1. **题目重述：**

**已知人的初始坐标为5，可以左右移动，移动速度最大1m/s**

**坐标0~11的范围掉落n个馅饼，馅饼在第T秒掉落在坐标x处，可以同时掉落**

**求出在所有馅饼都落地之前，人和馅饼的坐标重合多少次（同一时间同一坐标）**

**二、最优子结构分析**

**人的位置坐标A(t+1)=A(t)+1或A(t)-1或A(t)，已知A(0) = 5。**

**馅饼的分布情况可以视为一个二维表：在第 t 秒，第 x 位置上有一个馅饼掉落**

**因此，当前时刻 t、当前所在位置 A 最多接住的馅饼数等于t-1 时刻的A或A-1或A+1位置的最优解加上当前位置A、当前时刻t掉落的馅饼数**

**问题可以用动态规划（DP）方法来求解**

**三、递归式**

**dp[A][t]=max(dp[A-1][t-1],dp[A][t-1],dp[A+1][t-1])+luo(A,t)**

**dp[A][t] 表示在第 t 秒站在位置 A 时，最多可以接到的馅饼数量**

**luo(A, t) 表示在第 t 秒、第 A 位置掉落的馅饼数量。**

**边界条件：当 A=0 时不能向左，A=10 时不能向右，超出范围时默认对应的 dp 值为 0**

**初始条件：dp[5][0] = 0，其他位置在时间 0 时初始为 0**

**四、完整代码**

**//本地运行**

**#include <iostream>**

**#include <algorithm>**

**using namespace std;**

**#define MAX\_N 10000**

**// 自定义的求三个数最大值的函数**

**int max(int num1, int num2, int num3) {**

**return max(num1, max(num2, num3));**

**}**

**// 统计在时间 t1 落在位置 x1 的馅饼数量**

**int luo(int x1, int t1, int X[], int T[], int n) {**

**int count = 0;**

**for (int i = 0; i < n; i++) {**

**if (X[i] == x1 && T[i] == t1) {**

**count++;**

**}**

**}**

**return count;**

**}**

**// 动态规划计算最大馅饼数量**

**int sum(int dp[11][MAX\_N], int X[], int T[], int n) {**

**int maxT = 0;**

**// 找出最大的时间 T**

**for (int i = 0; i < n; i++) {**

**if (T[i] > maxT) {**

**maxT = T[i];**

**}**

**}**

**for (int t = 1; t <= maxT; t++) {**

**for (int A = 0; A <= 10; A++) {**

**int prev1 = (A - 1 >= 0) ? dp[A - 1][t - 1] : 0;**

**int prev2 = dp[A][t - 1];**

**int prev3 = (A + 1 <= 10) ? dp[A + 1][t - 1] : 0;**

**dp[A][t] = max(prev1, prev2, prev3) + luo(A, t, X, T, n);**

**}**

**}**

**int ans = 0;**

**for (int A = 0; A <= 10; A++) {**

**ans = max(ans, dp[A][maxT]);**

**}**

**return ans;**

**}**

**int main() {**

**int n; // 馅饼个数**

**cin >> n;**

**int x[MAX\_N], T[MAX\_N];**

**for (int i = 0; i < n; i++) {**

**cin >> x[i] >> T[i];**

**}**

**int dp[11][MAX\_N] = {0};**

**dp[5][0] = 0;**

**int result = sum(dp, x, T, n);**

**cout << "能接到的最大馅饼数是: " << result << endl;**

**return 0;**

**}**

**//oj平台**

#include <iostream>

#include <algorithm>

using namespace std;

#define MAX\_N 100000

// 自定义的求三个数最大值的函数

int max(int num1, int num2, int num3) {

return std::max(num1, std::max(num2, num3));

}

// 统计在时间 t 落在位置 x 的馅饼数量

int luo(int x, int t, int X[], int T[], int n) {

int count = 0;

for (int i = 0; i < n; i++) {

if (X[i] == x && T[i] == t) {

count++;

}

}

return count;

}

// 动态规划计算最大馅饼数量

int sum(int dp[11][MAX\_N], int X[], int T[], int n) {

int maxT = 0;

// 找出最大的时间 T

for (int i = 0; i < n; i++) {

if (T[i] > maxT) {

maxT = T[i];

}

}

for (int t = 1; t <= maxT; t++) {

for (int A = 0; A <= 10; A++) {

int prev1 = (A - 1 >= 0 && dp[A - 1][t - 1] != -MAX\_N) ? dp[A - 1][t - 1] : -MAX\_N;

int prev2 = dp[A][t - 1] != -MAX\_N ? dp[A][t - 1] : -MAX\_N;

int prev3 = (A + 1 <= 10 && dp[A + 1][t - 1] != -MAX\_N) ? dp[A + 1][t - 1] : -MAX\_N;

dp[A][t] = max(prev1, prev2, prev3) + luo(A, t, X, T, n);

}

}

int ans = 0;

for (int A = 0; A <= 10; A++) {

if (dp[A][maxT] != -MAX\_N) {

ans = std::max(ans, dp[A][maxT]);

}

}

return ans;

}

int main() {

int n; // 馅饼个数

while (true) {

scanf("%d", &n);

if (n == 0) {

break;

}

int x[MAX\_N], T[MAX\_N];

for (int i = 0; i < n; i++) {

// 逐行读取坐标 x, 掉落时间 T

scanf("%d %d", &x[i], &T[i]);

}

int dp[11][MAX\_N];

for (int i = 0; i <= 10; i++) {

for (int j = 0; j < MAX\_N; j++) {

dp[i][j] = -MAX\_N;

}

}

// 初始位置在 5 处，t = 0 时没有馅饼掉落

dp[5][0] = 0;

int result = sum(dp, x, T, n);

printf("%d\n", result);

}

return 0;

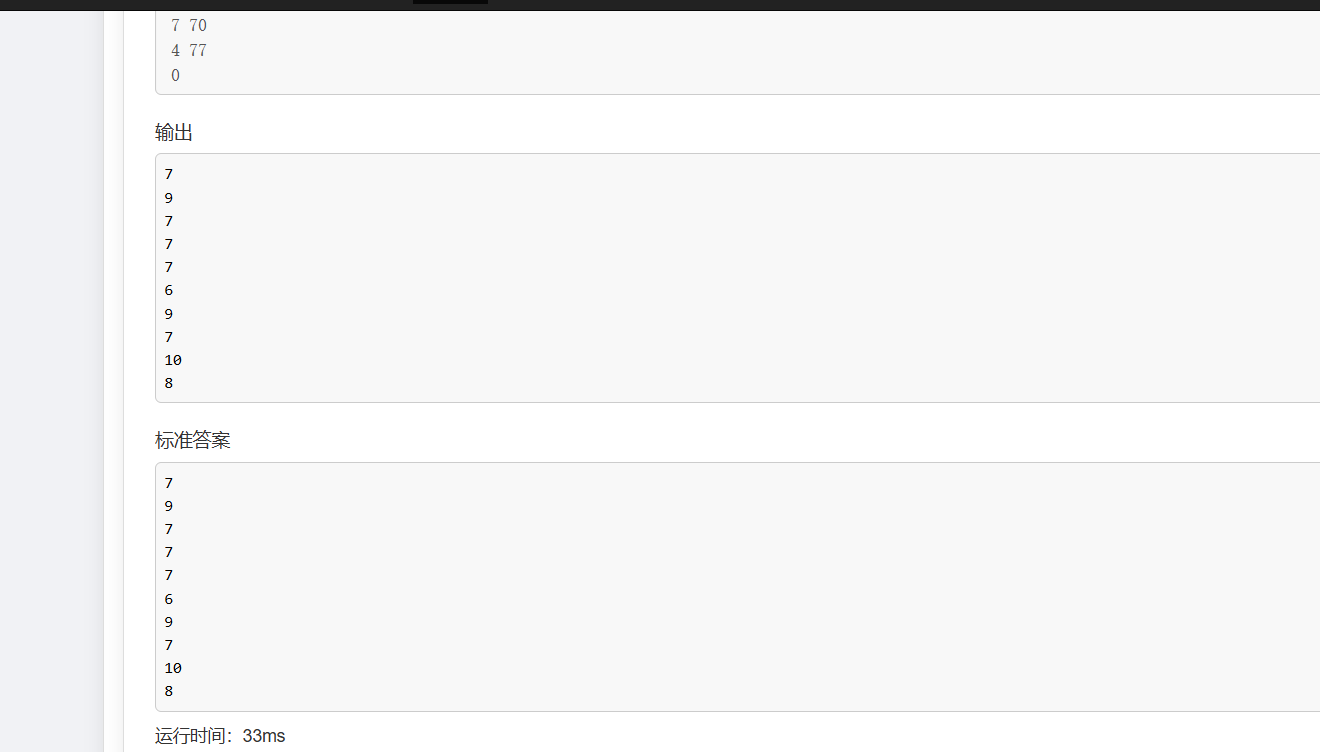
}

**五、执行结果**

**本地执行**

****

**oj平台小范围通过**

****

**https://www.acwing.com/problem/content/description/4555/**