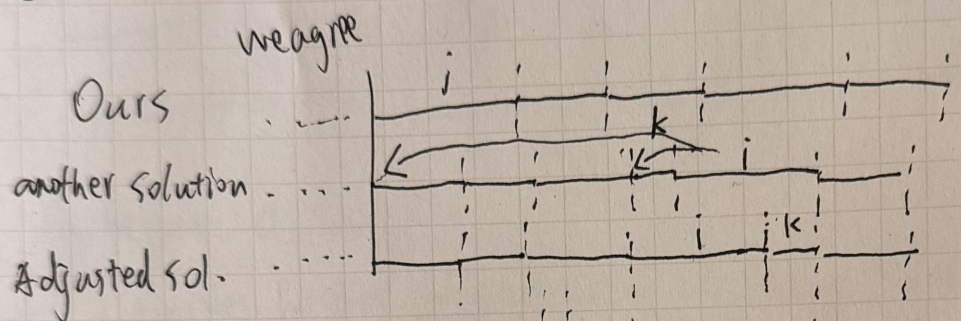


9/23 lecture: more on ^{scheduling} MSTs, dijkstras
scheduling algorithm. cont'd.

Input: Assignments with a designated processing time and deadline.

GOAL: Compute a schedule of assignments that minimized the summed ^{maximal} lateness of assignments.

Algorithm: Sort by deadline.



(assume d_i d_k new)

deadline didn't change. t .

P_k : time of processing k .

Before: $(s + P_k - d_k) + (t - d_i)$.

After: $(t - d_k) + (s + P_i - d_i)$ ^{Before-after} $= P_k - P_i$. which we don't know about

~~For~~ After changing the goal:

For maximal lateness:

If k is late now (in the adjusted), which is ~~bigger~~ at least as big:

i 's lateness before - because $d_i \leq d_k$

k 's lateness now.

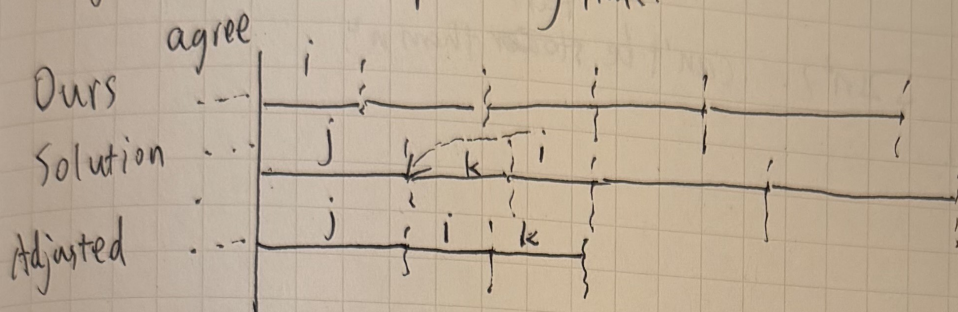
As a result, the maximal lateness can't have increased.

example 2:

A different "summed" objective:

Minimize the sum of completion times. $\rightarrow$ GOAL.

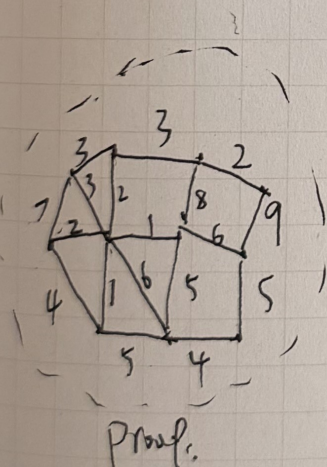
Algorithm: Sort by processing time.



check: 1. didn't touch on agreed part. 2. didn't change processing time other than i and k (swapped).

3. $P_i \leq P_k$. } assignments don't change. feasible.
one step toward ours.

example 3: Minimum Spanning Trees.



Kruskal's: sort the edges.

while an edge remains:

If adding it to the solution forms cycle, delete.

Otherwise, add to solution.

2 types of disagreements: (already agreed on the ^{our} original graph).

1. I took, they didn't: add an edge to form a cycle. delete the biggest (longest) edge in the cycle.

2. They took, I didn't: they have a cycle, so just delete.

Still a spanning tree.