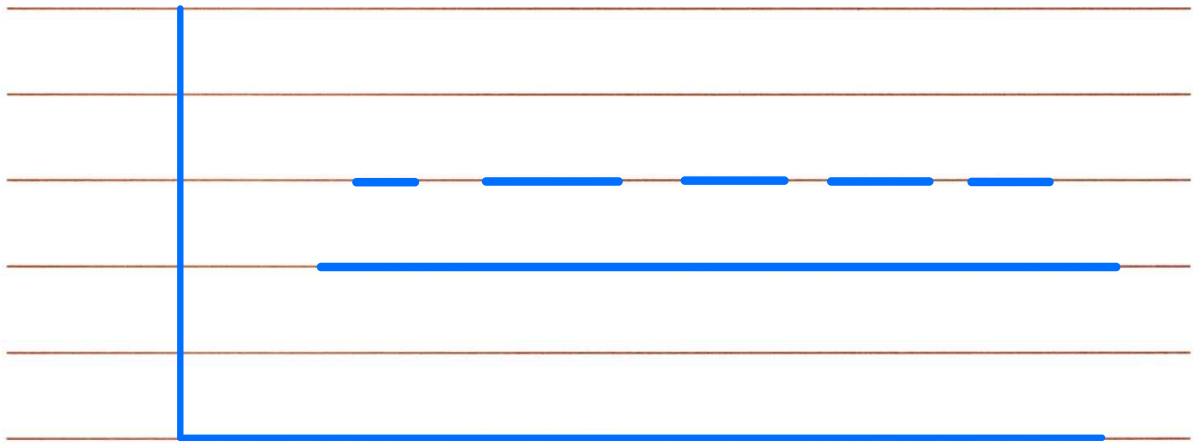
Interval Scheduling Problem

Input: A set of requests $\{1..n\}$, where the i^{th} request starts at $s(i)$ and ends at $f(i)$.

Output: A largest compatible subset of these requests.



Try #1 Earliest Start Time First ✗



Counterexample

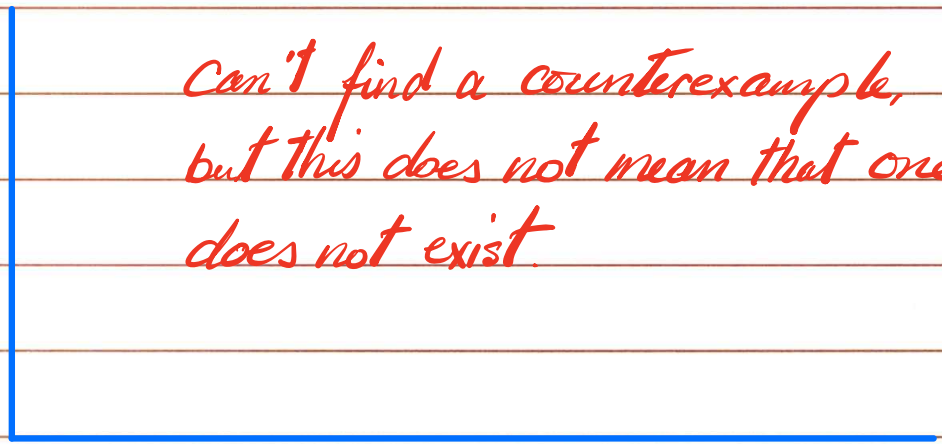
Try #2 Smallest Request First ✗

Counterexample

Try #3 Smallest # of Overlaps First ✗

Counterexample

Try #4 Earliest Finish Time First



can't find a counterexample, but this does not mean that one does not exist.

High Level Solution

Initially R is the complete set of requests and A is empty

While R is not empty

Choose a request $i \in R$ that has the smallest finish time

Add request i to A

Delete all requests from R that are not compatible with i

Endwhile

Return A

Proof of Correctness

1. Prove that A is a compatible set

2. Prove that A is an optimal set

Easy to show #1: Since we always delete all overlapping requests before choosing the next request, we can never end up with overlapping requests in A.

#2: Say A is of size k, and suppose there is an optimal solution O.

Label requests in A: $i_1, i_2, \ldots i_k$

" " " O: $j_1, j_2, \ldots, j_m$

We will first prove that for all indices $1 \leq r \leq k$, we have $f(i_r) \leq f(j_r)$

Proof by mathematical induction

A i_1 ..... i_{r-1} i_r

O j_1 ..... j_{r-1} j_r

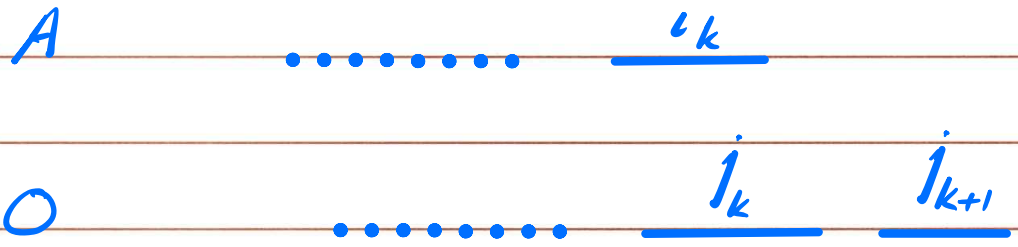
Base Case: i_1 is the request with the earliest finish time, so $f(i_1) \leq f(j_1)$

Inductive hypothesis: We assume that $f(i_{r-1}) \leq f(j_{r-1})$

Inductive Step:
We now need to show that $f(i_r) \leq f(j_r)$.
Since j_{r-1} and j_r are compatible and $f(i_{r-1}) \leq f(j_{r-1})$, then i_{r-1} and j_r are also compatible. So our algorithm can choose either i_r or j_r after i_{r-1}, and since it always picks the one w/ earliest finish time, then i_r must finish no later than j_r.

Having proven that for all indices $1 \leq r \leq k$, $f(i_r) \leq f(j_r)$, we can now prove that $|A| = |O|$

Proof by Contradiction:
We assume that $|O| > |A|$, i.e. the optimal solution contains a request j_{k+1}



Since $f(i_r) \leq f(j_r)$ for all indices $1 \leq r \leq k$, then $f(i_k) \leq f(j_k)$.
Also, since j_{k+1} and j_k are compatible, then i_k and j_{k+1} will also be compatible. So if this request existed our algorithm could have picked it. Contradiction! and

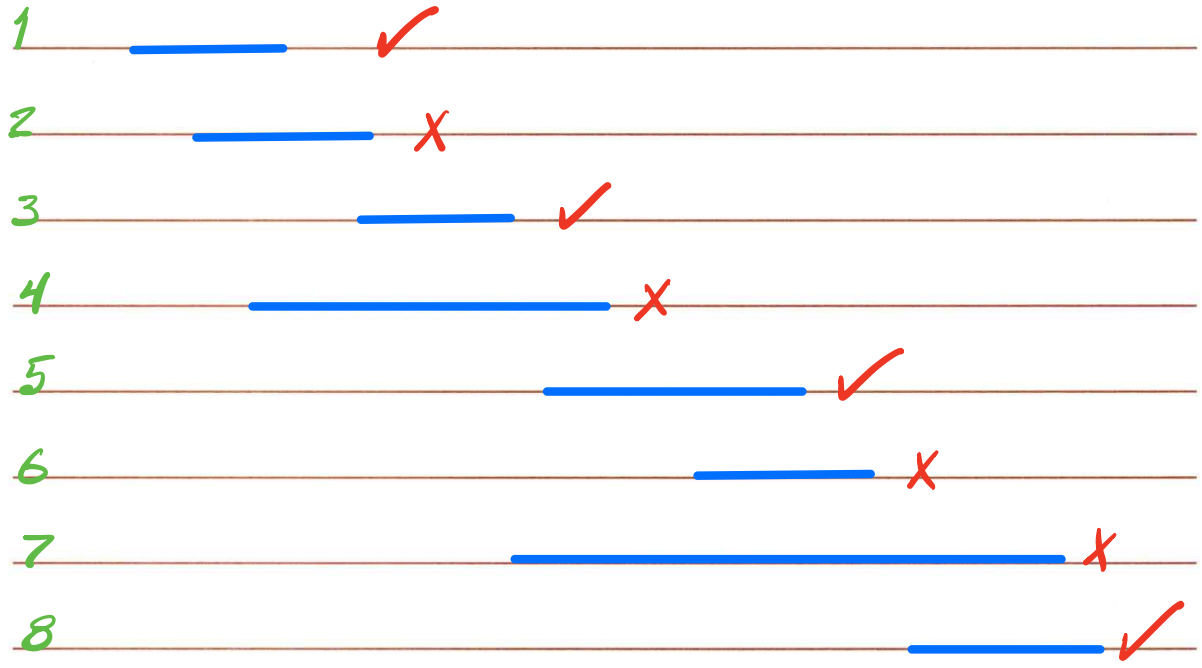
Implementation

$O(n \log n)$ Sort requests in order of finish time and label in this order:

$f(i) \leq f(j)$ where $i < j$

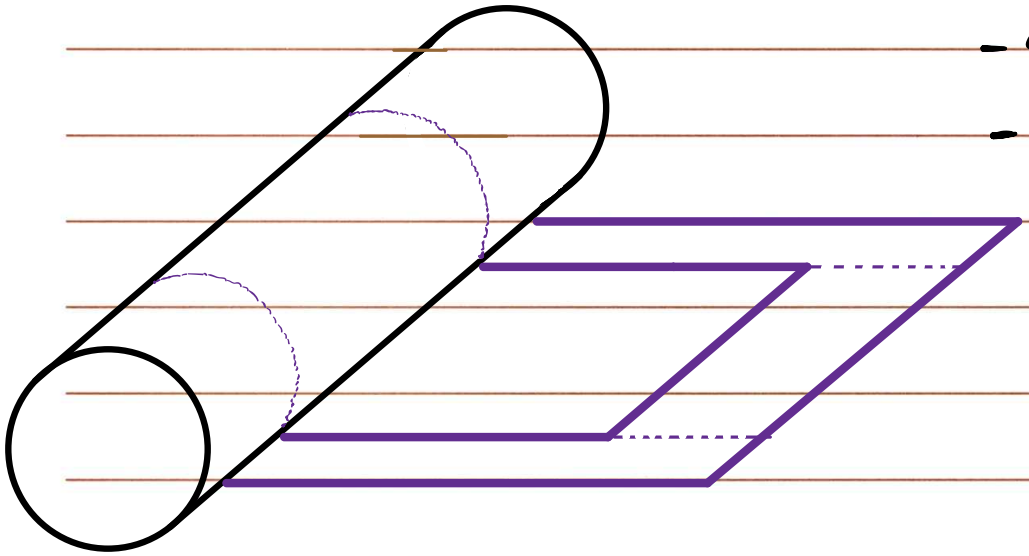
$O(n)$ Select requests in order of increasing $f(i)$, always selecting the first request.
Then iterate through the intervals in this order until reaching the first interval j where $s(j) \geq f(i)$ and then pick j.

Overall Complexity $= O(n \log n)$

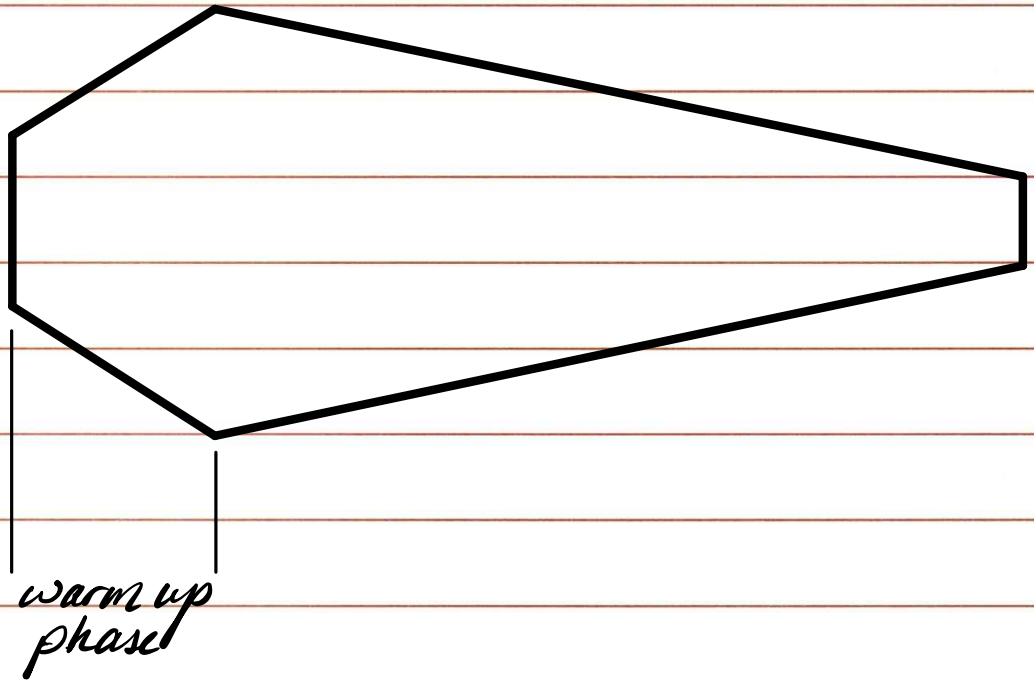


Customer orders specify:

- Quantity
- Width
- Grade



Coffin scheduling



Scheduling to Minimize Lateness

Input: A set of requests $\{1..n\}$, where the i^{th} request has deadline d_i (and time duration t_i.

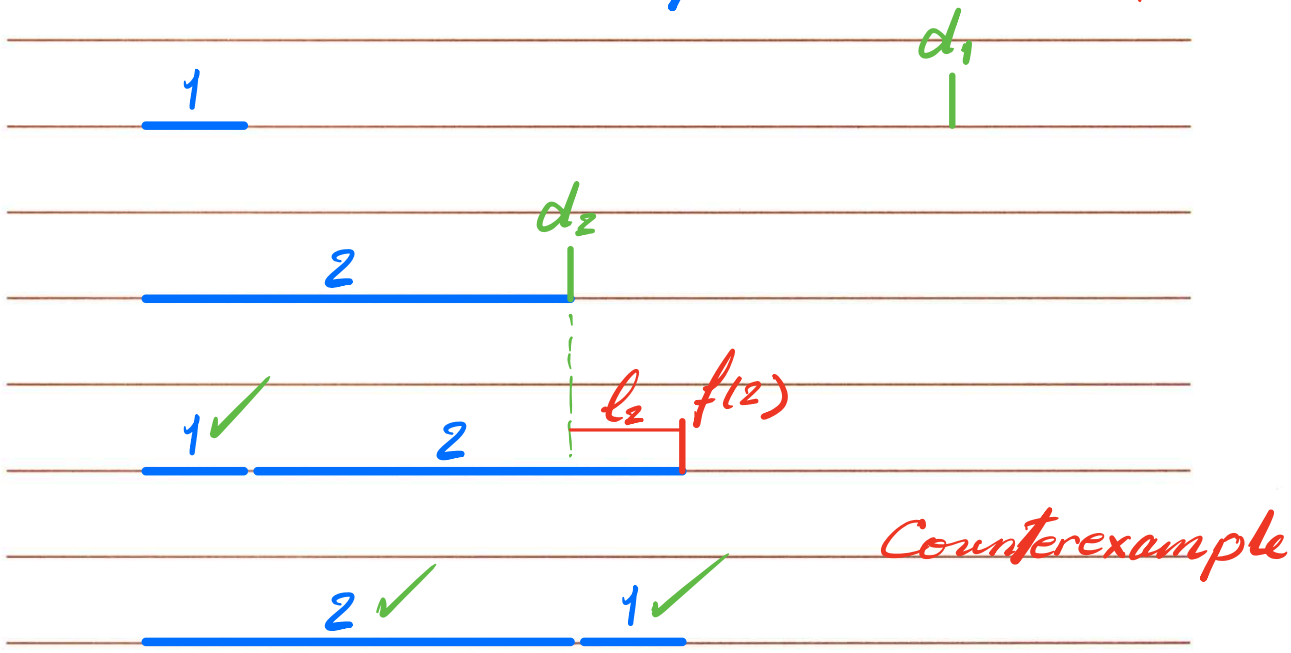
Output: A schedule of the requests that minimizes the maximum lateness. where the lateness for request i is $l_i = f(i) - d_i$

Based on the above objective, which of the following solutions is preferred?

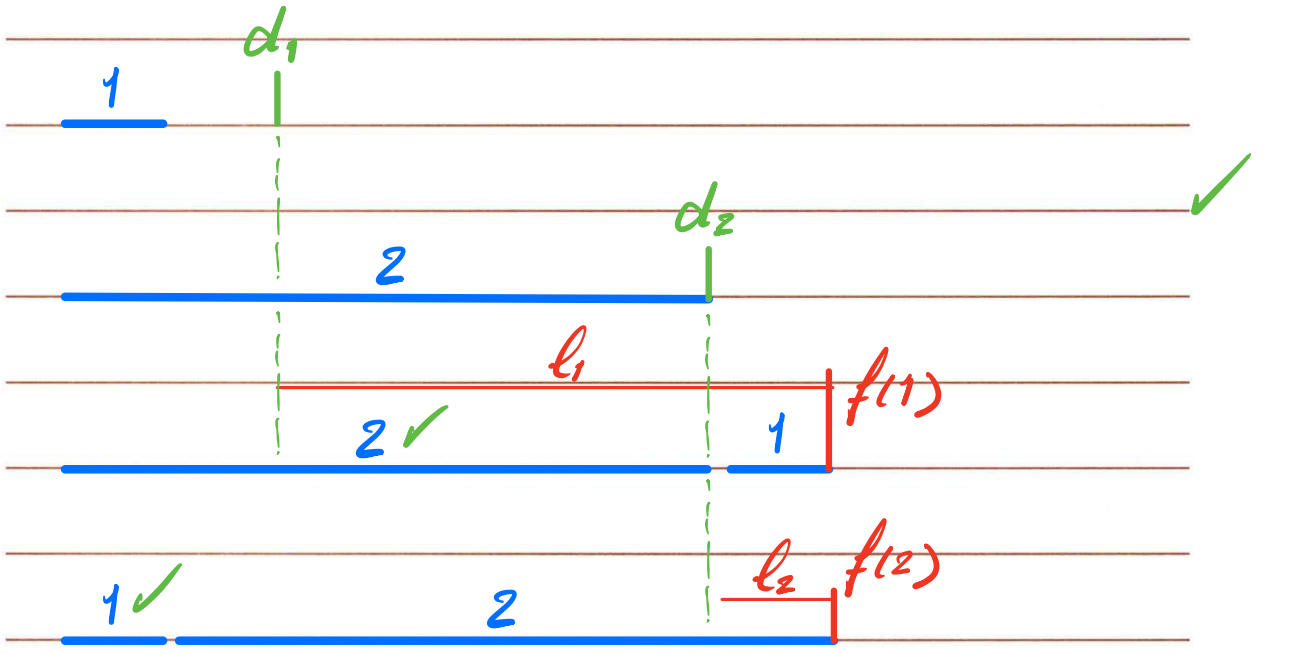
Sol. 1 — job 1 late by 5 hrs; job 2 late by 6 hrs
max. lateness = 6

Sol. 2 — job 1 late by 0 hrs; job 2 late by 7 hrs
max. lateness = 7

Try # 1 Smallest Request First ✗



Try # 1 Shortest Slack First ✗

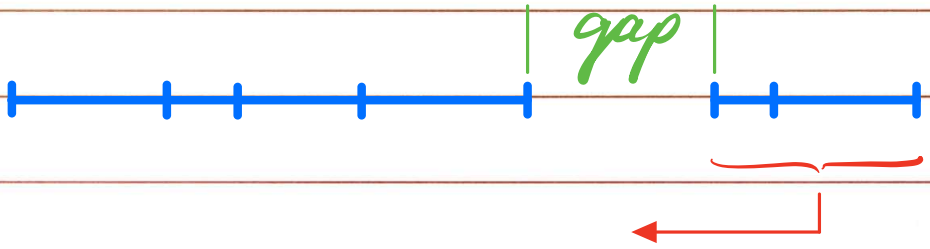


Solution:

Schedule jobs in order of their deadline without any gaps between jobs.

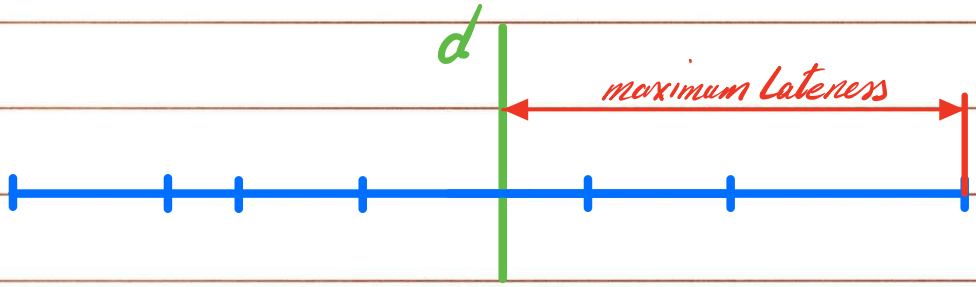
Proof of Correctness:

1- There is an optimal solution with no gaps.



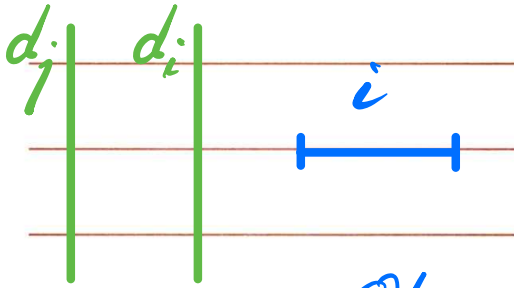
Can always close gaps in an optimal solution to produce an optimal solution with no gaps.

2- Jobs with identical deadlines can be scheduled in any order without affecting maximum lateness.



Maximum lateness will be independent of the order of jobs

3- Def. Schedule A' has an inversion if a job i with deadline d_i is scheduled before job j with an earlier deadline.

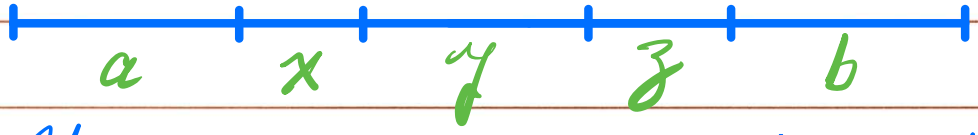


Observation: Our solution contains no inversions.

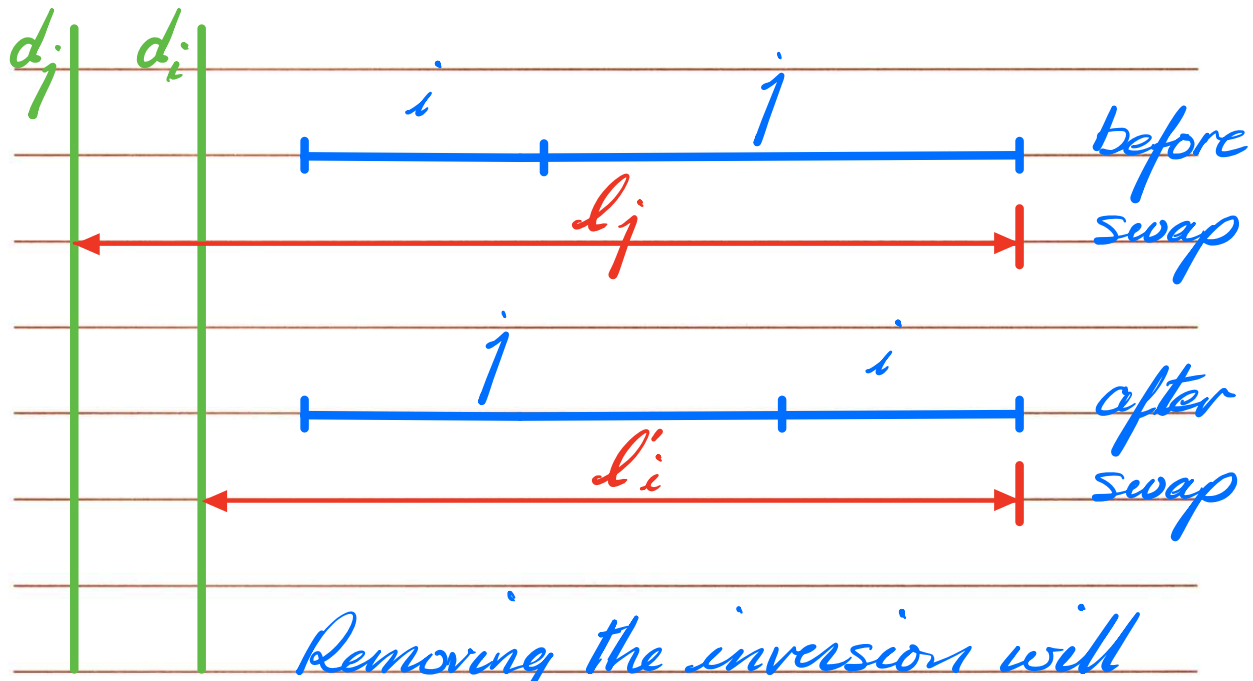
4- All schedules with no inversions and no idle time have the same maximum lateness

If deadlines are distinct, there will only be one such schedule.
And if a subset of jobs have the same deadline, the order of jobs within that subset does not affect their maximum lateness (Fact #2)

5- There is an optimal schedule that has no inversions and no idle time.



If there is an inversion between jobs a and b, we can always find two adjacent jobs between a and b with inversion between them.



Removing the inversion will not increase the maximum lateness

If there is an optimal solution that has inversions, we can eliminate the inversions one by one as shown above until there are no more inversions. This solution will also be optimal.

6_ Proved that there exists an optimal schedule with no inversions and no idle time.

Also proved that all schedules with no inversions and no idle time have the same maximum lateness.

Our greedy algorithm produces one such solution, therefore our solution is also optimal.