E2-G题题解

题目描述

What's happening?
What?! they're everywhere!

在无限大的二维平面上会出现 n 枚金币，第 i 枚金币仅在时刻 t_i 出现在整点 (x_i,y_i)。你能拿到第 i 枚金币，当且仅当你在时刻 t_i 恰好位于 (x_i,y_i)。

初始时你在 (0,0)，时刻为 t=0。每个时刻你可以选择保持不动，或是沿平行坐标轴的方向移动一单位距离。也即如果你在时刻 t 位于 (x,y)，则时刻 t+1 你必须位于 (x,y),(x−1,y),(x+1,y),(x,y−1),(x,y+1) 之一。

你最多能拿到多少枚金币呢?

输入

本题测试点包含多组数据。

第一行，一个正整数 T（1≤T≤10），表示数据组数。

对于每组数据：

第一行，一个正整数 n（1≤n≤2000），表示金币枚数。

接下来 n 行，每行三个整数 x_i,y_i,t_i（1≤x_i,y_i≤10^9，1≤t_i≤2×10^9），表示第 i 枚金币的位置与出现时间。

输出

对于每组数据：

输出一行，一个整数，表示最多能拿到的金币枚数。

思路分析

我们先按时间顺序对所有金币进行排序。

定义dp[i]数组含义为：对于前i枚金币，**在拿到第i枚金币的前提下**，最多可以拿到的金币数。记dp[0]=0。

再记r（0<=r<=i-1），对于第r枚金币，若**r和i间的距离（x坐标差值绝对值与y坐标差值绝对值之和）小于二者的时间差**，且**能够拿到第r枚金币**，则dp[i]的一个可能的值为dp[r]+1，表示在dp[r]的基础上多拿到了第i枚金币。

显然，dp[i]的值为max{dp[r]+1}，0<=r<=i-1，即所有满足要求的dp[r]+1中的最大值。

通过以上分析，可以得到转移方程：

dp[i] = max{dp[r]+1}

其中：0<=r<=i-1，且r满足**r和i间的距离（x坐标差值绝对值与y坐标差值绝对值之和）小于二者的时间差**，且**能够拿到第r枚金币。**

最后，我们只需要找到dp[i]（0<=i<=n中）的最大值，即为所求答案。

注意：我们需要使用一个数组enter[]，用于标记是否能获取某一金币。

时间复杂度O(n^2)，题目中n<=2000，满足要求！

核心代码

```
for (int i = 1; i <= n; i++) {
            for (int j = 0; j < i; j++) {
                if (flag_enter[j] && dis(i, j) <= coins[i].t - coins[j].t) {
                    dp[i] = MAX(dp[i], dp[j] + 1);
                    flag_enter[i] = true;
                }
            }
            ans = MAX(dp[i], ans);
        }
```

完整代码

```
#include <stdio.h>
#include <math.h>
#include <time.h>
#include <ctype.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>

#define MAX(a, b) (((a)>(b))?(a):(b))
#define MIN(a, b) (((a)<(b))?(a):(b))

typedef struct coins_s {
    int x, y, t;
} coins_s;

coins_s coins[2221];

int cmp(const void *a, const void *b) {
    return (*(coins_s *) a).t - (*(coins_s *) b).t;
}

int dis(int i, int j) {
    return abs(coins[i].x - coins[j].x) + abs(coins[i].y - coins[j].y);
}

int main() {
    int T;
    scanf("%d", &T);
    coins[0].x = 0, coins[0].y = 0, coins[0].t = 0;
    while (T--) {
        int ans = 0, n;
        scanf("%d", &n);
        for (int i = 1; i <= n; i++) {
            scanf("%d %d %d", &coins[i].x, &coins[i].y, &coins[i].t);
        }

        qsort(coins, n + 1, sizeof(coins[0]), cmp);

```

```
        int dp[2221] = {0};
        bool flag_enter[2221] = {true};

        for (int i = 1; i <= n; i++) {
            for (int j = 0; j < i; j++) {
                if (flag_enter[j] && dis(i, j) <= coins[i].t - coins[j].t)
{
                    dp[i] = MAX(dp[i], dp[j] + 1);
                    flag_enter[i] = true;
                }
            }
            ans = MAX(dp[i], ans);
        }
        printf("%d\n", ans);
    }
    return 0;
}
```