Ông A cần đi qua 1 đoạn đường B.  
Trên đoạn đường đi qua có N cổng. Tại mỗi cổng có 1 số lượng binh sĩ và giá để đi qua cổng đó. Muốn đi qua mỗi cổng ông A có 3 cách lựa chọn.  
1. Pass  
Trả số tiền quy định ở cổng đó để được đi qua  
2. Hire  
Trả gấp đôi số tiền ở cổng đó để thuê số binh sĩ gộp vào đoàn quân của mình  
3. battle  
Điều kiện để đánh nhau là số quân của ông A >= số lượng lính tại cổng đó. Có các lưu ý:  
+ Ông A k được tính vào số lượng của quân  
+ Mỗi người lính chỉ tham gia được vào tối đa 3 trận đánh. Sau 3 trận đánh nếu đi nhóm binh sĩ đó còn sống thì cũng giải tán.  
+ Mỗi trận đánh thì tất cả số binh sĩ đều tham gia.  
+ Đánh nhau chết theo tỉ lệ 1: 1. Ai tham gia trước sẽ bị chết trước  
  
Điều kiện input: số cổng <=20  
-Số lính và chi phí đi qua >=2 và <=1000  
Tìm chi phí nhỏ nhất để ông A có thể đi qua đoạn đường B  
  
VD: Có 7 cổng  
 1 2 3 4 5 6 7  
Số binh sĩ 10 70 80 20 50 30 10  
Chi phí 100 5 15 60 90 80 10

- Dòng đầu tiên cho số lượng trường hợp thử nghiệm T (T <= 50)

- Trong mỗi trường hợp thử nghiệm:

  + Dòng đầu tiên được cho số nhóm orc Frodo sẽ va vào N (5 <= N <= 20)

  + N dòng tiếp theo sẽ cung cấp thông tin của nhóm orc:

   > Số lượng Orc trong nhóm G đó

   > Phí cầu đường phải vượt qua nhóm đó

[Đầu ra]

- Chi phí tối thiểu để đưa Frodo đến đích.  
  
  
  
  
Có thể tính chi phí đi nhỏ nhất  
 1 2 3 4 5 6 7  
Số binh sĩ 10 70 80 20 50 30 10  
Chi phí 100 5 15 60 90 80 10  
Chọn pp Pass Hire Hire Battle Battle Battle Pass  
Chi phí 100 110 140 150

5  
7  
10 100  
70 5  
80 15  
20 60  
50 90  
30 80  
10 10  
9  
600 800  
300 400  
300 400  
1000 400  
300 600  
100 300  
600 300  
600 500  
1000 300  
11  
1000 10  
700 900  
400 500  
300 10  
900 900  
300 10  
50 900  
50 900  
700 900  
500 900  
50 10  
20  
896 546  
543 216  
454 310  
408 367  
40 602  
252 582  
954 627  
850 234  
763 479  
232 278  
301 538  
528 508  
936 154  
629 443  
758 336  
432 700  
882 256  
278 738  
517 882  
317 136  
20  
410 610  
831 909  
675 629  
421 774  
386 869  
544 219  
492 414  
996 557  
499 482  
231 285  
804 978  
304 881  
489 911  
75 315  
927 648  
252 914  
330 396  
937 133  
495 882  
813 717

#1 150  
#2 3000  
#3 2370  
#4 4721  
#5 8231

[#include](tg://search_hashtag?hashtag=include) <iostream>  
[#define](tg://search_hashtag?hashtag=define) SIZE 100  
using namespace std;  
int SoCong;  
int Cong[20][2];  
int QuanSo1[20][2], Tien;  
int Answer;  
int CachDi[20];  
int cnt = 0;  
void BackTrack(int Buoc, int SoTien, int SoLinh){  
cnt = cnt +1;  
  
if(SoTien >= Answer)  
return;  
  
if(Buoc == SoCong){  
if(SoTien < Answer)  
Answer = SoTien;  
/\*for(int i = 0; i < SoCong; i++){  
cout << CachDi[i] << " ";   
}  
cout << "\t" << SoTien << endl;\*/  
return;  
}  
  
for(int i = 0; i < 3; i++){  
// Pass  
if(i == 1){  
CachDi[Buoc] = 1;  
BackTrack(Buoc+1, SoTien + Cong[Buoc][1], SoLinh);  
CachDi[Buoc] = 0;  
}  
// Hire  
else if(i == 2){  
CachDi[Buoc] = 2;  
QuanSo1[Buoc][0] = Cong[Buoc][0];  
QuanSo1[Buoc][1] = 3;  
  
BackTrack(Buoc+1, SoTien + Cong[Buoc][1]\*2, SoLinh+Cong[Buoc][0]);  
  
QuanSo1[Buoc][0] = 0;  
QuanSo1[Buoc][1] = 0;  
CachDi[Buoc] = 0;  
}  
// Atk  
else{  
if(SoLinh >= Cong[Buoc][0]){  
int temp = SoLinh;  
int dich = Cong[Buoc][0];  
int luu[20][2];  
  
for(int k = 0; k < Buoc; k++){  
luu[k][0] = QuanSo1[k][0];  
luu[k][1] = QuanSo1[k][1];  
}  
  
for(int k = 0; k < Buoc; k++){  
if(QuanSo1[k][0] > 0 && QuanSo1[k][1] > 0){  
if(dich >= QuanSo1[k][0]){  
dich = dich - QuanSo1[k][0];  
QuanSo1[k][0] = 0;  
}  
else{  
QuanSo1[k][0] = QuanSo1[k][0] - dich;   
dich = 0;  
}  
}  
QuanSo1[k][1]--;  
if(QuanSo1[k][1] == 0){  
QuanSo1[k][0] = 0;  
}  
}  
CachDi[Buoc] = 3;  
SoLinh = 0;  
for(int t = 0; t < Buoc; t++){  
if(QuanSo1[t][1] > 0)  
SoLinh = SoLinh + QuanSo1[t][0];  
}  
BackTrack(Buoc+1, SoTien, SoLinh);  
  
SoLinh = temp;  
CachDi[Buoc] = 0;  
for(int k = 0; k < Buoc; k++){  
QuanSo1[k][0] = luu[k][0];  
QuanSo1[k][1] = luu[k][1];  
}  
}  
}  
}  
}  
int main()  
{  
int tc,TestCase;  
  
freopen("Text.txt", "r", stdin);  
freopen("Text1.txt", "w", stdout);  
cin>>TestCase;  
  
for(tc=0; tc<TestCase; tc++)  
{  
Answer = 10000000;  
  
Tien = 0;  
cnt = 0;  
cin >> SoCong;  
for(int i = 0; i < SoCong; i++){  
cin >> Cong[i][0] >> Cong[i][1];  
CachDi[i] = 0;  
QuanSo1[i][0] = 0;  
QuanSo1[i][1] = 0;  
}  
  
BackTrack(0, 0, 0);  
////////////////////////////////  
  
  
  
////////////////////////////////  
  
  
  
cout<< "#" << tc+1 <<" "<< Answer << endl;  
}  
  
return 0;  
}

package QuaCong;  
  
import java.io.FileInputStream;  
import java.util.Scanner;  
  
public class QuaCong {  
static int N;  
static int[] pawn;  
static int[] gold;  
static int min;  
  
public static void main(String args[]) throws Exception {  
System.setIn(new FileInputStream("src/QuaCong/input"));  
Scanner sc = new Scanner([System.in](http://system.in/));  
int T = sc.nextInt();  
for (int test = 1; test <= T; test++) {  
N = sc.nextInt();  
pawn = new int[N];  
gold = new int[N];  
for (int i = 0; i < N; i++) {  
pawn[i] = sc.nextInt();  
gold[i] = sc.nextInt();  
  
}  
min = 99999;  
BT(0, 0, 0, 0, 0);  
System.out.println(min);  
}  
sc.close();  
}  
  
static void BT(int index, int sumGold, int pawn0, int pawn1, int pawn2) {  
if (index == N) {  
if (sumGold < min) {  
min = sumGold;  
}  
} else {  
if (sumGold <= min)  
for (int i = 0; i < 3; i++) {  
if (i == 0) {  
BT(index + 1, sumGold + gold[index], pawn0, pawn1, pawn2);  
} else if (i == 1) {  
BT(index + 1, sumGold + 2 \* gold[index], pawn0 + pawn[index], pawn1, pawn2);  
} else {  
if (pawn0 + pawn1 + pawn2 >= pawn[index]) {  
if (pawn2 >= pawn[index]) {  
BT(index + 1, sumGold, 0, pawn0, pawn1);  
} else if (pawn1 + pawn2 >= pawn[index]) {  
BT(index + 1, sumGold, 0, pawn0, pawn1 + pawn2 - pawn[index]);  
} else {  
BT(index + 1, sumGold, 0, pawn0 + pawn1 + pawn2 - pawn[index], 0);  
}  
}  
}  
}  
}  
}  
}

Top of Form

Bottom of Form