

It is clear that the working time *for the super-computer* does not depend on the ordering of jobs. Thus we can not change the time when the last job hands off to a PC. It is intuitively clear that the last job in our schedule should have the shortest *finishing* time.

This informal reasoning suggests that the following greedy schedule should be the optimal one.

**Schedule  $G$ :**

Run jobs in the order of decreasing finishing time  $f_i$ .

Now we show that  $G$  is actually the optimal schedule, using an exchange argument. We will show that for any given schedule  $S \neq G$ , we can repeatedly swap adjacent jobs so as to convert  $S$  into  $G$  without increasing the completion time.

Consider any schedule  $S$ , and suppose it does not use the order of  $G$ . Then this schedule must contain two jobs  $J_k$  and  $J_l$  so that  $J_l$  runs directly after  $J_k$ , but the finishing time for the first job is less than the finishing time for the second one, i.e.  $f_k < f_l$ . We can optimize this schedule by swapping the order of these two jobs. Let  $S'$  be the schedule  $S$  where we swap only the order of  $J_k$  and  $J_l$ . It is clear that the finishing times for all jobs except  $J_k$  and  $J_l$  does not change. The job  $J_l$  now schedules earlier, thus this job will finish earlier than in the original schedule. The job  $J_k$  schedules later, but the super-computer hands off  $J_k$  to a PC in the new schedule  $S'$  at the same time as it would handed off  $J_l$  in the original schedule  $S$ . Since the finishing time for  $J_k$  is less than the finishing time for  $J_l$ , the job  $J_k$  will finish earlier in the new schedule than  $J_l$  would finish in the original one. Hence our swapped schedule does not have a greater completion time.

If we define an inversion, as in the text, to be a pair of jobs whose order in the schedule does not agree with the order of their finishing times, then such a swap decreases the number of inversions in  $S$  while not increasing the completion time. Using a sequence of such swaps, we can therefore convert  $S$  to  $G$  without increasing the completion time. Therefore the completion time for  $G$  is not greater than the completion time for any arbitrary schedule  $S$ . Thus  $G$  is optimal.

*Notes.* As with all exchange arguments, there are some common kinds of mistakes to watch out for. We summarize some of these here; they illustrate principles that apply to others of the problems as well.

- The exchange argument should start with an arbitrary schedule  $S$  (which, in particular, could be an optimal one), and use exchanges to show that this schedule  $S$  can be turned into the schedule the algorithm produces without making the overall completion time worse. It does not work to start with the algorithm's schedule  $G$  and simply argue that  $G$  cannot be improved by swapping two jobs. This argument would show only that a schedule obtained from  $G$  by a *single swap* is not better; it would not rule out the possibility of other schedules, obtainable by multiple swaps, that are better.

---

<sup>1</sup>ex172.268.910

- To make the argument work smoothly, one should to swap neighboring jobs. If you swap two jobs  $J_l$  and  $J_k$  that are not neighboring, then all the jobs between the two also change their finishing times.
- In general, it does not work to phrase the above exchange argument as a proof by contradiction — that is, considering an optimal schedule  $\mathcal{O}$ , assuming it is not equal to  $G$ , and getting a contradiction. The problem is that there could many optimal schedules, so there is no contradiction in the existence of one that is not  $G$ . Note that when we swap adjacent, inverted jobs above, it does not necessarily make the schedule better; we only argue that such swaps do not make it worse.

Finally, it's worth noting the following alternate proof of the optimality of the schedule  $G$ , not directly using an exchange argument. Let job  $J_j$  be the job that finishes last on the PC in the greedy algorithm's schedule  $G$ , and let  $S_j$  be the time this job finishes on the supercomputer. So the overall finish time is  $S_j + f_j$ . In any other schedule, one of the first  $j$  jobs, in the other specified by  $G$ , must finish on the supercomputer at some time  $T \geq S_j$  (as the first  $j$  jobs give exactly  $S_j$  work to the supercomputer). Let that be job  $J_i$ . Now job  $J_i$  needs PC time at least as much as job  $j$  (due to the ordering of  $G$ ), and so it finishes at time  $T + f_i \geq S_j + f_j$ . So this other schedule is no better.