Problem Set 1

1. (Easy) You are given a list of intervals, where each interval represents the time during which a professor at IIT is teaching on a Monday. Each entry in the list is a tuple containing the starting and ending time of the lecture for each professor. Your task is to determine the time slot during which the maximum number of professors are simultaneously teaching. In other words, find the time at which the most professors are conducting their lectures concurrently.

2. (Easy) You are given an array A of intervals, where each interval $A[i]$ is represented as (s_i, e_i), indicating an interval starting at s_i and ending at e_i. Implement a function to merge all overlapping intervals in A and return an array of non-overlapping intervals that cover all the intervals in the input array.

   For example, if $A[(1,3),(2,4),(6,10),(7,12)]$, then the output is $[(1,4),(6,12)]$

3. (Medium) Given a collection of intervals, find the minimum number of intervals you need to remove to make the rest of the intervals non-overlapping.

4. (Easy) In a cricket maidan in Mumbai, typically on Sunday, you will see hundered of matches being played simultaneously on the same maidan. Suppose you have the list of all the matches with the start and end times. Find out the total number of time units for which there is only one team playing in the maidan.

   If there are two teams with timings [(2,5),(3,8)], then the answer is 4.

5. (Medium) You are given two set of intervals A and B. An interval in $a \in A$, covers an interval $b \in B$, if a and b overlaps. Find the minimum number of intervals in A, that cover all intervals in B.

6. (Medium) A village has n houses where each house i is built at the location $(i, 0)$ from the origin. A telephone company has installed k towers on top of k houses to cover all the houses in the village. For each tower j, you know its location, say ℓ_j and a range, say r_ℓ. The tower j, can cover all the houses in the range $[\ell_j - r_j, \ell + r_j]$. Can all the k tower cover all the houses in the village?

7. (Medium) Given two lists of closed intervals, each list of intervals is pairwise disjoint and in sorted order, return the intersection of these two interval lists.

8. (Hard) You are given a circular array A of n numbers. An index i is called *inferior* if $A[i] \leq i$. However, there is a catch. Since the array is circular, you can define the starting index of the array. You job is to find the starting position in the array which maximizes the number of inferior numbers in the array.

   For example, if $A = [0,1,2]$, then the answer is 0. If $A = [3,4,5,0,1,2]$, then the answer is 3. If $A = [4,1,0]$, then the ans is 2.

9. (Easy) The two processor interval scheduling problem takes as input a sequence of request intervals $(s_1, f_1), \ldots, (s_n, f_n)$ just like the unweighted interval scheduling problem except that it produces two disjoint sets $A_1, A_2 \subset [n]$ such that all requests in A_1 are compatible with each other and all requests in A_2 are compatible with each other and $|A_1 \cup A_2|$ is as large as possible. (A_1 might contain requests that are incompatible with requests in A_2). Does the following greedy algorithm produce optimal results? If yes, argue why it does; if no, produce a counterexample.

```
Sort requests by increasing finish time;
$A_1 = \emptyset$;
$A_2 = \emptyset$;
while there is any request $(s_i, f_i)$ compatible with either $A_1$ or $A_2$ do
    Add the first unused request, if any, compatible with $A_1$ to $A_1$;
    Add the first unused request, if any, compatible with $A_2$ to $A_2$;
end
```

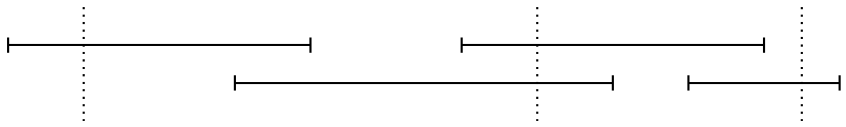
10. (Medium) You are given an array $A[1, \ldots, n]$ with the following specifications:

    - Each element $A[i]$ is either a cop ('c') or a robber ('r').
    - A cop can catch any robber who is within a distance of K units from the cop, but each cop can catch at most one robber.

    Write an algorithm to find the maximum number of robbers that can be caught.

11. (Medium) Suppose that you are in charge of the PA system at a school. When you make an announcement over the PA system, all classes currently in session hear it. There is a very important announcement you need to make, and to ensure that everyone hears it, you want to announce it enough times to guarantee that every single class hears the announcement at least once. Because some students might be in multiple classes during the day, you want to minimize the number of times that you have to make the announcement. It's fine if you make the announcement two or more times during a particular class; it might be a nuisance, but your priority is ensuring that each class hears it at least once.

    For example, if you were given the set of classes (denoted by their timespans), one possible set of times you could make the announcements is given by the vertical dotted lines:

    This particular set of times isn't a minimum set of times; it's possible to cover each class using only two announcements. You are given as input a list of intervals representing the start and end times of each class. Design and analyze a polynomial-time algorithm that finds a minimal set of announcement times that is guaranteed to reach every class.

    For simplicity, you can assume that no class starts as soon as another one ends and that when you start making the announcement all classes in session will hear it, including those that are just starting or just ending.

12. (Hard) You are given n task each with a duration and a deadline. The i task requires d_i duration and deadline t_i. If you attempt the task then you need to work on it for d_i hours and complete it by time t_i. You cannot work on two tasks simultaneously. Design an algorithm that finds the maximum number of tasks you can perform. For example, if the n tasks are $[(1, 5), (2, 4), (5, 5)]$, then you can finish the first task in one hour. Then you complete the second tasks by the third hour. Unfortunately, you will not be able to perform the last task. So, the answer to this instance is 2.