E1-E题题解

题目描述

在 x-轴上分布着 n个暗点，第 i个暗点位于 x=a_i处，暗点坐标**两两不同。**

每次操作时，你可以选择一个暗点并将其点亮。

当一个暗点被点亮，与其距离**不超过** d的所有暗点也将被点亮。一个暗点被点亮后会成为亮点，之后无法被操作。

你需要求出将所有暗点点亮的最少操作次数。

输入

本题测试点包含多组数据。

第一行，一个正整数 T（1≤T≤50），表示数据组数。

对于每组数据：

第一行，两个正整数 n,d（1≤n≤10^4，1≤d≤10^9）。

第二行，n个正整数 $a_1,a_2,...,a_n$（1≤a_i≤10^9），表示暗点所在的横坐标。

输出

对于每组数据，输出一行，一个整数，表示将所有暗点点亮的最少操作次数。

输入样例

```
2
5 2
1 5 9 3 7
4 3
8 2 6 1
```

输出样例

```
1
2
```

样例解释

解释第一组样例。首先考虑位于 x=1 处的暗点，它能点亮的范围最远可以到 x=1+2=3处，因此同步点亮了 x=3 处的暗点。而 x=3处的暗点最远可以点亮到 x=5 处，因此位于 x=5处的暗点也同时被点亮。一直这样传递过去会将全部的暗点都点亮。因此只需要点亮 x=1处的暗点一次即可将全部的暗点都点亮。

解题思路

根据题目：

> 当一个暗点被点亮，与其距离**不超过** d的所有暗点也将被点亮。

可知，将一个暗点点亮时，可以从其**左右两边**传递点亮与其距离不大于d的点。换句话说，若一个点是**被传递点亮的**，那么它要么从左边被点亮，要么从右边被点亮。

但只要我们稍稍思考就能发现，传递点亮的两个点间的关系是相互的，比如：有两个点a和b，a在b左侧，若a点能被b点传递点亮，那么b点肯定也能被a点传递点亮。

因此，我们只需要从x=1处，从左到右依次判断各个暗点的操作情况即可。

解题步骤

1. 将所有暗点按坐标顺序从小到大排序；
2. 默认选中第一个暗点，记被选中的点为**起始u点**；
3. 依次检测在**u点**后的所有暗点，按照如下规则处理：
    - 若该暗点与**u点**距离小于d，则标记为点亮；
    - 当遇到第一个与**u点**距离大于d的暗点时，结束本轮检测，记该暗点为新的**起始u点**；
4. 从当前的**起始u点**开始，重复进行第三步，每更改一次**起始u点**标记，点亮操作次数就加一，直到检测完所有的暗点。

完整代码

```c
#include <stdio.h>
#include <stdlib.h>

int compare(const void *a, const void *b) {
    return (*(int *) a - *(int *) b);
}

int main() {

    int t;
    scanf("%d", &t);
    while (t--) {
        // 输入
        int n, d;
        scanf("%d %d", &n, &d);
        int a[10005] = {0};
        for (int i = 0; i < n; ++i) {
            scanf("%d", &a[i]);
        }

        // 对暗点按照坐标排序
        qsort(a, n, sizeof(int), compare);

        int ans = 0; // 记录操作次数
        for (int i = 0; i < n; ++i) {
            if (a[i] == -1) continue; // 用-1标记已点亮的暗点
            ans++;
            int u = a[i]; // 记录最初的起始u点
            a[i] = -1; // 标记u点已经点亮
            for (int j = i + 1; j < n; ++j) {
                if (a[j] - u <= d) { // 距离小于d，点亮该暗点
                    if (a[j] != -1) {
                        u = a[j];
```

```
                        a[j] = -1;
                    }
                }
                else {
                    i = j - 1; // 更改起始u点位置
                    break;
                }
            }
        }
        printf("%d\n", ans);
    }
    return 0;
}
```