CS 341 – Algorithms

Lecture 8 – Greedy Algorithms

11 June 2021

Today’s Plan

1. Minimizing Total Completion Time
2. Interval Scheduling
3. Interval Coloring

Greedy Algorithms

We will have three lectures about greedy algorithms.

Generally speaking, greedy algorithms work by using simple and/or local rules to make decisions and then commit on them, an analogy to making maximum profit in short term.

Greedy algorithms are usually easy to propose, but may not work or not easy to analyze.

Today we will use scheduling problems as examples to analyze greedy algorithms.

Then we will talk about Hoffman coding, and then Dijkstra's algorithm for single source shortest paths.

We won't study the minimum spanning tree problem, solved by a classical greedy algorithm.

Minimizing Total Completion Time

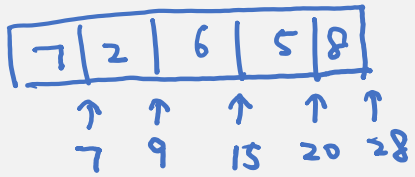
We start with a problem which has a very intuition solution.

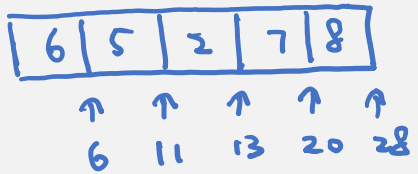
Input: n jobs, with processing times $p_1, p_2, \ldots, p_n$.

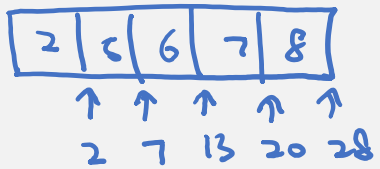
Output: an ordering of the jobs to finish so as to minimize the total completion time,
where the completion time of a job is defined as the time when it is finished.

n = 5

7, 2, 6, 5, 8







total = 60

total completion time = 79

total = 78

Algorithm

Imagine we are in a supermarket.

It is very intuitive that we should process the jobs in increasing order of processing times.

That’s the algorithm.

How to prove that it is correct?

Again just imagine that we are in a supermarket.

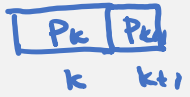
Exchange Argument

We use an exchange argument to prove the correctness of the greedy algorithm.

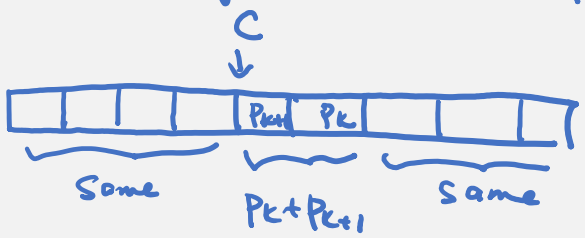
Goal : not non-decreasing (increasing) order of processing times. then not optimal.

not non-decreasing order $\Rightarrow$ $\exists$ an inversion pair

$\Rightarrow$ $\exists$ an adjacent inversion pair



Consider the new ordering



the completion of all jobs are the same. except k-th and $(k+1)$-th

old ordering : $C + p_k$ and $C + p_k + p_{k+1}$. total of two = $2C + 2p_k + p_{k+1}$

new ordering : $C + p_{k+1}$ and $C + p_{k+1} + p_k$, total of two = $2C + 2p_{k+1} + p_k$

since $p_{k+1} < p_k$, then new ordering is better. □

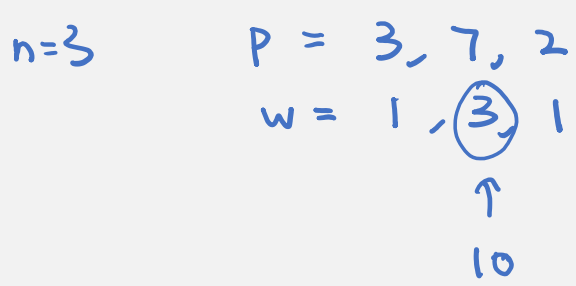
Generalization

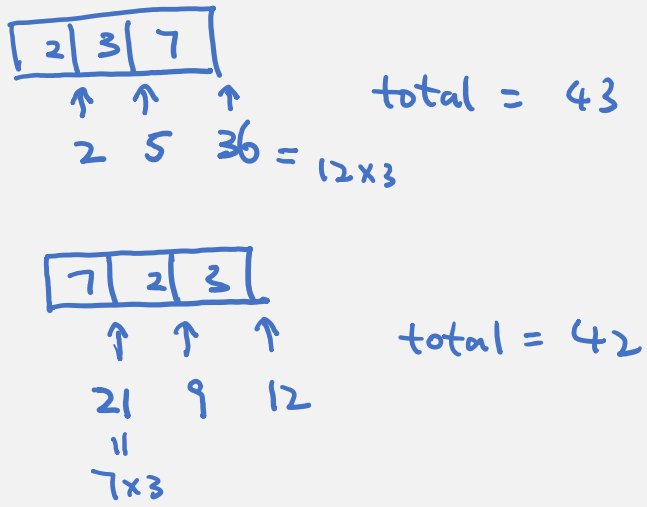
The following is a generalization where the greedy approach still works.

Input: n jobs, each with a processing time p_i and a weight w_i.

Output: an ordering of the jobs to finish so as to minimize the total *weighted* completion time.

where the weighted completion time of a job is defined as its weight times its completion time.





We leave it as a problem for you to think about.

Today’s Plan

1. Minimizing Total Completion Time
2. Interval Scheduling
3. Interval Coloring

Interval Scheduling

Input: n intervals $[s_1, f_1], [s_2, f_2], \ldots, [s_n, f_n]$

Output: a maximum subset of disjoint intervals

Imagine we have a room and we would like to use it to hold as many activities as possible.

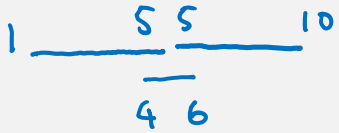
Greedy Algorithms

There are multiple natural greedy algorithms for this problem.

1. Earliest starting time

2. Earliest finishing time ✓

3. Shortest intervals



← Good approximation algorithm?

↙

4. Minimum conflicts (i.e. choose an interval which overlaps with fewest other intervals)

Algorithm

- Sort the intervals so that $f_1 \leq f_2 \leq \cdots \leq f_n$. Initial solution $S = \emptyset$. $O(n \log n)$
- For $1 \leq i \leq n$

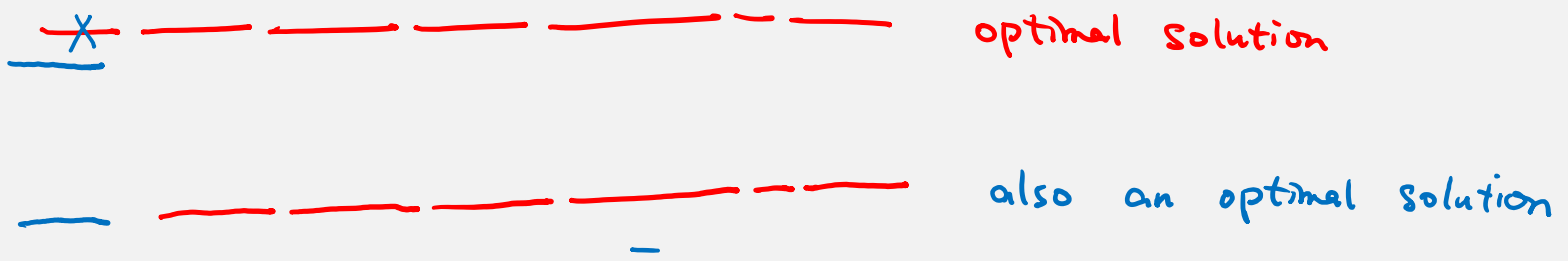
  If interval $[s_i, f_i]$ is disjoint from all intervals in S, then $S := S + i$. $O(1)$ } $O(n)$
- Output S.

time complexity: $O(n \log n)$

Proof of Correctness

We argue that any (optimal) solution would do no worse by using the interval with earliest finishing time.

Claim. There exists an optimal solution with $[s_1, f_1]$ chosen.



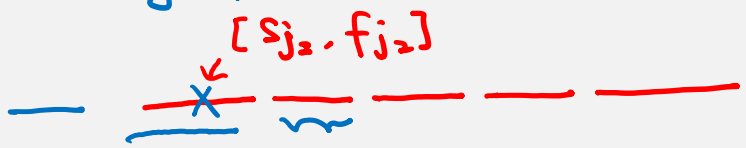
Proof of Correctness

Then, we can extend this exchange argument to show that the greedy solution always "stays ahead".

Lemma. There is an optimal solution using the first k intervals of the greedy solution for any k.

proof

base case : by previous claim



second interval by the greedy solution $[s_{i_2}, f_{i_2}]$

greedy algorithm $\Rightarrow$ $f_{i_2} \le f_{j_2}$

$\Rightarrow$ 

still an optimal solution

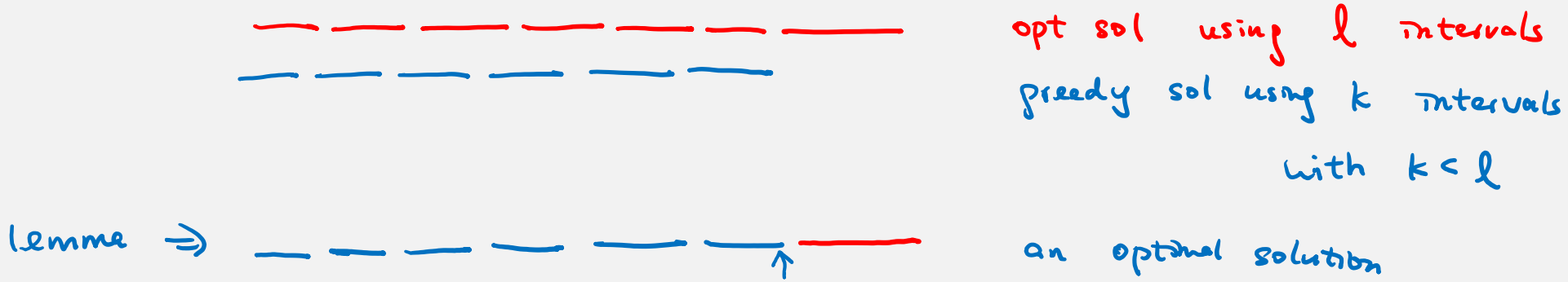
repeat the same argument $f_{i_2} \le f_{j_3}$

□

Proof of Correctness

Lemma. There is an optimal solution using the first k intervals of the greedy solution for any k.

With the lemma, it is easy to finish the proof.



Today’s Plan

1. Minimizing Total Completion Time
2. Interval Scheduling
3. Interval Coloring

Interval Coloring

Input: n intervals $[s_1, f_1], [s_2, f_2], \ldots, [s_n, f_n]$

Output: a coloring of the intervals using the minimum number of colors possible,
so that each interval gets one color and any two overlapping intervals get different colors.

7 colors

6 colors

this is a special case of graph coloring

min # of rooms for all activities

min # of gates for all flights.

① interval with least conflicts
↑
color it last

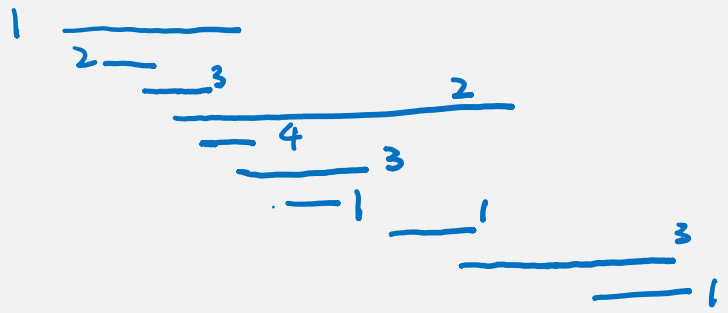
② use the previous algorithm to find max disjoint subset use one color for them

✗

Algorithm

- Sort the intervals so that $s_1 \leq s_2 \leq \cdots \leq s_n$. ~~Initial solution $S = \emptyset$.~~
- For $1 \leq i \leq n$

  Use the minimum available color to color the i-th interval.

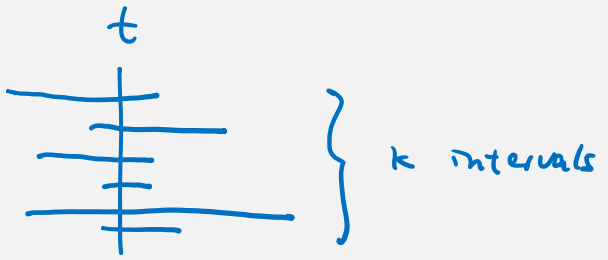


Analysis

To prove the correctness, if our algorithm uses k colors, we need to prove that there are no other ways to color the intervals using less than k colors satisfying the constraints.

How to argue that?

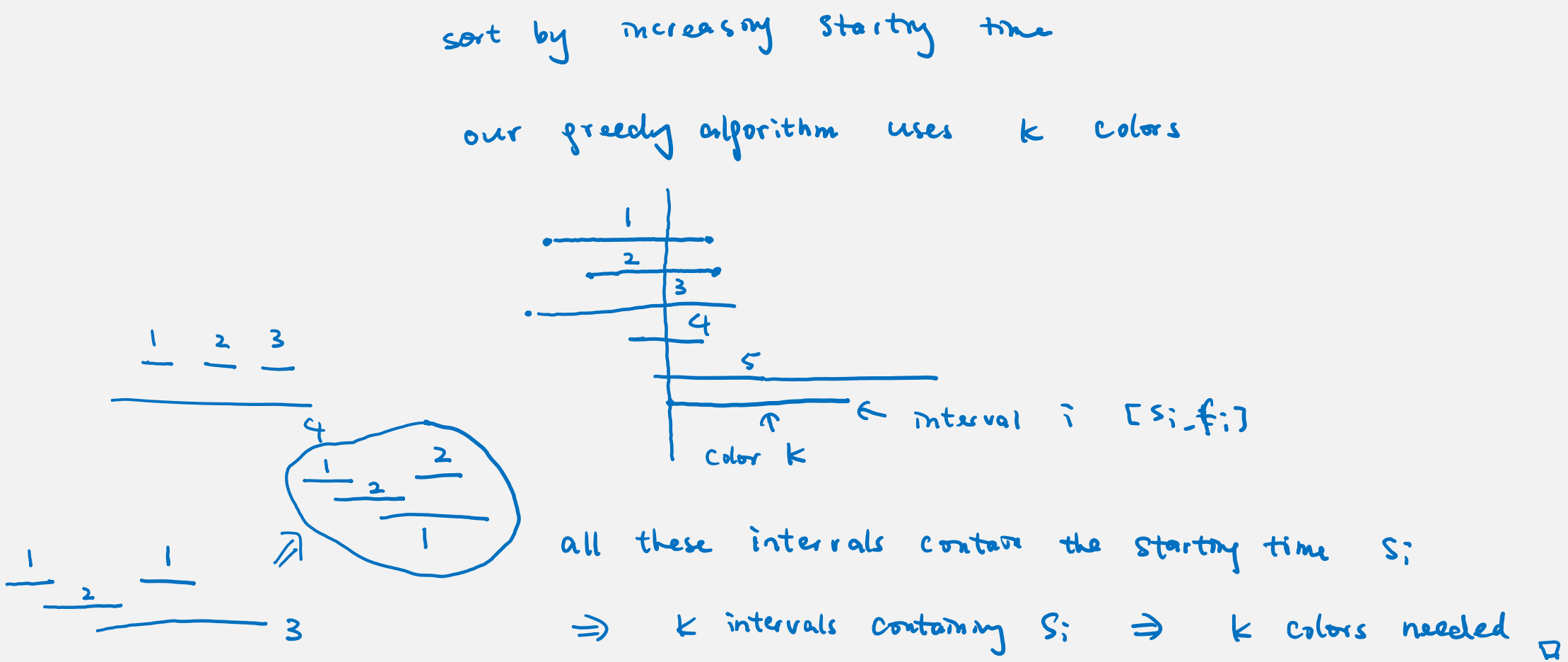
One way is to show that when the algorithm uses k colors, there is a time t such that it is contained in k intervals.



Then these k intervals are pairwise overlapping, and so any algorithm needs to use k colors just for these k intervals.

This would prove that our algorithm is optimal.

Proof of Correctness



Find a counterexample showing that using an arbitrary ordering of the intervals would not work.