Chapter 1

Greedy Algorithms and Exchange Argument

1.1 Greedy Algorithms

Definition: Greedy Algorithm

Choosing the best option at the current step. There is no real "formal" definition of a greedy algorithm.

Example.

For the Travelling Salesperson Problem, we would go to the clostest location at every step. This would be suboptimal (except in special cases) but greedy.

Problem: Scheduling

Given: n jobs. Job i has runtime t_i and deadline d_i
Goal: Form a schedule to minimize the maximum lateness $L = \max l_i$
This problem can be thought of as assembly line jobs or conference meetings, one cannot start without the last job ending.

After you define a problem, don't just try to solve it immediately. Instead think and play around with it. This will allow you to find edgecases and also counterexamples.

We can define the finish time of the job in our schedule to be $f(i)$ and $l(i)$ to be $f(i) - d_i$.

Example.

We have two possible schedules. One causes all jobs to be late by 99 minutes, while the other causes all jobs to be on time except for one that is 100 minutes late.

Although we may think that the second is a better option due to the fact that most jobs are in time, our objective in the algorithm is only to minimize the worst lateness (Egalitarian), which will it cause it to choose the first option.

Possible Greedy Algorithms

1. Shortest job first (sort by t_i)
2. Earliest job first (sort by t_i)
3. Smallest slack time first (sort by $d_i - t_i$)

It is easy to come up with greedy algorithms, but it is hard to tell/prove if one is correct.
We can create contradictions for two of the algorithms.

1. Two jobs, one with $t = 1$ and $d = 3$, and another with $t = 2$ and $d = 2$.
2. Correct algorithm.
3. Two jobs, one with $t = 100$ and $d = 100$, and another with $t = 1$ and $d = 2$.

1.2 Proving the Algorithm

First, we rename jobs in order of deadline so that

$$d_1 \leq d_2 \leq d_3 \leq ... \leq d_n.$$

Claim: Maximum lateness stays the same no matter how we order jobs with the same deadline

Take an arbituary number of jobs with the same deadline d. If maximum lateness happens before or after d, the max lateness will not hange. If maximum lateness is equal to d, the rightmost task will be the max lateness and therefore will not change no matter the order due to the fact that the times of the jobs that are d deadline being constant.

We can now start to prove using the exchange argument.

Definition: Exchange Argument

Consider an optimal solution which is most similar to the Greedy Solution. We can use proof by cases to compare OPT (the optimal solution) and the Greedy Solution.

1. OPT = Greedy
   In this case, by reiteration the Greedy Solution is the optimal solution.
2. We try to change OPT to be more like the Greedy solution without making it worse (but still keep it optimal).
   This is a contradiction because we cannot make the OPT any more similar to the greedy argument.

Now we can use this on the scheduling problem.

Definition: Inversion

Jobs such that i is before j in the schedule but $d_j < d_i$ (basically out of order from Greedy).

A Greedy schedule would have zero inversions (no differences from itself). How similar to the Greedy schedule is how few inversion exist.

Lemma

If an inversion exists, there exists a pair of adjacent jobs which are inverted.

Proof. We can prove this by contradiction. We have jobs i and j that are inverted. We have $k_1, k_2, ...k_n$ jobs between the two. We can then say that

$$d_j < d_i < d_{k_1} < d_{k_n}$$

Because of this, we can say that j and k_n are inversions, which are adjacent. □

Now we can apply the Exchange argument. Consider an optimal schedule which has fewest inversions.

1. OPT has 0 inversions
2. We can decrease the number of inversions in OPT to make it more similar to greedy.

We can then make a claim for case 2 that

Claim

Swapping adjacent inverted jobs in OPT does not increase max lateness.

By this claim, case 2 is a contradiction because the optimal schedule we have has the fewest inversions. Therefore, if we prove this claim, only case 1 could be possible and OPT = Greedy.

Proof. We have that $d_i < d_j$. We start with j before i with j starting at time F. We then swap i and j.

$$l_i = F + t_j + t_i - d_i$$

$$l_j = F + t_j - d_j$$

which turns into

$$l'_j = F + t_i + t_j - d_j$$

$$l'_i = F + t_i - d_i$$

$$d_i < d_j \rightarrow l'_j < l'_i \leq L_{max}$$

□

Therefore, the greedy algorithm by earliest deadline is the optimal solution.