1. Recall the class scheduling problem described in lecture on Tuesday. We are given two arrays $S[1..n]$ and $F[1..n]$, where $S[i] < F[i]$ for each i, representing the start and finish times of n classes. Your goal is to find the largest number of classes you can take without ever taking two classes simultaneously.

   For each of the following greedy algorithms, either prove that the algorithm always constructs an optimal schedule, or describe a small input example for which the algorithm does not produce an optimal schedule. Assume that all algorithms break ties arbitrarily (that is, in a manner that is completely out of your control). ***Exactly three of these greedy strategies actually work.***

   (a) Choose the course x that *ends last*, discard classes that conflict with x, and recurse.

   Solution: This doesn't work. Given the input  , the greedy algorithm chooses the single long course, but the optimal schedule contains all the short courses. ■

   (b) Choose the course x that *starts first*, discard all classes that conflict with x, and recurse.

   Solution: This doesn't work. Given the input  , the greedy algorithm chooses the single long course, but the optimal schedule contains all the short courses. ■

   (c) Choose the course x that *starts last*, discard all classes that conflict with x, and recurse.

   Solution: ***This greedy strategy works!*** In fact, this is just a time-reversed version of the greedy algorithm proved correct in class. Correctness follows by induction from the following claim:

   Claim 1. *There is an optimal schedule that includes the course that starts last.*

   Proof: Let x be the course that starts last. Let S be any schedule that does not contain x, and let z be the last course in S. Because x starts last, we have $S[z] < S[x]$. Thus $F[i] < S[z] < S[x]$ for every other class i in S, which implies that $S' = S - z + x$ is still a valid schedule, containing the same number of classes as S. In particular, if S is an optimal schedule, then S' is an optimal schedule containing x. □

   ■

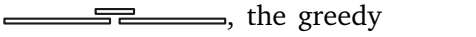
   (d) Choose the course x with *shortest duration*, discard all classes that conflict with x, and recurse.

   Solution: This doesn't work. Given the input  , this greedy algorithm chooses the single course in the middle, but the optimal schedule contains the other two courses. ■

   (e) Choose a course x that *conflicts with the fewest other courses*, discard all classes that conflict with x, and recurse.

Solution: This doesn't work. Given the input  , this greedy algorithm would choose the course in the center, which has only two conflicts, and thus would return a schedule containing only three courses. But the optimal schedule contains four courses. ■

(f) If no classes conflict, choose them all. Otherwise, discard the course with *longest duration* and recurse.

Solution: This doesn't work. Given the intervals  , the greedy algorithm chooses the single interval in the middle, but the optimal schedule contains the other two intervals. ■

(g) If no classes conflict, choose them all. Otherwise, discard a course that *conflicts with the most other courses* and recurse.

Solution: This doesn't work. Given the input  , this greedy algorithm would discard one of the courses in the middle of the bottom row, which has only five conflicts, and thus would return a schedule containing only three courses. But the optimal schedule contains four courses. ■

(h) Let x be the class with the *earliest start time*, and let y be the class with the *second earliest start time*.

- If x and y are disjoint, choose x and recurse on everything but x.
- If x completely contains y, discard x and recurse.
- Otherwise, discard y and recurse.

Solution: ***This greedy strategy works!*** We need to prove three claims, one for each case.

Claim 2. *If x and y are disjoint, then every optimal schedule contains x.*

Proof: If x and y are disjoint, then $F[x] < S[y]$, which ipmlues that $F[x] < S[i]$ for all $i \neq x$. Thus, if S is any valid schedule that does not contain x, then $S + x$ is a larger valid schedule. Thus, no optimal schedule excludes x. □

Claim 3. *If x contains y, there is an optimal schedule that excludes x.*

Proof: If x contains y, then every class that conflicts with y also conflicts with x. Thus, for any valid schedule S that contains x, there is another valid schedule $S - x + y$ of the same size that excludes x. □

Claim 4. *If x and y overlap, but x does not contain y, there is an optimal schedule that excludes y.*

Proof: Suppose x and y overlap, but x does not contain y. Then x must end before y ends, and therefore every class that conflicts with x also conflicts with y. Thus, for any valid schedule S that contains y, there is another valid schedule $S - y + x$ of the same size that excludes y. □

The correctness of this greedy strategy now follows by induction. ■

(i) If any course x completely contains another course, discard x and recurse. Otherwise, choose the course y that *ends last*, discard all classes that conflict with y, and recurse.

Solution: ***This strategy actually works!***

Claim 5. *If any course x contains another course y, there is an optimal schedule that does not include x.*

Proof: Let S be any valid schedule that contains x. Then $S - x + y$ is another valid schedule of the same size. □

Claim 6. *Suppose no course contains any other course. Then there is an optimal schedule containing the course that ends last.*

Proof: If no course contains any other course, then the class that ends last is also the class that *starts* last. This claim now follows from Claim 1 (in problem 3). □

The correctness of this greedy strategy now follows by induction. ■

2. A party of n people have come to dine at a fancy restaurant and each person has ordered a different item from the menu. Let $D_1, D_2, \ldots, D_n$ be the items ordered by the diners. Since this is a fancy place, each item is prepared in a two-stage process. First, the head chef (there is only one head chef) spends a few minutes on each item to take care of the essential aspects and then hands it over to one of the many sous-chefs to finish off. Assume that there are essentially an unlimited number of sous-chefs who can work in parallel on the items once the head chef is done. Each item D_i takes h_i units of time for the head chef followed by s_i units of time for the sous-chef (the sous-chefs are all identical). The diners want all their items to be served at the same time which means that the last item to be finished defines the time when they can be served. The goal of the restaurant is to serve the diners as early as possible. Consider the following greedy algorithms that order the items according to different criteria. For each of them either describe a counter example that shows that the order does not yield an optimum solution or give a proof that the ordering yields an optimum solution for all instances.

   - Order the items in increasing order of $h_i + s_i$.
   - Order the items in decreasing order of $h_i + s_i$.
   - Order the items in increasing order of h_i.
   - Order the items in decreasing order of h_i.
   - Order the items in increasing order of s_i.
   - Order the items in decreasing order of s_i.

   Solution: The last option gives an optimal schedule. One can find counter examples for all the others with just *two* items. We leave it as an exercise to find them.

   We will first prove that ordering the items in decreasing order of s_i is optimal. Assume without loss of generality that the items are numbered such that $s_1 \geq s_2 \geq \ldots \geq s_n$. Let $S = i_1, i_2, \ldots, i_n$ an optimum ordering of items with fewest inversions. If there no inversions in this ordering then we are done. Otherwise there are two adjecent items in the ordering i_ℓ and $i_{\ell+1}$ such that $i_\ell > i_{\ell+1}$. Consider swapping these items to obtain a new ordering S'. Let f_{i_ℓ} and f'_{i_ℓ} denote the finish times of i_ℓ in S and S' and similarly $f_{i_{\ell+1}}$ and $f'_{i_{\ell+1}}$. Note that no other items are affected by the swapm. Letting t denote the time at which i_ℓ is started by the head chef in S we observe that $f_{i_\ell} = t + h_{i_\ell} + s_{i_\ell}$ and $f_{i_{\ell+1}} = t + h_{i_\ell} + h_{i_{\ell+1}} + s_{i_{\ell+1}}$. After the swap we have $f'_{i_\ell} = t + h_{i_\ell} + h_{i_{\ell+1}} + s_{i_\ell}$ and $f'_{i_{\ell+1}} = t + h_{i_{\ell+1}} + s_{i_{\ell+1}}$. Since $s_{i_\ell} \leq s_{i_{\ell+1}}$ we have

$$\max\{f'_{i_\ell}, f'_{i_{\ell+1}}\} = \max\{t+h_{i_\ell}+h_{i_{\ell+1}}+s_{i_\ell}, t+h_{i_{\ell+1}}+s_{i_{\ell+1}}\} \leq t+h_{i_\ell}+h_{i_{\ell+1}}+s_{i_{\ell+1}} = \max\{f_{i_\ell}, f_{i_{\ell+1}}\}.$$

   This implies that the new ordering is no worse than the previous one (why?) but has one fewer inversion contradicting the choice of the ordering S as the an optimum schedule with the fewest inversions.

   ■