计算几何

题目大意

给定一个包含 n 个点的序列，第 i 个点的坐标为 (a_i, b_i)。

共有 q 次询问，每次询问一个区间 $[l, r]$，你需要求出下面式子的值。

$$\min_{i=l}^{r} \min_{j=i+1}^{r} (|a_i - a_j| + |b_i - b_j|)$$

保证序列中点的坐标和询问区间在指定范围内用指定方式随机生成。

数据范围

对于所有测试点，$2 \le n \le 10^6$，$1 \le q \le 10^6$，$|a_i|, |b_i| \le 10^9$，$1 \le l < r \le n$，保证 a_i, b_i, l, r 在指定范围内用指定方式随机生成。

测试点编号	$n \le$	$q \le$
$1 \sim 2$	2×10^3	2×10^3
$3 \sim 8$	2×10^4	2×10^4
$9 \sim 14$	2×10^5	2×10^5
$15 \sim 16$	2×10^3	10^6
$17 \sim 19$	10^6	10
$20 \sim 25$	10^6	10^6

对于每一档部分分，设其测试点编号范围为 $l \sim r$，则测试点 $l+1 \sim r-1$ 满足 $|a_i|, |b_i| \le 10^6$，测试点 $\lfloor \frac{l+r}{2} \rfloor + 1 \sim r$ 满足 $b_i = 0$。

时空限制

时间限制：4 秒。

空间限制：1024 MB。

测试点 $1 \sim 2$

直接暴力模拟题面中的式子即可，由于数据随机且暴力常数小，故跑 2×10^3 完全没问题，总时间复杂度 $O(qn^2)$。

测试点 $15 \sim 16$

直接预处理出所有询问的答案，具体就是枚举左端点暴力 $O(n)$ 扩展右端点，总时间复杂度 $O(n^3 + q)$，常数很小。

测试点 $17 \sim 19$

考虑单次询问有没有更快速的做法，首先如果 $b_i = 0$ 的话就直接按 a_i 排序后相邻两个 a 作差求最小值即可。

对于一般的情况，还是按 a_i 排序，每次新加入一个点 (a_i, b_i) 时按另一个点与该点的 b 的大小关系分两类讨论即可。

具体的，维护满足 $b_j \le b_i$ 的最小的 $-a_j - b_j$ 的值和满足 $b_j \ge b_i$ 的最小的 $-a_j + b_j$ 的值，再加上该点坐标即可。

这个东西可以 `离散化` 后使用 `树状数组` 等 `数据结构` 简单的维护，总时间复杂度 $O(qn \log n)$，需常数较小才可通过。

测试点 $3 \sim 8$

对于 $b_i = 0$ 的部分分可以直接 `莫队` + `set` 维护按 a_i 排序后的数组，并维护答案，总时间复杂度 $O(n\sqrt{q} \log n)$。

而 $b_i \neq 0$ 时出题人并没有想到什么好的做法，~~可以看作下一档分被卡常得到的分~~，如果有好做法可以和出题人讨论。

测试点 $9 \sim 14$

当 $b_i = 0$ 时是一个极为经典的问题，可以使用 `分块` 、 `回滚莫队` 或者 `权值线段树` 等算法解决，原题： CF765F 。

由于上述几个做法都比较复杂，故在这里不再过多阐述，~~其实出题人也不会会~~，大家感兴趣的话可以参考原题题解。

而 $b_i \neq 0$ 时出题人不知道有没有不基于随机的做法，如果有的话可以和出题人讨论，基于随机的做法详见后面。

测试点 $20 \sim 25$

出题人依旧不知道该部分分有没有不基于随机的做法，如果有的话可以和出题人讨论，基于随机的做法详见后面。

算法 1

通过观察可以得到（~~出题人太坏了观察大样例无法得到~~）当 $b_i = 0$ 且 $|a_i| \leq 10^6$ 时大概率会有很多重合的点。

> 生日悖论：随机若干个范围在 $1 \sim n$ 的整数，期望随机 $O(\sqrt{n})$ 个数就会出现相等元素。
>
> > （感性）证明：随机 k 个数没有相等元素的概率为 $\prod_{i=1}^{k} \frac{n-i+1}{n}$ ，当 k 取到 $O(\sqrt{n})$ 时概率达到 $\frac{1}{2}$ 。
> >
> > 当 k 取到 $O(\sqrt{n})$ 时原式约等于 $(\frac{\sqrt{n}-1}{\sqrt{n}})^{\sqrt{n}} \approx \frac{1}{e} \approx \frac{1}{2}$ ，而随着 k 的变化概率变化幅度是指数级的。

根据 `生日悖论` ，当 n 达到 2×10^4 及以上时会出现很多对重合的点，由于询问区间是随机的所以大概率答案为 0 。

具体的，设编号为 $id_1, id_2, \ldots, id_k$ 的点坐标相同，则提取出 $(id_1, id_2), (id_2, id_3), \ldots, (id_{k-1}, id_k)$ 这 $k-1$ 对点。

然后判断一个询问答案是否为 0 就直接遍历这些点对，若询问区间包含任意一个点对，则该询问答案为 0 。

只加入相邻 id 下标的两个点组成的点对的原因是不相邻的点对被包含所需的询问区间更长，必定没有相邻的优。

由于区间较长的询问答案基本均为 0 ，剩余询问区间较短，故直接用 `测试点 17~19` 的暴力跑答案不为 0 的询问即可。

至此应该就可以基本通过所有 $b_i = 0$ 且 $|a_i| \leq 10^6$ 的测试点，特别的，由于 n 很大，故 `测试点 25` 也有可能通过。

至于该算法的时间复杂度，由于算法还并不太成型，故不分析时间复杂度，~~其实就是出题人不会~~。

算法 2

考虑拓展 `算法 1` ，把找重合的点对变为计算全局内前 k 小的点对，查询时依次遍历这 k 对点，若包含则得到答案。

首先考虑当 $b_i = 0$ 时如何计算上述内容，直接将所有点按 a_i 从小到大排序，此时最小的点对必定为相邻的两个点。

求前 k 小可以初始化 $l_i = i - 1$ ，将 $a_i - a_{l_i}$ 插入 `优先队列` ，每次弹出最小的点对，将 $l_i \leftarrow l_i - 1$ 并插入 (l_i, i) 。

而当 $b_i \neq 0$ 时做法也是大同小异，考虑拓展 `测试点 17~19` 的做法，维护一棵 `可持久化线段树` 用来查询。

具体的，还是按 a_i 排序，依旧按 b 的大小维护两个信息，每次弹出一个点对时直接重新查询 i 对应的 `主席树` 即可。

至此，我们已经得到了 $O((n+k)\log n)$ 求得全局前 k 小点对的做法，而空间复杂度 `离散化` 后则为 $O(n \log n)$ 。

k 可以看作是一个用来平衡预处理与询问时间复杂度的阈值，上述算法还需要再优化，不过已经可以通过较小数据。

算法 3

我们曾得到过如下性质：若点对 (x_1, y_1) 的答案 $\leq$ 点对 (x_2, y_2) 的答案且 $x_2 \leq x_1 < y_1 \leq y_2$ ，则 (x_2, y_2) 无用。

考虑利用上述性质排除掉所有无用的点对，如果直接做的话必然是麻烦的，考虑继续利用保证数据随机这一性质。

由于数据随机，故可以看作点对在 $1 \sim n$ 范围内近乎随机生成，首先考虑这 k 个点对做完上述操作后会剩余多少个。

将点对按照答案从小到大插入，第 i 个点对有用的概率即为前 $i-1$ 个点对均无法包含它的概率。

~~出题人曾这么考虑过：包含某点对的概率 $\approx \frac{1}{6}$ ，故点对 i 无用的概率为 $1-(\frac{5}{6})^{i-1}$ ，即只会剩余 $O(1)$ 个点对！~~

上述思路显然是错误的，因为事实上这若干个点对是不能独立的，考虑确定一个东西使得点对与点对可以独立计算。

设点对 i 的两个点的编号差的绝对值为 l ，则它包含某点对的概率 $\approx \frac{l^2}{n^2}$ ，而该点对无用的概率 $\approx (1-\frac{l^2}{n^2})^{i-1}$ 。

当 i 取到 $\frac{n^2}{l^2}$ 时概率 $\approx \frac{1}{e} \approx \frac{1}{2}$ ，由于概率指数级变化，所以可以基本看作 $i \leq \frac{n^2}{l^2}$ 时点对 i 会被保留，否则不会。

而点对编号差的绝对值基本可以看作是均匀抽取的，由于 l 需要满足 $\leq \frac{n}{\sqrt{i}}$ ，故点对 i 会被保留的概率就 $\approx \frac{1}{\sqrt{i}}$ 。

$$\sum_{i=1}^{k}\frac{1}{\sqrt{i}}\approx 2\sqrt{k}$$

k 个点对最后期望会剩余的点数即为上式，即 $O(\sqrt{k})$ 级别，上式容易通过 `积分` 证明，这里不再过多赘述。

回到原问题，如何去掉无用点对，容易想到一个简单的方法，直接遍历点对 i 前有用的点对，时间复杂度 $O(k\sqrt{k})$。

写一下发现跑得巨快，怎么会是呢？原来是若一个点对已经包含某个有用点对后就直接 `break`，这会改变复杂度吗？

考虑枚举 m 个有用点对后依旧没有 `break` 的概率，由于只枚举有用点对，故此时就相当于枚举了 $O(m^2)$ 个点对。

所以没 `break` 的概率 $\approx(1-\frac{l^2}{n^2})^{m^2}$，即可以看作 $m^2\le\frac{n^2}{l^2}\iff l\le\frac{n}{m}$ 时不会 `break`，也就是说概率 $\approx\frac{1}{m}$。

$$\sum_{m=1}^{O(\sqrt{i})}\frac{1}{m}=O(\ln\sqrt{i})=O(\ln i)$$

根据上述推导可以得到点对 i 期望判断的有用点对数即为上式，而去掉所有无用点对的时间复杂度即为 $O(k\ln k)$。

至此，我们就已经得到了 $O(\sqrt{k})$ 个有用的点对，里面浓缩着 $O(k)$ 个点对，现在我们要拿它们去回答 q 个询问。

发现由于询问区间的长度也是均匀随机的，故时间复杂度与上面的分析类似，需要判断的有用点对数为 $O(q\ln k)$。

但是如果不包含任何一个有用点对怎么办呢？考虑此时期望的询问长度为 $O(\frac{n}{\sqrt{k}})$，而这样的询问共有 $O(\frac{q}{\sqrt{k}})$ 个。

所以这样的询问区间长度之和是 $O(\frac{qn}{k})$ 的，而如果直接使用 `测试点 17~19` 的暴力即可做到 $O(\frac{qn}{k}\log n)$。

总时间复杂度 $O(n\log n+k\ln k+q\ln k+\frac{qn}{k}\log n)$，由于 n,q 同阶，故 k 取到 $O(n)$ 时总时间复杂度最优。

总时间复杂度 $O((n+q)\log n)$，由于预处理部分是 `主席树`，而询问部分则是 `树状数组`，故 k 取小一点较快。

注意该算法总空间复杂度 $O(n\log n)$，因为最开始需要建 `主席树`，而它事实上也是时间复杂度的巨大瓶颈……

如果选手有更优秀或者常数更小的做法也欢迎和出题人讨论，`主席树` 建树跑 $n=10^6$ 大概要占一半多的时间……

~~顺带一提，这个算法竟然还是在线的，非常神奇。~~

参考资料

CF765F

生日悖论 - 百度百科

致谢

感谢李欣隆与我关于本题不保证数据随机时做法的交流。