

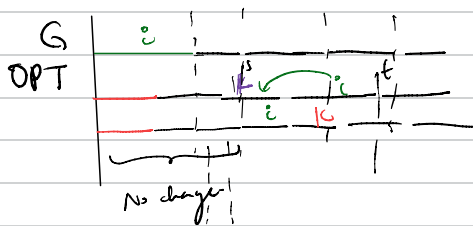
Input: Assignments with a designated processing time and deadline

Goal: Compute a schedule of assignments that minimizes the lateness of the summed lateness of assignments.

Feasibility? No overlap

Optimality! - Minimum lateness! Summed lateness

Alg: Sort by:



Before:

$$k \text{ lateness} = s + p_k - d_k$$

$$i \text{ lateness} = t - d_i$$

$$\text{Total lateness} = s + t + p_k - d_k - d_i$$

After:

$$k \text{ lateness} = t - d_k$$

$$i \text{ lateness} = s + p_i - d_i$$

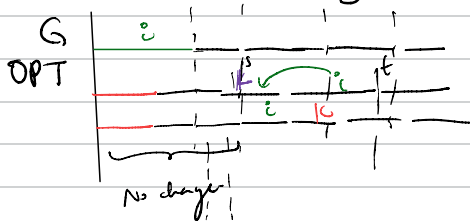
$$\text{Total lateness} = s + t - d_k - d_i + p_i$$

$$\Rightarrow p_k - p_i$$

$\rightarrow$ Lateness

* We have nothing to say about $p_k - p_i \Rightarrow$ Hence the greedy exchange might not work here.

Goal 2! Optimality: Maximal lateness



Know: $d_i < d_k$

* No need to worry about i 's lateness

* k 's lateness could go up!

* If k is late now, then i must have been late

* Which lateness is bigger?

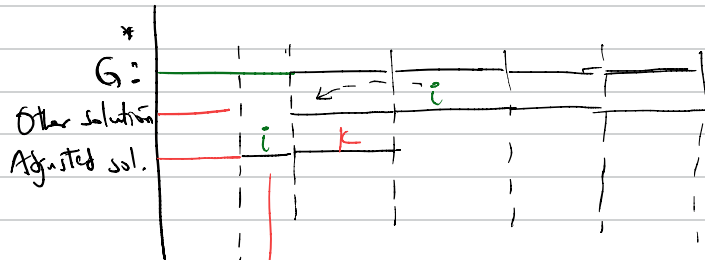
i 's lateness is $\rightarrow$ i 's lateness before
higher k 's lateness now



* The maximal lateness can't have increased! OPTIMALITY preserved

- * New objectives: A different "summed" objective:
 - * Minimize the sum of completion times:

Algorithm is sort by processing time:

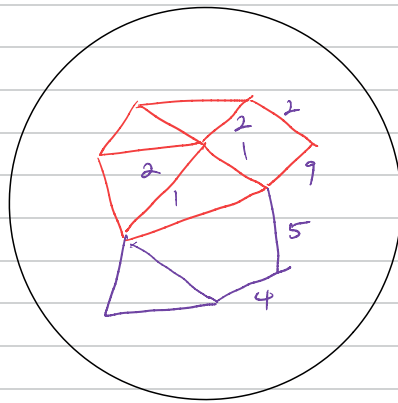


$p_i \leq p_k$ → Feasibility: We didn't delete any assignment
 → Optimality: $p_i \leq p_k$: the sum of completion time couldn't have gone up in the adjusted solution

Minimum Spanning Trees

- * Given some graph,

Goal: Compute the smallest subgraph that connects all the vertices



Feasibility: No cycles

Kruskal's Algorithm:

- * Until you have reached every vertex,
 - while an edge remains, select the SMALLEST EDGE!
 - * delete if it's not feasible
 - * Add to solution if it's optimal.

Disagreements:

- I took, they didn't
- They took, I didn't → They have a cycle → DELETE that node. → Feasible: still a ST

The only reason to skip things! - Forms & cycle

I took, they didn't!