

**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.**

$$\sum_{i=1}^1 l_i = 1$$

$$\sum_{i=1}^2 l_i = 3$$

$$\sum_{i=1}^3 l_i = 6$$

**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  $\sigma$  such that it minimizes the completion time, that is  $\min_{\sigma} \sum_{j=1}^n w_j \cdot C_j(\sigma)$ . 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  $\sigma$  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  $\sigma$ . 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  $j_1 < j_2 < \dots < j_n$  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  $w_1, w_2, \dots, w_n$  and positive job lengths  $l_1, l_2, \dots, l_n$ , the *GreedyRatio* algorithm outputs a schedule with the minimum-possible sum of weighted completion 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  $i \neq j$ .

**7** An *inversion* in a schedule is a pair  $k, m$  of jobs with  $k < m$  and  $m$  processed before  $k$ . Suppose  $\sigma_1$  is a schedule with a consecutive inversion  $i, j$  with  $i > j$ , and obtain  $\sigma_2$  from  $\sigma_1$  by reversing the order of  $i$  and  $j$ . How does the number of inversion in  $\sigma_2$  compare to that in  $\sigma_1$ ?

1.  $\sigma_2$  has one fewer inversion than  $\sigma_1$ .
2.  $\sigma_2$  has the same number of inversion as  $\sigma_1$ .
3.  $\sigma_2$  has one more inversion than  $\sigma_1$ .
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  $\sigma_2$  must have one fewer inversion than  $\sigma_1$ .

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

**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  $\sigma$  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

**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 is 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 conflict 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.