Editorial for CCC '22 S4 - Good Triplets

Author: **d**

For the first subtask, it suffices to follow the definition of a good triplet. Firstly, there are $\mathcal{O}(N^3)$ ways to select (a, b, c). Secondly, consider the condition that the origin is strictly inside the triangle. This is equivalent to saying that P_a, P_b, and P_c divide the circle into 3 arcs, and each arc is strictly shorter than $\frac{C}{2}$. This leads to the following approach. Sort the positions so that $P_a \le P_b \le P_c$, then check that

$$P_b - P_a < \tfrac{C}{2},\ P_c - P_b < \tfrac{C}{2},\ \text{and}\ (P_a + C) - P_c < \tfrac{C}{2}.$$

For example, the first of these checks can be implemented in code as `P[b]-P[a] < (C+1)/2` in C++ or Java, or `P[b]-P[a] < (C+1)//2` in Python 3.

For the second subtask, the goal is to find a solution that works well when C is small. Let L_x be the number of points drawn at location x. If three locations i, j, k satisfy the second condition, we want to add $L_i \times L_j \times L_k$ to the answer. This approach is $\mathcal{O}(N + C^3)$ and is too slow. However, we can improve from three nested loops to two nested loops. If i and j are chosen already, either k does not exist, or k is in an interval. It is possible to replace the third loop with a prefix sum array. This algorithm's time complexity is $\mathcal{O}(N + C^2)$.

It is possible to optimize this approach to $\mathcal{O}(N + C)$ and earn 15 marks. The idea is to eliminate both the j and k loop. Start at $i = 0$ and compute $L_j \times L_k$ in $\mathcal{O}(C)$ time. The next step is to transition from $i = 0$ to $i = 1$, and to maintain $L_j \times L_k$ in $\mathcal{O}(1)$ time. This requires strong familiarity with prefix sum arrays. Care with overflow and off-by-one errors is required.

Comments

13

neynt commented on March 6, 2022, 4:36 p.m.

I found this problem a lot easier to think about if you count triplets of points that *don't* form a triangle and subtract that from the total number of triplets. These are exactly the triplets whose points all belong on a closed half of the circle, so sweeping two pointers along the circle suffices to do this in O(N + C) time.