307-08-背包

P2160 [SHOI2007]书柜的尺寸

题目描述

Tom不喜欢那种一字长龙式的大书架，他只想要一个小书柜来存放他的系列工具书。
Tom打算把书柜放在桌子的后面，这样需要查书的时候就可以不用起身离开了。

显然，这种书柜不能太大，Tom希望它的体积越小越好。另外，出于他的审美要求，他只想要一个三层的书柜。为了物尽其用，Tom规定每层必须至少放一本书。现在的问题是，Tom怎么分配他的工具书，才能让木匠造出最小的书柜来呢?

Tom很快意识到这是一个数学问题。每本书都有自己的高度hi和厚度ti。我们需要求的是一个分配方案，也就是要求把所有的书分配在S1、S2和S3三个非空集合里面的一个，不重复也不遗漏，那么，很明显，书柜正面表面积（S）的计算公式就是:

$$S=\left(\sum_{j=1}^{3}\max_{i\in S_j}h_i\right)\times\left(\max_{j=1}^{3}\sum_{i\in S_j}t_i\right) \tag{1}$$

由于书柜的深度是固定的（显然，它应该等于那本最宽的书的长度），所以要求书柜的体积最小就是要求S最小。Tom离答案只有一步之遥了。不过很遗憾，Tom并不擅长于编程，于是他邀请你来帮助他解决这个问题。

输入输出格式

输入格式:

文件的第一行只有一个整数n（3≤n≤70），代表书本的本数。
接下来有n行，每行有两个整数hi和ti，代表每本书的高度和厚度，我们保证150≤hi≤300，5≤ti≤30。

输出格式:

只有一行，即输出最小的S。

输入输出样例

输入样例#1: 复制

```
4
220 29
```

```
195 20
200 9
180 30
```

输出样例#1： 复制

```
18000
```

题解

首先将书按照高度

$$f[i][j][k][l] = \min \begin{cases} f[i-1][j-t[i]][k][l] + (j == t[i]) * h[i] \\ f[i-1][j][k-t[i]][l] + (k == t[i]) * h[i] \\ f[i-1][j][k][l-t[i]] + (l == t[i]) * h[i] \end{cases} \tag{2}$$

但是我们可以发现这样空间复杂度会太高，所以为了解决这个问题，可以有以下做法：

1. 最后一维可以同过维护前缀和：$l = s[i] - j - k$来得到，可以省略一维
2. 第一维i可以用滚动数组滚掉

代码：

```cpp
#include <iostream>
#include <cstring>
#include <algorithm>

using namespace std;
typedef long long ll;

const ll maxn = 2105;

struct node {
    ll h, t;
    friend bool operator < (const node &a, const node &b) {
        return a.h > b.h;
    }
} a[maxn];

ll n, s[maxn], f[maxn][maxn];

int main() {
    cin >> n;
    for (ll i = 1; i <= n; i ++)
        cin >> a[i].h >> a[i].t;
    sort(a + 1, a + n + 1);
    for (ll i = 1; i <= n; i ++)
        s[i] = s[i - 1] + a[i].t;
    ll sum = s[n];
```

```
    memset(f, 0x3f, sizeof(f));
    f[0][0] = 0;
    for (ll i = 1; i <= n; i ++) {
        for (ll j = sum; j >= 0; j --) {
            for (ll k = sum - j; k >= 0; k --) {
                ll x1 = j - a[i].t;
                ll x2 = k - a[i].t;
                ll x3 = s[i] - j - k - a[i].t;
                ll t1 = 1e9, t2 = 1e9, t3 = 1e9;
                if (x1 >= 0) t1 = f[x1][k];
                if (x1 == 0) t1 += a[i].h;
                if (x2 >= 0) t2 = f[j][x2];
                if (x2 == 0) t2 += a[i].h;
                if (x3 >= 0) t3 = f[j][k];
                if (x3 == 0) t3 += a[i].h;
                f[j][k] = min(t1, min(t2, t3));
            }
        }
    }
    ll ans = 8e18;
    // i, j, l至少为1
    for (ll i = 1; i <= sum - 2; i ++) {
        for (ll j = 1; i + j < sum; j ++) {
            ans = min(ans, max(i, max(j, sum - i - j)) * f[i][j]);
        }
    }
    cout << ans << endl;
    return 0;
}
```

P2967 [USACO09DEC]视频游戏的麻烦Video Game Troubles

题意翻译

农夫约翰的奶牛们打游戏上瘾了！本来约翰是想要按照调教兽的做法拿她们去电击戒瘾的，可后来他发现奶牛们玩游戏之后比原先产更多的奶。很明显，这是因为满足的牛会产更多的奶。

但是，奶牛们因何者为最好的游戏主机而吵得不可开交。约翰想要在给定的预算内购入一些游戏平台和一些游戏，使他的奶牛们生产最多的奶牛以养育最多的小牛。

约翰考察了 N 种游戏主机，第 i 种主机的价格是 P_i，该主机有 G_i 个独占游戏。很明显，奶牛必须先买进一种游戏主机，才能买进在这种主机上运行的游戏。在每种主机中，游戏 j 的价格为 ，GP_j每头奶牛在玩了该游戏后的牛奶产量为 PV_j。

农夫约翰的预算为 V。请帮助他确定应该买什么游戏主机和游戏，使得他能够获得的产出值的和最大。

题解

定义：

$h[i]$：总共花费i元可以获得的最大产出

$f[i]$：花费i元且购买当前游戏机所获得的最大产出

背包问题方程：

$$f[i] = \max\{f[i],\ f[i - gp[j]] + pv[j]\} \tag{4}$$

则可以有以下过程：

每次循环：

1. 将上一轮的$h[i]$赋值给$f[i]$
2. 对于$f[i]$做背包问题
3. 对于每一个$h[i]$，比较$f[i-p]$与$h[i]$，之所以要$i-p$：必须购买游戏机，游戏机要花p元。即更新后的$h[i] = \max\{h[i],\ f[i-p]\}$

注：在整个过程中，要注意边界的问题。

代码：

```cpp
#include <iostream>

using namespace std;

const int maxn = 1000005;

int n, v;

int f[maxn], h[maxn];

int main() {
    cin >> n >> v;
    while (n --> 0) {
        int p, g; cin >> p >> g;
        for (int i = v; i >= 0; i --) f[i] = h[i];
        while (g --> 0) {
            int gp, pv; cin >> gp >> pv;
            for (int i = v; i >= gp; i --)
```

```
                f[i] = max(f[i], f[i - gp] + pv);
        }
        for (int i = v; i >= p; i --)
            h[i] = max(h[i], f[i - p]);
    }
    cout << h[v] << endl;
    return 0;
}
```