习题课

TEACHING COURSEWARE POWPOINT

授课时间：2025.08.14

目录

01

走方格

TEACHING
COURSEWARE

TEACH

走方格

题目描述

在平面上有一些二维的点阵。

这些点的编号就像二维数组的编号一样，从上到下依次为第 1 至第 n 行，从左到右依次为第 1 至第 m 列，每一个点可以用行号和列号来表示。

现在有个人站在第 1 行第 1 列，要走到第 n 行第 m 列。只能向右或者向下走。

注意，如果行号和列数都是偶数，不能走入这一格中。

问有多少种方案。

输入格式

输入一行包含两个整数 n，m。

输出格式

输出一个整数，表示答案。

输入输出样例

输入 #1 复制	输出 #1 复制
3 4	2

说明/提示

$1 \leq n, m \leq 30$。

蓝桥杯 2020 第一轮省赛 A 组 G 题（B 组 H 题）。

走方格

状态

在这个问题中，dp[i][j] 表示从起点（1,1）走到（i，j）的所有有效路径数量。

i 代表当前所在的行号

j 代表当前所在的列号

dp[i][j] 的值就是到达（i，j）的方案总数

状态转移方程

由于只能向右或向下走，所以到达（i，j）的路径只能来自两个方向：

从上方的（i-1，j）向下走一步

从左方的（i，j-1）向右走一步

走方格


```cpp
#include<bits/stdc++.h>
using namespace std;
// dp[i][j]表示从(1,1)走到(i,j)的有效路径数量
int dp[35][35];
int main()
{
    int n, m;
    // 输入网格的行数n和列数m
    cin >> n >> m;
    // 初始化起点：从(1,1)到自身只有1种路径
    dp[1][1] = 1;
    // 遍历网格中的每个位置(i,j)
```

走方格


```
    // 遍历网格中的每个位置(i,j)
    for(int i = 1 ; i <= n ; i++)
    {
        for(int j = 1 ; j <= m ; j++)
        {
            // 特殊情况处理:
            // 1. 起点(1,1)已经初始化，不需要再计算
            // 2. 行列均为偶数的格子不可进入，跳过计算
            if( (i%2 == 0 && j%2 == 0) || (i == 1 && j == 1) )
            {
                continue;
            }
            // 状态转移方程:
            // 到达(i,j)的路径数 = 从上方(i-1,j)来的路径数 + 从左方(i,j-1)来的路径数
            dp[i][j] = dp[i-1][j] + dp[i][j-1];
        }
    }
    cout << dp[n][m] << endl;

    return 0;
}
```

02

开心的金明

TEACHING
COURSEWARE

TEACH

开心的金明

题目描述

金明今天很开心，家里购置的新房就要领钥匙了，新房里有一间他自己专用的很宽敞的房间。更让他高兴的是，妈妈昨天对他说："你的房间需要购买哪些物品，怎么布置，你说了算，只要不超过 N 元钱就行"。今天一早金明就开始做预算，但是他想买的东西太多了，肯定会超过妈妈限定的 N 元。于是，他把每件物品规定了一个重要度，分为 5 等：用整数 $1-5$ 表示，第 5 等最重要。他还从因特网上查到了每件物品的价格（都是整数元）。他希望在不超过 N 元（可以等于 N 元）的前提下，使每件物品的价格与重要度的乘积的总和最大。

设第j件物品的价格为 v_j，重要度为 w_j，共选中了 k 件物品，编号依次为 $j_1, j_2, \ldots, j_k$，则所求的总和为：

$$v_{j_1} \times w_{j_1} + v_{j_2} \times w_{j_2} \ldots + v_{j_k} \times w_{j_k}$$

请你帮助金明设计一个满足要求的购物单。

输入格式

第一行，为 2 个正整数，用一个空格隔开：n, m（$n < 30000, m < 25$）其中 n 表示总钱数，m 为希望购买物品的个数。

从第 2 行到第 $m+1$ 行，第 j 行给出了编号为 $j-1$ 的物品的基本数据，每行有 2 个非负整数 v, p（其中 v 表示该物品的价格 $(v \leq 10000)$，p 表示该物品的重要度（$1 \leq p \leq 5$）。

开心的金明

输出格式

1 个正整数，为不超过总钱数的物品的价格与重要度乘积的总和的最大值（< 100000000）。

输入输出样例

输入 #1 复制

```
1000 5
800 2
400 5
300 5
400 3
200 2
```

输出 #1 复制

```
3900
```

说明/提示

NOIP 2006 普及组 第二题

开心的金明

状态定义

dp[j]表示：用不超过j元钱所能买到的物品的价格与重要度乘积的总和的最大值

状态转移方程

对于每件物品i，考虑两种选择：买或不买

如果不买，dp[j]保持不变

如果买，dp[j] = max(dp[j], dp[j - v[i]] + v[i] * p[i])

其中v[i]是物品价格，p[i]是重要度

dp[j - v[i]]表示买了当前物品后剩余钱能获得的最大价值

v[i] * p[i]是当前物品的价值

开心的金明


```cpp
#include<bits/stdc++.h>
using namespace std;
// p数组存储物品的重要度
int p[30];
// v数组存储物品的价格
int v[30];
// dp[j]表示用不超过j元钱能获得的最大价值总和
int dp[300005];
int main()
{
    int n, m;
    // 输入总钱数n和物品数量m
    cin >> n >> m;
    // 输入每个物品的价格和重要度
    for(int i = 1 ; i <= m ; i++)
    {
        cin >> v[i] >> p[i];
    }
```

开心的金明


```
    // 遍历每件物品
    for(int i = 1 ; i <= m ; i++)
    {
        // 从总钱数n开始向下遍历到物品i的价格
        // 这种遍历方式确保每件物品只被考虑一次
        for(int j = n ; j >= v[i] ; j--)
        {
            // 状态转移方程：取不买该物品和买该物品两种情况的最大值
            dp[j] = max(dp[j], dp[j - v[i]] + v[i] * p[i]);
        }
    }

    // 输出用不超过n元钱能获得的最大价值总和
    cout << dp[n] << endl;

    return 0;
}
```

03

最大差值

TEACHING
COURSEWARE

TEACH

最大差值

题目描述

HKE 最近热衷于研究序列，有一次他发现了一个有趣的问题：

对于一个序列 $A_1, A_2, \cdots, A_n$，找出两个数 i, j $(1 \leq i < j \leq n)$，使得 $A_j - A_i$ 最大。

现在给出这个序列，请找出 $A_j - A_i$ 的最大值。

输入格式

第一行为一个正整数 n。

接下来 n 行，每行一个整数，第 $(i+1)$ 行的整数为 A_i。

输出格式

一行，为 $A_j - A_i$ 的最大值。

最大差值

输出格式

一行，为 $A_j - A_i$ 的最大值。

输入输出样例

输入 #1 复制

```
10
1
3
4
6
7
9
10
1
2
9
```

输出 #1 复制

```
9
```

说明/提示

数据规模与约定

- 对于 30% 的数据，$n \leq 1000$；
- 对于 70% 的数据，$n \leq 10^5$；
- 对于 100% 的数据：$2 \leq n \leq 10^6$，A_i 在 int 范围内。

最大差值

b[i]表示前i个元素中的最小值
dp[i]表示前i个元素中能形成的最大差值（即后面元素减去前面元素的最大值）。

状态转移方程
b[i]通过取前i-1个元素的最小值b[i-1]与当前元素a[i]的较小值来更新
dp[i]则通过比较前i-1个元素的最大差值dp[i-1]和当前元素a[i]减去前i-1个元素最小值b[i-1]的结果
取其中较大值来更新

最大差值


```cpp
#include <bits/stdc++.h>
using namespace std;
long long a[1000001];  // 存储原始序列
long long b[1000001];  // 存储前i个元素的最小值
long long dp[1000001]; // 存储前i个元素的最大差值
int main() {
    int n;
    cin >> n;
    // 读取数据
    for (int i = 1; i <= n; ++i) {
        cin >> a[i];
    }
    // 初始化状态
    b[1] = a[1];                    // 第一个元素的最小值是自身
    dp[1] = -1e18;                  // 第一个元素无法形成差值，初始化一个极小值
    // 状态转移
    for (int i = 2; i <= n; ++i) {
        b[i] = min(b[i-1], a[i]);               // 更新最小值状态
        dp[i] = max(dp[i-1], a[i] - b[i-1]); // 更新最大差值状态
    }
    cout << dp[n] << endl;
    return 0;
}
```

04

奶牛

TEACHING
COURSEWARE

TEACH

奶牛

题目描述

奶牛想证明它们是聪明而风趣的。为此，贝西筹备了一个奶牛博览会，她已经对 N 头奶牛进行了面试，确定了每头奶牛的智商和情商。

贝西有权选择让哪些奶牛参加展览。由于负的智商或情商会造成负面效果，所以贝西不希望出展奶牛的智商之和小于零，或情商之和小于零。满足这两个条件下，她希望出展奶牛的智商与情商之和越大越好，请帮助贝西求出这个最大值。

输入格式

第一行：单个整数 N，$1 \leq N \leq 400$。

第二行到第 $N+1$ 行：第 $i+1$ 行有两个整数：S_i 和 F_i，表示第 i 头奶牛的智商和情商，$-1000 \leq S_i; F_i \leq 1000$。

输出格式

输出单个整数：表示情商与智商和的最大值。贝西可以不让任何奶牛参加展览，如果这样做是最好的，输出 0。

奶牛

输入格式

第一行：单个整数 N，$1 \leq N \leq 400$。

第二行到第 $N+1$ 行：第 $i+1$ 行有两个整数：S_i 和 F_i，表示第 i 头奶牛的智商和情商，$-1000 \leq S_i; F_i \leq 1000$。

输出格式

输出单个整数：表示情商与智商和的最大值。贝西可以不让任何奶牛参加展览，如果这样做是最好的，输出 0。

输入输出样例

输入 #1 复制

```
5
-5 7
8 -6
6 -3
2 1
-8 -5
```

输出 #1 复制

```
8
```

说明/提示

选择第一头，第三头，第四头奶牛，智商和为-5+6+2 = 3，情商和为7-3+1 = 5。再加

入第二号奶牛可使总和提升到10，不过由于情商和变成负的了，所以是不允许的

奶牛

状态定义

定义dp[j]表示：当智商总和为(j - 400000)时，能获得的最大情商总和。

偏移量400000的作用：智商总和可能为负数（每头奶牛智商≥-1000，400 头奶牛最小总和为 -400000），用j = 智商总和 + 400000将负索引转为非负索引（j范围为 0~800000，覆盖所有可能的智商总和）。

例如：j=400000对应智商总和 0；j=0对应智商总和 - 400000；j=800000对应智商总和 400000。

$$dp[j]=max(dp[j],dp[j-s[i]]+f[i])$$

奶牛　　dp[j]=max(dp[j],dp[j-s[i]]+f[i])

01 背包要求每个物品只能选一次，遍历顺序需避免 “重复选择”。关键在于物品 “重量”（此处为S_i）的正负：

当S_i ≥ 0（智商为非负）

若正序遍历j（从S_i到 800000），会导致同一奶牛被多次选择。

假设S_i=2，当j=2时，j'=4会用j=2（已选当前奶牛）的状态更新，导致重复选择。

因此需倒序遍历（从 800000 到S_i）：此时j从大到小，j - S_i（原索引）尚未被当前奶牛更新，保证每个奶牛只选一次。

当S_i < 0（智商为负）

此时j' = j + S_i = j - |S_i|（j' < j）。若倒序遍历j（从大到小），j'会在j之后被处理，可能用已选当前奶牛的状态更新，导致重复选择。

因此需正序遍历（从 0 到800000 + S_i）：此时j从小到大，j'（j + S_i）已被处理且未受当前奶牛影响，保证每个奶牛只选一次。

奶牛

情况：当前奶牛的智商是负数（S_i ＜ 0）

比如这头奶牛的智商是 − 2（S_i = −2），情商不管多少。
当我们考虑选这头奶牛时：

假设原来的智商总和对应索引是j（比如j=10）

选了之后，新的索引是j' = j + S_i = 10 + (−2) = 8（j'比j小）

为什么不能用倒序遍历（从大到小）？

如果我们从大到小遍历j（比如先算j=10，再算j=9，再算j=8. . .）：

当处理j=10时，我们会更新j'=8的状态（把这头奶牛算进去）。

接下来处理j=8时，注意此时j=8已经被刚才的操作更新过了（已经包含了这头奶牛）。

如果再用j=8去更新j'=6（8 + (−2) = 6），就相当于这头奶牛被选了两次

这就违反了 01 背包 “每个物品只能选一次” 的规则。

奶牛


```cpp
#include <iostream>
#include <algorithm>
using namespace std;
int s[405], f[405];  // 数组从1开始使用
int dp[800001];      // 数组大小为800001，索引从0到800000
int n, ans = 0;
int main() {
    cin >> n;
    for (int i = 1; i <= n; i++) {  // 奶牛编号从1开始
        cin >> s[i] >> f[i];
    }
    // 初始化dp数组，全部设为极小值
    for (int i = 1; i <= 800000; i++) {  // 从1开始初始化到800000
        dp[i] = -1e9;
    }
```

奶牛


```
dp[400000] = 0;  // 偏移量400000，对应智商总和为0
for (int i = 1; i <= n; i++) {
    if (s[i] > 0) {
        // 智商为正时，倒序遍历防止重复选择
        for (int j = 800000; j >= s[i]; j--) {
            dp[j] = max(dp[j], dp[j - s[i]] + f[i]);
        }
    } else {
        // 智商为负或零时，正序遍历防止重复选择
        for (int j = 1; j <= 800000 + s[i]; j++) {  // 从1开始遍历
            dp[j] = max(dp[j], dp[j - s[i]] + f[i]);
        }
    }
}
```

奶牛


```
    // 寻找最优解：智商和非负且情商和非负的情况下，两者之和最大
    for (int i = 400000; i <= 800000; i++) {
        if (dp[i] >= 0) {
            ans = max(ans, (i - 400000) + dp[i]);
        }
    }
    cout << ans << endl;
    return 0;
}
```