**★★★☆☆**

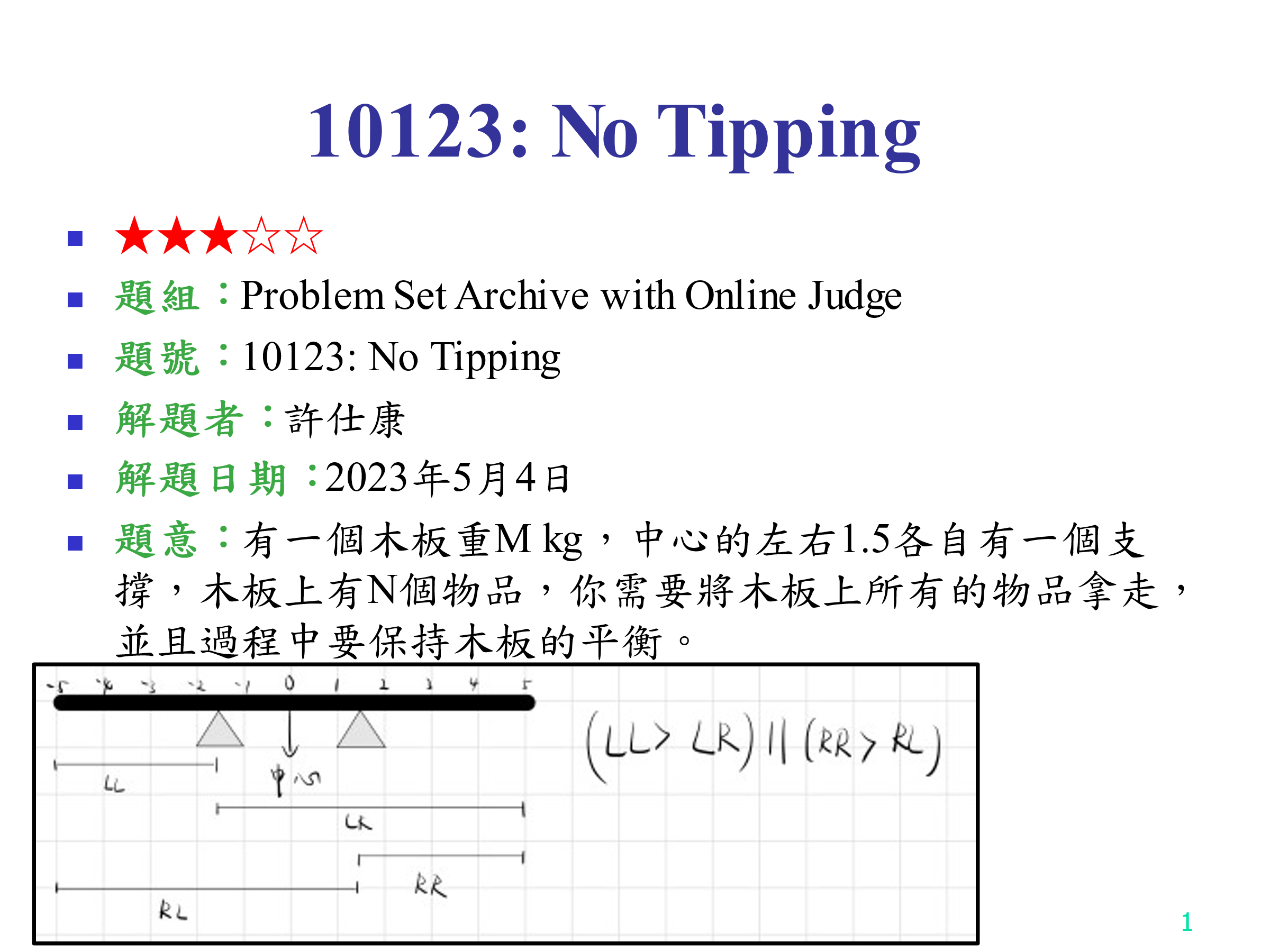
**題組：Problem Set Archive** **with Online Judge**

**題號：10123: No Tipping**

**解題者：許仕康**

**解題日期：2023年5月4日**

**題意：**

有一個木板重M kg，中心的左右1.5各自有一個支撐，木板上有N個物品，你需要將木板上所有的物品拿走，並且過程中要保持木板的平衡。若LL>LR或者RR>RL就發生傾斜。

**題意範例：**

輸入：

20 3 6 L(木板長度） M（木板重量） N（物品數量）

-8 4 物品在的位置 物品重量

-4 10

-3 10

2 4

5 7

8 8

30 10 2

-8 100

9 91

0 0 0

輸出：

Case 1:

-4 10

8 8

-8 4

5 7

-3 10

2 4

Case 2:

Impossible

**解法：**

力矩公式：力\*物品與支點的距離

把題目想成一個空的木板，目標是把所有物品放上去，最後只要能保持平衡就代表題目有解，反之無解。根據放置的順序反向輸出即可。

處於兩個支點之間的物品有增加穩固性的功能，所以最先放入，其放入順序根據輸入的順序反著來。

想要把物品放上木板，且保持平衡，就需要放入對木板影響最小的物品，也就是放上力矩最小的物品

將物品分類左半邊和右半邊，算出各個物品的力矩，力矩大小順序排列。假如左邊某個物品的力矩最小就先放入，接著繼續放左邊的物品，直到發現繼續放左邊的物品傾斜發生就停止，放右邊的物品，以此類推，反之亦然。

如果發現左邊右邊的物品都不能使木板平衡就代表impossible所有物品都放上去且保持平衡就代表題目有解。

**解法範例：**

無

**討論：**

網站上case 2的output是錯的。

**程式：**

#include<iostream>

#include<algorithm>

#include<queue>

#include<stack>

using namespace std;

double ll,lr,rl,rr;//ll,lr,rl,rr就如題意解釋中

struct things{

int p;

int w;

int lj;

things(int po,int we, double liju){//po是位置，we是重量，liju是力矩

p=po;w=we;lj=liju;

}

things(){};

};

struct cmp{

bool operator()(things const &t1,things const &t2){

return t1.lj>t2.lj;

}

};

bool test(things const &q){//測試有沒有ll>lr或者rr>rl

double ll1=ll,rl1=rl,rr1=rr,lr1=lr;

if(q.p<0){

ll+=q.w\*(abs(q.p)-1.5);

rl+=q.w\*(abs(q.p)+1.5);

}

else{

rr+=q.w\*(abs(q.p)-1.5);

lr+=q.w\*(abs(q.p)+1.5);

}

//cout<<"ll:"<<ll<<" lr:"<<lr<<" rr:"<<rr<<" rl:"<<rl;

//cout<<" qp:"<<q.p<<" qw:"<<q.w<<endl;

if(ll>lr||rr>rl){

ll=ll1;rl=rl1;rr=rr1;lr=lr1;

return false;

}

return true;

}

int main(){

int a,b,c,time=0;

while(1) {

time++;

cin>>a>>b>>c;

if(a==0&&b==0&&c==0)break;

ll=0;lr=1.5\*b;rl=1.5\*b;rr=0;

priority\_queue<things, vector<things>, cmp> ql;

priority\_queue<things, vector<things>, cmp> qr;

stack <things> s;

int fx=1;//測試可以放嗎，初始值是1，如果其中一邊不可以放就會變成0，兩邊不能放就會變成-1，每次可以放的時候都會重新把值變成1

for(int i=0;i<c;i++){

int position,weight;

double liju;

cin>>position>>weight;

if((position<1.5&&position>0)||(position>-1.5&&position<0)){

things t(position,weight,0);

s.push(t);

if(position<0){

rl+=(abs(position)+1.5)\*weight;

lr+=(position+1.5)\*weight;

}

else {

lr+=(abs(position)+1.5)\*weight;

rl+=(1.5-position)\*weight;

}

i--;c--;

}

else{

liju=abs(position)\*weight;

things t(position,weight,liju);

if(t.p<0)ql.push(t);

else qr.push(t);

}

}

int fangxiang;//看現在是要放左邊還是放右邊

for(int i=0;i<c;i++){

if(fx<-1)break; 如果fx<=-1的話就是兩邊都不能放了，都會傾斜

else if(i==0){

if(ql.empty()){

if(test(qr.top())){

s.push(qr.top());

qr.pop();

fangxiang=1;//fangxiang為1的時候放右邊

}

else{

fx=-1;

break;

}

}

else if(qr.empty()){

if(test(ql.top())){

s.push((ql.top()));

ql.pop();

fangxiang=-1;//fangxiang是-1的時候是放左邊

}

else{

fx=-1;

break;

}

}

else if(ql.top().lj>=qr.top().lj){//如果放左邊的影響大過右邊

if(test(qr.top())){

s.push(qr.top());

qr.pop();

fangxiang=1;

}

else{

fx=-1;

break;

}

}

else if(ql.top().lj<qr.top().lj){ //如果放右邊的影響大過左邊

if(test(ql.top())){

s.push((ql.top()));

ql.pop();

fangxiang=-1;

}

else{

fx=-1;

break;

}

}

}

else {

if(fangxiang==1){

if(test(qr.top())&&!qr.empty()){

s.push(qr.top());

qr.pop();

fangxiang=1;

fx=1;

}

else{

if(ql.empty()){

fx=-1;break;}

fx--;

fangxiang=-1;

i--;

}

}

else {

if(test(ql.top())&&!ql.empty()){

s.push((ql.top()));

ql.pop();

fangxiang=-1;

fx=1;

}

else{

if(qr.empty()){

fx=-1;break;}

fx--;

fangxiang=1;

i--;

}

}

}

}

cout<<"Case "<<time<<":\n";

if(fx>-1){

while(!s.empty()){

things t=s.top();

s.pop();

cout<<t.p<<" "<<t.w<<endl;

}

}

else{

cout<<"Impossible\n";

}

}

return 0;

}