The Holy Fire

Problem Description

Given N intervals, each having its start s_i and end e_i and the circle of length M, we need to take the minimum number of those intervals to cover the full circle. If it is not possible, the answer should be -1.

Solution

Subtask 1: $N \leq 20$

In this subtask it is possible to check each subset of the intervals. If there are subsets that cover the full circle, then we choose the one that has the smallest size. Otherwise, the answer is -1.

Checking that the selected intervals cover the full circle is not trivial, though. One way is to sort the intervals by starts and keep track of furthest found end. There might be other ways.

The time complexity for this is $O(2^N N \log N)$.

Subtask 2: $N \leq 300$

Let's say that *next best* (or just *next*) interval for interval i is such an interval j that overlaps (or touches) with i, and also the one that has the furthest end (taking into account that it can round around the point 0). We will denote it as $next_i = j$.

Observe that if we are taking some interval i into the answer, then we can also take $next_i$ into the answer, as taking any other interval after the i-th one (apart the $next_i$) can only make the answer worse.

To solve this subtask, we can try each interval as a "starting" interval. Suppose we are trying a starting interval a. Then we take interval $b = next_a$, then $c = next_b$, and so on. We keep on taking *next* intervals until the full circle got covered. Searching for the *next* interval takes $O(N)$ time, and we will do this at most $O(N)$ times. Thus, the time complexity to find intervals from some specific interval takes $O(N^2)$ times.

We need to try each of the N intervals as a starting interval, so in total the solution takes $O(N^3)$ time.

Subtask 3: $N \leq 5\,000$

The solution for this subtask is the same as the last one. The only difference is that before solving we can precalculate each of the *next* intervals.

This precalculation step takes $O(N^2)$ time, and the rest of the solution now becomes $O(N^2)$ as well.

Useful observation

Observe that we don't need any intervals that are fully covered by some other interval.

The solutions below will assume that our interval list doesn't have the intervals that are fully covered by some other interval in the list.

It is possible to remove these intervals from the list in $O(N \log N)$ time. We can sort the original intervals by increasing starting time (and then by decreasing ending time). Then we iterate through this list while keeping track what was the furthest ending time. When we iterate over some interval i and the furthest ending time so far is not closer than e_i, then we can discard interval i.

Calculating *nexts* faster

To precalculate *next* intervals faster, we can sort the intervals by their starts s_i. Then we can traverse through this list using two pointers i and j. For each i, we try to find the last j in the list, such that i still overlaps with j. When we do that, we know that $next_i = j$. However, don't forget to deal with some tricky edge cases in the implementation that may arise since in the task we are dealing with the circle.

This way precalculating *nexts* takes $O(N \log N)$ time (due to sorting).

Subtask 4: for all intervals $s_i < e_i$, or $e_i = 0$

In this subtask there are no intervals that go over the point 0 (unless they end at that point). That means we are basically dealing with the line (of length M) case, and don't need to consider the circle. We can consider any interval $[s_i, 0]$ as $[s_i, M]$.

We can find the suitable interval that we could take as a "starting" one. This interval is such an interval that starts at 0 ($s_i = 0$) and which has the furthest end (the largest e_i). From that interval we keep taking *next best* intervals until the full circle (in this subtask, full line) is covered.

Note that in this subtask it is not needed to precalculate *nexts* in an efficient way. It is also possible to just sort the intervals by their start and keep iterating through this list until we find the *next best* interval.

Overall, the time complexity for this solution is $O(N \log N)$.

Subtask 5: each interval has the same length

This subtask has the same solution as the full problem. However, for this subtask we can skip the step where we remove the intervals that are fully covered by other intervals, as there are none! The only exception is when there are some equal (same start and same end) intervals, but this case is not too hard to handle.

Full solution #1

We can use binary lifting[1] to speed up the $O(N^2)$ solution from subtask 3.

From each starting admin we can do binary lifting (imagine this as doing binary search) to find how much *nexts* should you take in order to still fill the full circle. Binary lifting allows to solve this in $O(N \log N)$.

Full solution #2

Here we will also give an alternative suggestion which might be easy to come up with, but harder to prove that it works. However, here we will also not give the proof. We apologize for that :D

[1] https://cp-algorithms.com/graph/lca_binary_lifting.html

The idea is as follows: start from any interval. Let's go to *next* of it, then to the *next* of that, and so on, until we have made N steps. The interval that we are at now belongs to a possible answer. We can choose this as a "starting" interval, and then keep selecting *nexts* until the full circle is covered. The number of intervals that were taken is the answer.

Credits

- Task: Sandra Schumann, Ahto Truu (Estonia)
- Solutions and Tests: Daumilas Ardickas, Aldas Lenkšas (Lithuania)