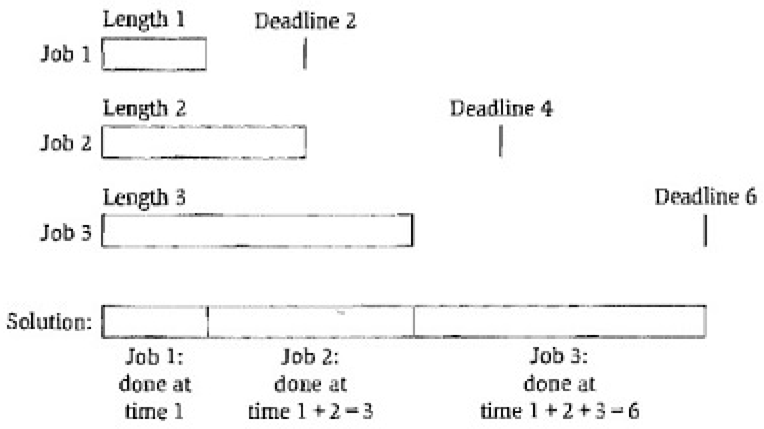
CSCI 270 Lecture 14: Scheduling to Minimize Lateness

We have n tasks, task j has duration t_j and deadline d_j. We choose an order to execute tasks, and we cannot interrupt a process until it is finished. As soon as one task finishes, we start the next one. If a task finishes after its deadline, it is late. We are concerned about which task is the **most** late. We want to minimize how late this task is.

If we start task i at s_i, then task i will finish at $f_i = s_i + t_i$. The lateness is $L_i = \max(0, f_i - d_i)$. $\max_i L_i$ indicates how tardy the **most** late task is, which we want to minimize.



- What greedy criteria might work for this problem?
- Can you find any counter-examples to these algorithms?

Proof:

For a given schedule S, an **inversion** is a pair of jobs i and j where $d_i < d_j$, but $f_j < f_i$. That is, our greedy algorithm says that we should schedule i first, but S scheduled j first.

Our algorithm is the unique schedule with no inversions (assuming no ties).

We want to prove that there is an optimal solution which has no inversions.

Base Case: There are $C(n,2) = \frac{n\cdot(n-1)}{2}$ pairs of jobs, and thus $C(n,2)$ possible inversions. Therefore there is an optimal solution with $\leq C(n,2)$ inversions.

Inductive Hypothesis: Assume there is an optimal solution with $\leq k$ inversions. (Call it OPT).

Now we need to prove there is an optimal solution OPT' with $\leq (k-1)$ inversions.

Note that if you're uncomfortable doing induction "backwards", you could just as well argue how many inversions are "missing". Your base case would be that there is an optimal solution missing at least 0 inversions, you'd assume there is an optimal solution missing $\geq k$ inversions, and then argue there is an optimal solution missing $\geq (k+1)$ inversions.

Case 1: there is a consecutive inversion $(i, i+1)$.

Our algorithm schedules $i+1$ first, but OPT scheduled i first. We want to remove this inversion, so swap their positions to produce OPT'. This will have one less inversion, which will contradict our assumption if we can show that OPT' is still a valid optimal solution.

Prove that OPT' is still valid (this one is easy), and prove that OPT' is still optimal (this one is hard).

- Are the latenesses of any task $< i$ changed from OPT to OPT'?
- Are the latenesses of any task $> i+1$ changed from OPT to OPT'?
- How does the lateness of task $i+1$ change from OPT to OPT'?
- How does the lateness of task $i+1$ in OPT compare to the lateness of task i in OPT'?
- Why is OPT' optimal?

Case 2: there are inversions, but none of them are consecutive.

Let $(i, i+k)$ be the smallest inversion, where $k > 1$. There is a contradiction in this information: where is it?

We have n tasks, task j has duration t_j and deadline d_j. We choose an order to execute tasks, and we cannot interrupt a process until it is finished. As soon as one task finishes, we start the next one. If a task finishes after its deadline, it is late. We are concerned about which task is the **most** late. We want to minimize how late this task is.

If we start task i at s_i, then task i will finish at $f_i = s_i + t_i$. The lateness is $L_i = \max(0, f_i - d_i)$. $\max_i L_i$ indicates how tardy the **most** late task is, which we want to minimize.

Algorithm: Always choose the earliest deadline task

Proof:

For a given schedule S, an **inversion** is a pair of jobs i and j where $d_i < d_j$, but $f_j < f_i$. That is, our greedy algorithm says that we should schedule i first, but S scheduled j first.

Our algorithm is the unique schedule with no inversions (assuming no ties).

We want to prove that there is an optimal solution which has no inversions.

Base Case: There are $C(n,2) = \frac{n \cdot (n-1)}{2}$ pairs of jobs, and thus $C(n,2)$ possible inversions. Therefore there is an optimal solution with $\leq C(n,2)$ inversions.

Inductive Hypothesis: Assume there is an optimal solution with $\leq k$ inversions. (Call it OPT).

Now we need to prove there is an optimal solution OPT' with $\leq (k-1)$ inversions.

Note that if you're uncomfortable doing induction "backwards", you could just as well argue how many inversions are "missing". Your base case would be that there is an optimal solution missing at least 0 inversions, you'd assume there is an optimal solution missing $\geq k$ inversions, and then argue there is an optimal solution missing $\geq (k+1)$ inversions.

Case 1: there is a consecutive inversion $(i, i+1)$.

Our algorithm schedules $i+1$ first, but OPT scheduled i first. We want to remove this inversion, so swap their positions to produce OPT'. This will have one less inversion, which will contradict our assumption if we can show that OPT' is still a valid optimal solution.

Prove that OPT' is still valid (this one is easy), and prove that OPT' is still optimal (this one is hard).

- Are the latenesses of any task $< i$ changed from OPT to OPT'?

- Are the latenesses of any task $> i+1$ changed from OPT to OPT'?

- How does the lateness of task $i+1$ change from OPT to OPT'?

- How does the lateness of task $i+1$ in OPT compare to the lateness of task i in OPT'?

- Why is OPT' optimal?

Case 2: there are inversions, but none of them are consecutive.

Let $(i, i+k)$ be the smallest inversion, where $k > 1$. There is a contradiction in this information: where is it?

Optimal Caching

A cache can store n items. There is a sequence of m requests $d_1, d_2, ..., d_m$ known in advance. If an item is requested which is not in the cache, it must be brought into the cache, resulting in a **cache miss**. You may only bring something into the cache when it is requested. The goal is to minimize the number of cache misses.

$k = 2$, initial contents = ‘ab’

requests: a b c b c a a b

- What greedy criteria might work for this problem?

- Can you find any counter-examples to these algorithms?

- How is this problem different than the standard version of Caching?

Base Case: There is an optimal solution that has the same cache contents as we do through the first 0 requests.

Inductive Hypothesis: There is an optimal solution OPT that has the same cache contents as we do through the first k requests.

Inductive Step: Request $k + 1$ is the first request at which the cache contents of US and OPT differ, and must be a cache miss.

- What must happen at request $k + 1$?