Homework 2:

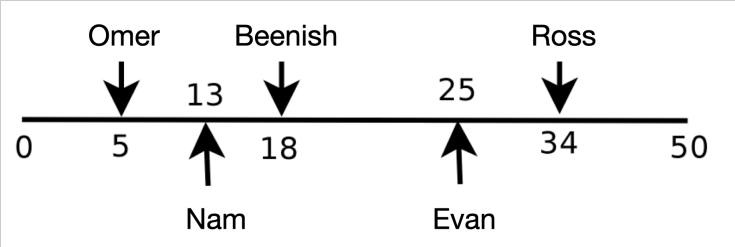
Due: September 25, 2025 at 2:30p.m.

This homework must be typed in LaTeX and submitted via Gradescope.

Please ensure that your solutions are complete, concise, and communicated clearly. Use full sentences and plan your presentation before your write. Except where indicated, consider every problem as asking for a proof.

Problem 1. The power lines along a country road have been modified to carry broadband Internet. Wi-Fi towers are being built along the road to provide the community with internet. To find the minimum number of towers required so that each house is sufficiently close to at least one tower, we model the problem as follows:

a) The entire course staff has taken up residence on Algorithm Street. The diagram below shows where they all live on the street.



Omer lives 5 miles down the road, Nam lives 13 miles down the road, and so on. Wi-Fi towers have an effective radius of 5 miles. Determine the minimum number of Wi-Fi towers needed such that each staff member has internet, and give the locations for these towers as well.

b) We're given a line segment ℓ, a set of non-negative numbers N that represents the locations of customers on ℓ, and a distance d. We wish to find a set of Wi-Fi towers of minimal size on ℓ such that each location in N is at most d away from some tower. Give an efficient greedy algorithm that returns a minimum size set of points. Prove its correctness and justify its runtime.

Now we generalize our model to account for houses that are not by the side of the road.

c) We're given a line segment ℓ, a set of pairs N representing the locations of customers, and a distance d. For each pair $(x, y) \in N$, let $x \in [0, \infty)$ be the distance along ℓ and $y \in [-d, d]$ be the distance above or below ℓ. We wish to find a set of Wi-Fi towers of minimal size on ℓ such that each location in N is at most distance d from some tower (here, we are using Euclidean distance).

Modify your algorithm from part (b) to solve this variation of the problem. You do not need to prove its correctness, but please explain how your proof from part (b) would (or would not) need to change based on your modifications.

Can we generalize further?

d) Does the correctness of your algorithm depend on the fact that ℓ is a line segment and not some curve? If so, give an example that illustrates the problem with your algorithm when ℓ is a curve. If not, explain how your algorithm could handle a curve. You shouldn't be writing another algorithm, or modifying your existing algorithm, just explain your reasoning.

Solution. □

Problem 2. Given an array A of n distinct integers sorted in non-decreasing order, design an $O(\log n)$ algorithm to decide (i.e. output true/false) whether there exists an index i such that $A[i] = i$.

1. Provide a succinct (but clear) description of your algorithm. You may provide pseudocode.
2. Prove the correctness (optimality) of your algorithm.
3. Analyze the running time and memory utilization of your algorithm.

Solution. □

Problem 3. Suppose you are given a set $S = \{a_1, a_2, ..., a_n\}$ of tasks, where task a_i requires p_i units of processing time to complete, once it has started. You have access to a computer to run these tasks one at a time. Let c_i be the **completion time** of task a_i, i.e. the time at which task a_i completes processing. Your goal is to minimize the average completion time:

$$\frac{1}{n}\sum_{i=1}^{n} c_i$$

For example, suppose there are two tasks, a_1 and a_2, with $p_1 = 3$ and $p_2 = 5$, and consider the schedule in which a_2 runs first, followed by a_1. Then, $c_2 = 5,\ c_1 = 8$, and the average completion time is 6.5.

(a) Give an algorithm that schedules the tasks to minimize the average completion time. Each task must run non-preemptively, that is, once task a_i is started, it must run continuously for p_i units of time. Prove that your algorithm minimizes the average completion time, and prove the running time of your algorithm.

(b) Suppose now that the tasks are not available at once. Each task has a **release time** r_i before which it is not available to be processed. Suppose also that we allow **preemption**, meaning a task can be suspended and restarted later.

For example, a task a_i with processing time $p_i = 6$ may start running at time 1 and be preempted at time 4. It can then resume at time 10 but be preempted at time 11 and finally resume at time 13 and complete at time 15. Task a_i has run for a total of 6 time units, but its running time has been divided into three pieces. We say that the completion time of a_i is 15.

Give an algorithm that schedules the tasks so as to minimize the average completion time in this new scenario. Prove that your algorithm minimizes the average completion time, and state the running time of your algorithm.

Solution. □