PERT Algorithm:

Given n tasks, each with a time and preceding tasks list:

- go through the list and
  - write in the first column all tasks that have no preceding tasks
  - write the time next to the task
- repeat until all tasks are recorded, moving over by one column each time:
  - go through the list of tasks that have not been put in the graph yet
    - if all of the preceding tasks are already in the graph in previous columns, write the task in the next column
    - write the time next to the task
    - draw a line or arrow from each preceding task to the task
- For all the tasks in the first column, write its time next to it again as the time subtotal for that task
- Go through all the tasks in the second column, then all in the third column, etc. until done
  - for each task find the subtotals of its preceding tasks (using the lines/arrows to find the preceding tasks) and find the maximum of those subtotals
  - Add that maximum subtotal to the time for that task, and record it as the subtotal for that task.
- Go through all of the tasks, and find the one or ones with the maximum subtotal. That maximum subtotal is the total time for the job.
- Starting with a task that has the maximum subtotal, record that task as the last in a critical path.
- then repeat until you reach a task in the first column:
  - Find the subtotals of the preceding tasks and find the one with the maximum subtotal. Record it as the prior task in the critical path.

Matching problem: algorithm A:

Given n cities, each of which has listed pilots that requested that city from a total of n pilots

1. For each city (in the order listed)
   - If the city has only one pilot that has requested it, assign that pilot to that city, and remove the pilots name from the remaining cities.
- repeat step 1 until there are no cities with only a single request
2. Find a city with a minimum positive number of requests.
   - Randomly assign a pilot who has requested that city to fly to that city and remove that pilot's name from the remaining cities
- repeat step 2 until there are no cities with pilots requesting them.
- randomly assign the remaining pilots to the remaining cities.

Matching problem: algorithm B:

- For each pilot in alphabetical order
  - Find the first pilot with a minimum number of city requests
  - Find the first of that pilot's requested cities that has a minimum number of pilot requests, and assign that pilot to that city.
  - Remove the pilot and the city from the lists.
- Repeat until no requests remain
- randomly assign the remaining cities to the remaining pilots

Matching problem: algorithm C:

- Sort the cities in ascending order by number of pilots requesting that city.
  - For each city, find a pilot requesting that city with minimum number of city requests.
  - Remove the pilot and city from the list(s)
  - If any city has no requests, skip it for this step
- Randomly assign pilots to the remaining cities.