10 Scheduling to Minimize Lateness, Divide and Conquer, Closest Points

- Greedy algorithms: scheduling to minimize lateness
- Divide and conquer: merge sort, closest points
- Fast fourier transform

10.1 Scheduling to minimize lateness

Setting: We are given tasks with durations t_i and deadlines d_i.
If we schedule job i for interval $[s_i, f_i]$ with $f_i = s_i + t_i$, its **lateness** is $l_i = \max\{0, f_i - d_i\}$.

Our goal is to schedule all jobs without overlaps to minimize $\max\limits_i l_i$ (trying to minimize the largest l_i only)

Example 10.1
$t_1 = 1$, $d_1 = 2$
$t_2 = 3$, $d_2 = 1$

There are two possible orders we can do things: 1 and then 2, or 2 and then 1.

Order 1, 2: $l_1 = 0$, $l_2 = 3$ - worst case lateness is 3.
Order 2, 1: $l_1 = 2$, $l_2 = 2$ - worst case lateness is 2.

So, we would prefer the order 2, 1.

Possible greedy strategies:

- Earliest deadline d_i first (optimal)
- Shortest duration t_i first
- Increasing "slack" $d_i - t_i$

We will show that sorting by earliest deadline first is optimal.

Definition 10.2
A schedule has an **inversion** if job i comes before job j but $d_i > d_j$.

Lemma 10.3
All schedules with no inversions (and no idle time) have the same max lateness.

Proof. The only flexibility in such a schedule is in the order of jobs with the same deadline.

Any such jobs must be ordered consecutively.

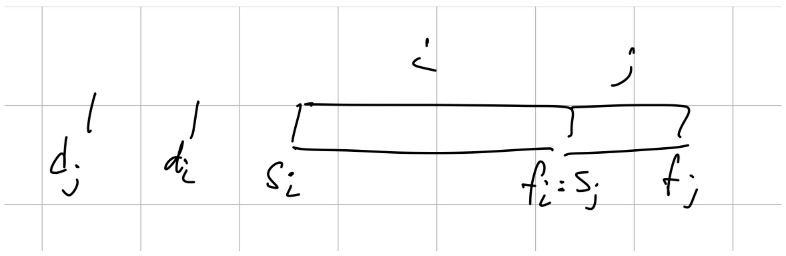
The finishing time of the last of these jobs doesn't dpeend on the ordering. □

Lemma 10.4
There is an optimal schedule with no inversions (and no idle time).

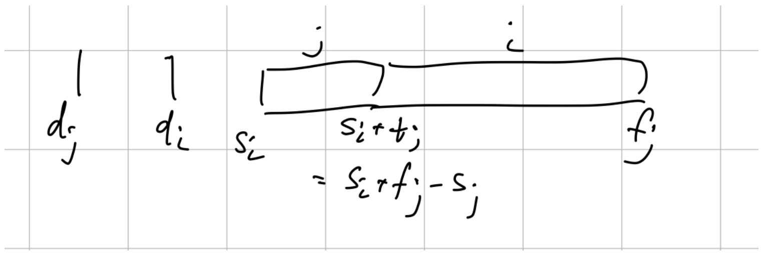
Proof. We'll show how to convert an optimal scedule with inversions into one with none.

If there is an inversion, then there must be jobs i and j that are adjacent (with i before j) with $d_i > d_j$.

Swapping i and j gives a schedule with one less inversion and (we claim) no worse lateness.
Why is the lateness no worse?



Before the swap, the lateness is $f_j - d_j$



After the swap, the lateness is $l = \max\{f_j - d_i, s_i + f_j - s_j - d_j\} = \max\{f_j - d_i, f_j - d_j - (s_j - s_i)\}$.

Clearly, $l \leq f_j - d_j$ since $d_i > d_j$ and $s_j - s_i \geq 0$.

Repeating this process, we eventually get a schedule with no inversions and no worse lateness. ☐

10.2 Divide and conquer

Main idea: Divide original problem into subproblems. Solve these subproblems and merge these solutions to solve the main problem.

Example 10.5
Merge sort. To sort a list of length n:

- Divide in half and sort each half (recursively)
- Merge the two halves in time $O(n)$

Let $T(n)$ be the running time on instances of length n.

We get the recurrence relation $T(n) = 2T(n/2) + O(n)$, and $T(1) = O(1)$

Here we can find that $T(n) = O(n \log n)$.

10.3 Closest points

Task: Given n points in the plane, find the closest pair.

This problem is easy to do in $O(n^2)$ time. Can we do better?