

CSC226, Fall 2023

Lab 6: Greedy Scheduling

Ali Mortazavi

October 26, 2023

1 Greedy Algorithm

In this lab session, we want to learn:

“how to prove a greedy algorithm can(/cannot) always find the optimal solution.”
We want to consider the problem of Interval Scheduling.

2 Interval Scheduling

- We have a set of n jobs j_1, \dots, j_n . (Notice that jobs are not ordered.)
- Each job j starts at s_j and finishes at f_j (Obviously $s_j < f_j$)
- Two jobs are *compatible* if they don't overlap
- The goal is to find the **maximum subset of mutually compatible jobs**.

Question 1. Try to formally define compatibility for jobs using math notations: consider job i with starting time s_i and finish time f_i , and job j with s_j and f_j respectively.
Job i and j are compatible if and only if _____

3 Trivial Algorithm

One way to solve this problem is to go over all possible subsets of jobs and check if they're all mutually compatible, and among those compatible sets return the largest subset.

The trivial algorithm has a bad time complexity. In fact:

Question 2. If we have a set A of size n , then the number of subsets of A is: _____

As you see, when we have n jobs, we need to go over 2^n subsets and check whether they're compatible or not. For slightly large n (like $n \geq 100$), it's impossible to implement it in practice.

4 Greedy Algorithm

Now we want to see if there is a greedy algorithm to solve this problem both “optimally” and efficiently.

- Optimally: the algorithm finds the optimal solution which is a subset of jobs with the maximum size, where all jobs are mutually compatible.
- Efficiently: we would like to find an algorithm with a polynomial running time.

The idea of greedy: Pick jobs in each iteration based on some local criteria.
But which local criteria?

4.1 Finding the right greedy approach

Here are some greedy criteria:

1. In each iteration, pick the job j that has the earliest start time s_j
2. In each iteration, pick the job j that has the shortest time interval $f_j - s_j$
3. In each iteration, pick the job j that has the earliest finish time f_j

5 Proving(/Disproving) the optimality of the solution

Here we will show that using the earliest-start-time-first greedy algorithm and shortest-time-interval algorithm would **not** lead to the optimal solution.

Usually, when we want to show an algorithm doesn't work, we use a counter-example to do so.

Question 3. Find a counter-example where “earliest-start-time-first” greedy does not lead to the optimal solution.

Hint: You need to find a set of jobs where if we use the “earliest-start-time-first” algorithm, then we get a sub-optimal solution.

Question 4. Find a counter-example where “shortest-time-interval” greedy does not lead to the optimal solution.

However, we will show that the “earliest-finish-time-first” Algorithm is optimal and also can be implemented efficiently.

Question 5. Prove that the earliest-finish-time-first algorithm is optimal.