贪心习题

TEACHING COURSEWARE POWPOINT

授课时间：20XX.01.01

目录

01

题目描述

TEACHING
COURSEWARE

TEACH

通用模版

题目描述

又是一年秋季时，陶陶家的苹果树结了 n 个果子。陶陶又跑去摘苹果，这次他有一个 a 公分的椅子。当他手够不着时，他会站到椅子上再试试。

这次与 NOIp2005 普及组第一题不同的是：陶陶之前搬凳子，力气只剩下 s 了。当然，每次摘苹果时都要用一定的力气。陶陶想知道在 $s < 0$ 之前最多能摘到多少个苹果。

现在已知 n 个苹果到达地上的高度 x_i，椅子的高度 a，陶陶手伸直的最大长度 b，陶陶所剩的力气 s，陶陶摘一个苹果需要的力气 y_i，求陶陶最多能摘到多少个苹果。

输入格式

第 1 行：两个数 苹果数 n，力气 s。

第 2 行：两个数 椅子的高度 a，陶陶手伸直的最大长度 b。

第 3 行~第 $3+n-1$ 行：每行两个数 苹果高度 x_i，摘这个苹果需要的力气 y_i。

输出格式

只有一个整数，表示陶陶最多能摘到的苹果数。

题目描述

输入输出样例

输入 #1 复制

```
8 15
20 130
120 3
150 2
110 7
180 1
50 8
200 0
140 3
120 2
```

输出 #1 复制

```
4
```

说明/提示

对于 100% 的数据，$n \leq 5000, a \leq 50, b \leq 200, s \leq 1000, x_i \leq 280, y_i \leq 100$。

02

知识回归

TEACHING
COURSEWARE

TEACH

知识回归

贪心，不考虑未来，不后悔之前，
总结成四个字的性质：

03

解题思路

TEACHING
COURSEWARE

TEACH

解题思路

贪心。核心思想是：在能摘取的苹果范围内（高度不超过椅子高度与手伸直长度之和），优先摘取所需力气小的苹果。因为这样可以用最少的力气消耗，获取最多的苹果数量，符合贪心算法追求当前局部最优以达成全局最优的特点。

04

标准程序

TEACHING
COURSEWARE

TEACH

标准程序


```cpp
#include <iostream>
#include <algorithm>
using namespace std;
// 定义结构体存储苹果的高度和所需力气
struct node {
    int h;  // 苹果高度
    int li; // 摘取该苹果需要的力气
};
// 排序比较函数：按所需力气从小到大排序
bool cmp(node a,node b) {
    return a.li < b.li;
}
```

标准程序


```
int main() {
    int n, s;           // n: 苹果总数; s: 陶陶剩余的力气
    int a, b;           // a: 椅子高度; b: 手伸直长度, 可及高度 = a + b
    cin >> n >> s;
    cin >> a >> b;
    node apple[5005];   // 数组下标从1开始使用, 最多存储5000个苹果
    int count = 0;      // 记录可摘取苹果的数量（最终下标范围1~count）
    // 筛选可及的苹果, 存储到数组的1~count位置
    for (int i = 0; i < n; i++) {
        int h, w;       // 当前苹果的高度和所需力气
        cin >> h >> w;
        // 判断苹果是否可及（高度 <= 可及高度）
        if (h <= a + b) {
            count++;    // 先增加计数, 使下标从1开始
            apple[count].h = h;  // 存储到第count个位置
            apple[count].li = w;
        }
    }
```

标准程序


```
    // 对1~count范围内的苹果按力气从小到大排序
    // sort函数的第二个参数是结束位置的下一个，所以是apple + count + 1
    sort(apple + 1, apple + count + 1, cmp);
    int remaining = s;  // 剩余力气
    int ans = 0;        // 摘到的苹果总数
    // 从第1个到第count个苹果依次尝试摘取
    for (int i = 1; i <= count; i++) {
        // 如果剩余力气足够摘当前苹果
        if (remaining >= apple[i].li) {
            ans++;
            remaining -= apple[i].li;
        } else {
            // 力气不足，后续苹果力气更大，直接退出
            break;
        }
    }
    cout << ans << endl;
    return 0;
}
```