1 Consider a problem instance that has three jobs with $l_1 = 1, l_2 = 2$ and $l_3 = 3$, and suppose they are scheduled in this order. What are the completion times of the three jobs in this schedule?

1. 1,2 and 3
2. 3, 5, and 6
3. 1, 3, and 6
4. 1, 4, and 6

(3) **is the correct answer.**

$$\begin{aligned}\sum_{i=1}^{1} l_i &= 1\\ \sum_{i=1}^{2} l_i &= 3\\ \sum_{i=1}^{3} l_i &= 6\end{aligned}$$

2 (1) If all jobs lengths are identical, should we schedule smaller or larger-weight jobs earlier? (2) If all job weights are identical, should we schedule shorter or longer jobs earlier?

1. larger/shorter
2. smaller/shorter
3. larger/longer
4. smaller/longer

(1) is correct. Our goal is to find the schedule σ such that it minimizes the completion time, that is $\overset{min}{\sigma} \sum_{j=1}^{n} w_j \cdot C_j(\sigma)$. We see that in the case that all job lengths are identical and we should schedule larger jobs first, the reason being the completion time of any job gets worst as time progresses so we should do the highest cost jobs first. In the case that all the weights are the same should do the shortest job first since we minimize the waiting for any other jobs to start.

3 What is the sum of weighted completion times in the schedules output by the *GreedyDiff* and *GreedyRatio* algorithms, respectively?

1. 22 and 23
2. 23 and 22
3. 17 and 17
4. 17 and 11

. (2) is correct. To see why we look at the data set supplied, that is we have a job j_1 with length $l_1 = 5$ and weight $w_1 = 3$ respectively and j_2 with length $l_2 = 2$ and weight $w_2 = 1$ respectively. We see that the first job j_1 has a bigger ratio than j_2 ($\frac{3}{5} > \frac{1}{2}$) while the second job has a bigger difference than job one ($-1 > -2$). We compute both weighted completion times. The weighted completion time of the greedy algorithm based on ratios is $3 \cdot 5 + 1 \cdot 7 = 22$, while the completion time of the greedy algorithm based on differences is $2 + 7 * \cdot 3 = 23$.

4 Prove that every schedule $\hat{\sigma}$ different from the greedy schedule σ has at least one consecutive inversion.

We proof the contrapositive that is we prove that if a schedule $\hat{\sigma}$ has no consecutive inversion then it must be the same as the greedy schedule σ. If we have no schedule with consecutive inversion then that means every job after it is atleast bigger by 1. Since we have n jobs and the max-possible index is n there cannot be a jump of 2 or more between indices of consecutive jobs. Hence the jobs will be schedule as $j_1, j_2, \ldots, j_n$ which is exactly the same as the greedy algorithm.

5 What effect does the exchange have on the completion item of: (i) a job other than i or j; (ii) the job i; and (iii) the job j? where $i > j$

1. (i) Not enough information to answer; (ii) goes up; (iii) goes down.
2. (i) Not enough information to answer; (ii) goes down; (iii) goes up.
3. (i) unaffected; (ii) goes up; (iii) goes down.
4. (i) unaffected; (ii) goes down; (iii) goes up.

(3) is correct. Any other job before i and j is unaffected since their completion times don't depend on either i or j. jobs after i and j will also not be affected because regardless of which pairing is chosen all other jobs afterwards will have to wait for both jobs to be completed. i cost must go up because now it has to wait for j to complete before it can be processed and j cost must go down because it no longer has to wait for i to complete.

6 Prove that for every set of positive job weights $w_1, w_2, \ldots, w_n$ and positive job lengths $l_1, l_2, \ldots, l_n$, the *GreedyRatio* algorithm outputs a shedule with the minimum-possible sum of weighted completition times in the case where there is no ties between ratios, that is $\frac{w_i}{l_i} \neq \frac{w_j}{l_j}$ whenever $i \neq j$.

We first assume that the jobs are index in nonincreasing order of weight-length ratio:

$$\frac{w_1}{l_1} \quad > \quad \frac{w_2}{l_2} > \ldots > \frac{w_n}{l_n}$$

From the problem statement we assume no ties thus the strict inequality is valid. We proceed by contradiction, that is we assume that the *GreedyRatio* algorithm, σ, does not produce an optimal solution but some algorithm, σ^*, does. By problem (4) we see that since the optimal algorithm σ^* produces a different output than the *GreedyRatio* σ then the σ^* algorithm must have at least one consecutive inversion. We also see by problem (5) that exchanging the jobs index at i and j where these jobs are a consecutive inverse will only improve the algorithm. Hence we could improve upon the optimal algorithm by exchanging the jobs index at i and j and keeping all other jobs the same, we denote this new schedule σ^{**}. But σ^* was supposed to be the optimal scheduling, a contradiction. Hence the *GreedyRatio* algorithm produces an optimal sequence of job scheduling.

7 An *inversion* in a schedule is a pair k, m of jobs with $k < m$ and m processed before k. Suppose σ_1 is a schedule with a consecutive inversion i, j with $i > j$, and obtain σ_2 from σ_1 by reversing the order of i and j. How does the number of inversion in σ_2 compare to that in σ_1?

1. σ_2 has one fewer inversion than σ_1.
2. σ_2 has the same number of inversion as σ_1.
3. σ_2 has one more inversion than σ_1.
4. None of the other answers are correct.

(1) is the correct answer. (2) and (3) are incorrect because if we fix an inversion we would expect to have one fewer inversion than before. Hence σ_2 must have one fewer inversion than σ_1.

8 Prove the general case of the correctness of the *GreedyRatio* algorithm when ties can exist.

We first assume that the jobs are index in nonincreasing order of weight-length ratio:

$$\frac{w_1}{l_1} \quad \geq \quad \frac{w_2}{l_2} \geq \ldots \geq \frac{w_n}{l_n}$$

We can reuse much of the work as the previous problem (6) but instead of proceeding with contradiction we now proceed directly. Let σ be the *GreedyRatio* algorithm and σ^* be an arbitrary competing schedule σ^* where $\sigma \neq \sigma^*$, optimal or otherwise. By problem (4) σ^* must have an consecutive inverse. By problem (7) by doing an exchange with indices that produce a consecutive inverse we either make a solution better or stay the same, in order words doing exchanges makes the solution no worse than before only possibly better. Since there are a finite number of consecutive inverse we can continously do exchanges until this process terminates. This algorithm terminates and shares the same schedule as σ. Since σ^* was arbitrary this holds for all schedulings hence σ is optimal.

9 You are given as input n jobs, each with a length l_j and a deadline d_j. Define the *lateness* $\lambda_j(\sigma)$ of a job j in a schedule σ as the difference $C_j(\sigma) - d_j$ between job's completion time and deadline, or as 0 if $C_j(\sigma) \leq d_j$. This problem considers the objective of minimizing the maximum lateness, $max_{j=1}^{n}\lambda_j(\sigma)$. Which of the following greedy algorithms produces a schedule that minimizes the maximum lateness?

1. Schedule the jobs in increasing order of deadline d_j.
2. Schedule the jobs in increasing order of processing time p_j
3. Schedule the jobs in increasing order of product $d_j \cdot p_j$.
4. None of the other answers are correct

(1) is correct. Let σ_1 be the schedule we produce by the algorithm where we schedule jobs in increasing order of deadline d_j, σ_2 be the schedule we produce by the algorithm where the jobs in increasing order of processing time p_j andσ_3 be the schedule produce by the algorithm where we schedule the jobs in increasing order of product $d_j \cdot p_j$. (2) and (3) can be shown to be wrong given the dataset

$$\begin{array}{ccc} l_1 = 1 & l_2 = 2 & l_3 = 3 \\ d_1 = 5 & d_2 = 3 & d_3 = 1 \end{array}$$

$$\begin{aligned} max_{j=1}^{n}\lambda_j(\sigma_1) &= 2 \\ max_{j=1}^{n}\lambda_j(\sigma_2) &= 5 \\ max_{j=1}^{n}\lambda_j(\sigma_3) &= 3 \end{aligned}$$

Which shows that σ_2 and σ_3 could not be optimal schedules. We still must show the correctness of the greedy algorithm schedule σ_1. We first assume that there are no ties that is, $d_i \neq d_j$ whenever $i \neq j$. We also also assume that the jobs are ordered in increasing order of deadlines:

$$d_1 \quad < d_2 < \ldots < d_n$$

Suppose an arbitrary schedule σ^*, that is possibly optimal or not. Since $\sigma_1 \neq \sigma^*$ we can use a similar argument to problem (4) to show that σ^* must at least have one consecutive inversion. Therefore in the algorithm σ^* there must exist indices i, j such that $i > j$ and the job at index i is processed before the job at index j. If we do an exchange of the jobs all jobs before i and j are not affected and all jobs after i and j are not affected. We show that fixing an inversion does not increase the maximum lateness

$$\begin{aligned} \lambda_{j'}(\sigma^*) &= C_{j'}(\sigma^*) - d_j \\ &= C_i(\sigma^*) - d_j \\ &\leq C_i(\sigma^*) - d_i \\ &\leq \lambda_i(\sigma^*) \end{aligned}$$

Since the number of inversions are finite the process will enventually terminate and the schedule will have no inversions and by (4) will be equal to the greedy algorithm schedule. Since the schedule σ^* was arbitrary this result holds for all schedules and we conclude that greedy algorithm is optimal.

10 Continuing Problem 7, consider instead the objective of minimizing the *total* lateness, $\sum_{j=1}^{n} \lambda_j(\sigma)$. Which of the following greedy algorithms produces a schedule that minimizes the total lateness?

1. Schedule the jobs in increasing order of deadline d_j.
2. Schedule the jobs in increasing order of processing time p_j.
3. Schedule the jobs in increasing order of the product $d_j \cdot p_j$.
4. None of the other answers are correct.

11 You are given as input n jobs, each with a start time s_j and a finish time t_j. Two jobs *conflict* if they overlap in time-if one of them starts between the start and finish times of the other, In this problem, the goal is to select a maximum-size subset of jobs that have no conflicts. The plan it to design an iterative greedy algorithm, in each iteration, irrevocably adds a new job j to the solution-so-far and removes the future consideration all jobs that conlfict with j. Which of the following greedy algorithms is guaranteed to compute an optimal solution?

1. At each iteration, choose the remaining job with the earliest finish time.
2. At each iteration, choose the remaining job with the earliest start time.
3. At each iteration, choose the remaining job the requires the least time.
4. At each iteration, choose the remaining job with the fewest number of conflicts with other remaining jobs.