Subway Signalling Error

Spoiler

One solution is to binary search for the answer. To test if it's possible to get the trains in order within time t, assume that at least one train has to move exactly the distance t (since a train can move both left and right, there are $2n$ possibilities to try). We can now determine where all other $n-1$ trains must be. To check if this is possiblity, note that this is a bipartite matching problem where one partition consists of the trains original position, and the other partition consists of a trains final position. An edge exists between nodes in the left and right partition if the distance between the posistions are at most t.

Solving the max match problem with standard bipartite matching is too slow however. This can instead be accomplished much more easily by realising that a greedy approach will work. The train with the leftmost original position can always be assigned to the leftmost of the new positions, and so on. If such an assignment is impossible (because the distance is greater than t), then no max matching exists.

The solution then involves a binary search (about 30 iterations or so needed to get the desired precision), fixing the posistion of one train ($O(n)$), and a greedy matching ($O(n)$). The overall complexity is thus $O(n^2)$ with a fairly large constant factor due to the binary search.

The optimal solution is more mathematical, and runs in $O(n \log n)$ time: First we notice that if we sort trains from left to right on the line, then the odd trains should be going in one direction and even ones in the other. That is because we are not swapping two trains in an optimal solution, and when the trains are evenly distributed, the odd ones face one direction and the even ones the other. So now we know the directions and we can only move each train to the left or right. Consider the example from problem statement. After sorting we calculate directions:

$$\mathtt{9R}, \mathtt{15L}, \mathtt{33R}, \mathtt{33L}, \mathtt{41R}, \mathtt{81L}, \mathtt{97R}, \mathtt{100L}$$

Now it is like we were on the circle of length 200:

$$9, 33, 41, 97, 100, 119, 167, 185$$

Assume that we don't move the first train. In such situation, we easily calculate where the trains should be:

$$9, 34, 59, 84, 109, 134, 159, 184$$

and the movements of the trains should be, respectively:

$$0, -1, -18, 13, -9, -15, 8, 1$$

Now, if the first train moves by x, all the others move by $a + x$, where a is the move if first train stands still. If we find minimum and maximum over the moves ($m = -18$ and $M = 13$), when first train stands, then when it moves by x, the minimum and maximum will be $m + x$ and $M + x$. Our result would then be $\max(|m+x|, |M+x|)$. This is minimized when $m+x = -(M+x)$, which gives $x = \frac{-M-m}{2}$ and thus the final result is $M+x = \frac{M-m}{2}$, i.e., 15.5 in the example. The whole algorithm runs in $O(N \log N)$

Test data overview

Each test case contains n trains whose positions are random with a uniform distribution over the line; all initial directions are randomized as well. There are 20 test cases with different input sizes, each one is worth 5 points:

Test case	m	n
1	100	5
2	130	8
3	1000	15
4	1197	18
5	13238	80
6	15729	97
7	52819	123
8	93723	140
9	1304932	197
10	2830247	200
11	5382395	3175
12	6837119	4812
13	12438290	10293
14	12841281	25108
15	23812731	37819
16	43716289	47182
17	65311298	58890
18	73184100	67189
19	91289271	96327
20	100000000	99998