题意：有一个人要抢银行，有n家银行，第i个银行有m[i]的财富，且抢第i个银行被抓的概率是p[i]，如果这个人抢银行被抓的概率不超过P，就是安全的，问保证安全的情况下能抢到的最大财富值是多少，每个银行只能被抢一次。

解题思路：银行1个1个的抢过去，抢到的钱随着抢劫银行的数量增加或者不变（不变的情况就是不抢劫该银行），被捕的概率随着银行数量的增加而增大或者不变（不变的情况就是不抢劫该银行）

所以这个题有三个量在改变，一个是银行的数量，一个是抢到的钱，还有一个是被捕的概率

所以用二维dp来表示三个变量，设dp[i][j]表示抢劫了前i个银行，抢了j的钱被捕的概率

那么dp[i][j] = min(dp[i-1][j],dp[i-1][j-money[i]] + (dp[i-1][j-money]) \* rob[i])

其中money[i]表示第i个银行的有多少钱，rob[i]表示抢劫第i个银行被捕的概率

一看n是100，贪心的话n肯定不可能是100

背包

#include<cstdio>

#include<algorithm>

#include<cmath>

#include<iostream>

#include<cstring>

using namespace std;

struct node

{

int m;

double p;

}a[110];

double dp[110][10010];

int main()

{

//freopen("input.txt","r",stdin);

int T,n;

double p;

scanf("%d",&T);

for(int cas=1;cas<=T;cas++)

{

scanf("%lf%d",&p,&n);

int sum(0);

for(int i=1;i<=n;i++)

{

scanf("%d%lf",&a[i].m,&a[i].p);

sum+=a[i].m;

}

for(int i=1;i<=n;i++)

for(int j=0;j<=sum;j++)

dp[i][j]=0;

for(int i=0;i<=n;i++)//注意要从0开始，否则在i=1的时候更新就会出问题

dp[i][0]=1;

p=1-p;

int i;

for(i=1;i<=n;i++)

{

for(int j=0;j<=sum;j++)

if(j<a[i].m)

dp[i][j]=dp[i-1][j];

else

dp[i][j]=max(dp[i-1][j],dp[i-1][j-a[i].m]\*(1-a[i].p));

}

//因为本题最后要看dp[n]的可能是0~sum所有状态，所以<a[i].m的状态也要更新

for(int i=sum;i>=0;i--)

{

if(dp[n][i]>=p)

{

printf("Case %d: %d\n",cas,i);

break;

}

}

}

return 0;

}

一维dp数组：

#include<cstdio>

#include<algorithm>

#include<cmath>

#include<iostream>

#include<cstring>

using namespace std;

struct node

{

int m;

double p;

} a[110];

double dp[10010];

int main()

{

//freopen("input.txt","r",stdin);

int T,n;

double p;

scanf("%d",&T);

for(int cas=1; cas<=T; cas++)

{

scanf("%lf%d",&p,&n);

int sum(0);

for(int i=1; i<=n; i++)

{

scanf("%d%lf",&a[i].m,&a[i].p);

sum+=a[i].m;

}

for(int j=0; j<=sum; j++)

dp[j]=0;

dp[0]=1;

p=1-p;

for(int i=1; i<=n; i++) //虽然背包是一维的了，但是这层循环也要带，因为还有a[i].p和a[i].m要处理

for(int j=sum; j>=a[i].m; j--)//注意倒序！

dp[j]=max(dp[j],dp[j-a[i].m]\*(1-a[i].p));

for(int i=sum; i>=0; i--)

{

if(dp[i]>=p)

{

printf("Case %d: %d\n",cas,i);

break;

}

}

}

return 0;

}