A

高中组合数学基础，要保证不越狱，后一个囚犯只受前一个影响，第一个不受影响，可选m种宗教，后一个不能与前一个相同，可选m-1种，答案即为m^n-m\*(m-1)^(n-1);

代码：

#include <math.h>

#include <stdio.h>

#include <string.h>

#include <iostream>

#include <algorithm>

using namespace std;

long long quickpow(long long m,long long n){

long long ans=1;

m%=100003;

while(n){

if(n&1) ans=ans\*m%100003;

m=m\*m%100003;

n>>=1;

}

return ans;

}

int main(){

long long m,n;

scanf("%lld%lld",&m,&n);

long long ans=quickpow(m,n);

ans=ans-m\*quickpow(m-1,n-1)%100003;

printf("%d",ans>0?ans:ans+100003);

}

B

两种操作，Q L 求末尾L个数中的最大数，A n，将n加在最近一次查询操作的t上，思路其实很清晰了，A操作记录当前的t+n，并更新末尾各个区间的最大值，Q操作更新t的值

代码：

#include <math.h>

#include <stdio.h>

#include <string.h>

#include <iostream>

#include <algorithm>

using namespace std;

int a[210000],maxn[210000];

int main(){

int m,d;

scanf("%d%d",&m,&d);

int t=0,len=0,n;

char sign[1];

while(m--){

scanf("%s%d",sign,&n);

if(sign[0]=='A'){

a[++len]=(t+n)%d;

for(int i=len;i;i--){

if(maxn[i]<a[len]) maxn[i]=a[len];

else break;

}//核心代码，更新末尾区间的最大值

}

else{

t=maxn[len+1-n];

printf("%d\n",t);

}

}

}

C

这题其实不管第二列情况如何，最多只有两种情况，因为第二列无雷，根据a[1] 和b[1]可以确定a[2]的情况，往后b[i],a[i-1],a[i-2]可以确定a[i]的情况，这其实最多只有两种情况，a[1]==0 or 1分别模拟一遍，再自己手写几组，推出关系式模拟即可

代码：

#include <math.h>

#include <stdio.h>

#include <string.h>

#include <iostream>

#include <algorithm>

using namespace std;

int a[10005],b[10005],ans=2;

int f(int n){

for(int i=2;i<=n;i++){

a[i]=b[i-1]-a[i-1]-a[i-2];

if(a[i]!=1&&a[i]!=0){ //只有a[i]==0 or 1合法

ans--;

break;

}

if(i==n){

if(b[i]!=a[i-1]+a[i]){

//末尾需要另一组判别式，a[i]=b[i]-a[i-1]

ans--;

break;

}

}

}

}

int main(){

int n;

scanf("%d",&n);

for(int i=1;i<=n;i++) scanf("%d",&b[i]);

a[1]=0;

f(n);

a[1]=1;

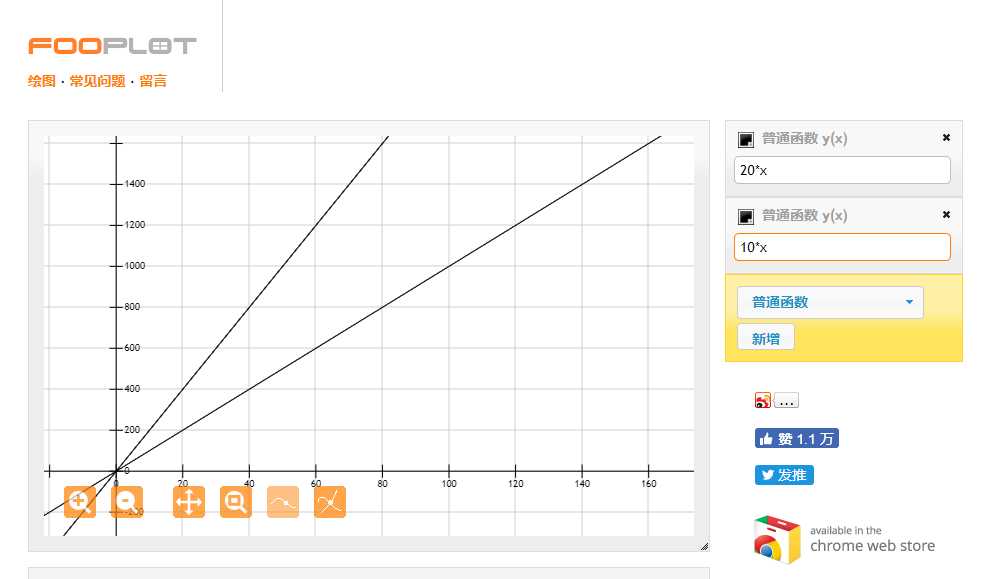
f(n);

printf("%d\n",ans);

}

E

这题问怎么买书最便宜，图书的总价关于图书的数量图像类似下图：



例如样例1：在100位置处，可以选择新的函数图像

总价y=p\*n，但是此时p是改变的，所以要求的其实是区间内最小的p，即图像的斜率较小点，此处有一个原则，买的书越多，优惠幅度越大，斜率呈现减小趋势，其实还是数学问题

代码：

#include <math.h>

#include <stdio.h>

#include <string.h>

#include <iostream>

#include <algorithm>

using namespace std;

typedef long long ll;

ll a[100005],p[100005],mini[100005];

int main(){

int t;

scanf("%d",&t);

while(t--){

int n,m;

scanf("%d%d",&n,&m);

for(int i=0;i<n;i++) scanf("%lld%lld",&a[i],&p[i]);

ll tmpMin=0x3f3f3f3f3f3f3f3fLL;

for(int i=n-1;i>=0;i--){

tmpMin=min(a[i]\*p[i],tmpMin);

mini[i]=tmpMin;

//先打出一份最小价格表

}

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

int ques;

scanf("%d",&ques);

if(ques>=a[n-1]) printf("%lld\n",ques\*p[n-1]);

//要买的数超过最大区间，直接按最后的价格购买

else{

int pos=upper\_bound(a,a+n,ques)-a;

//查找第一个大于ques的元素

ll ans=ques\*p[pos-1];

ans=min(ans,mini[pos]);

printf("%lld\n",ans);

}

}

}

}

F

在含7的时间醒来，每x分钟按一次按钮，直到到达给定时间，注意进位

代码：

#include <math.h>

#include <stdio.h>

#include <string.h>

#include <iostream>

#include <algorithm>

using namespace std;

int main(){

int x,hh,mm;

scanf("%d%d%d",&x,&hh,&mm);

int ans=0;

while(hh%10!=7&&mm%10!=7){

ans++;

mm-=x;

if(mm<0){

hh--;

mm+=60;

}

if(hh<0) hh=23;

}

printf("%d\n",ans);

}

G

将n分成k个数的和，这k个数是2^i形式，先转化成二进制，从高位依次降位，一个2^i可以变为两个2^(i-1)，接下来模拟降位操作即可，注意根据二进制降位只能增加位数，所以若二进制数里1的个数大于k，直接输出No，其余情况由于2^(-i)存在，都能满足条件

转换成二进制的方法

eg: 60%2=0，60/2=30%2=0，30/2=15%2=1，15/2=7%2=1，7/2=3%2=1，3/2=1%2=1，1/2=0，break；

则60的二进制码为 1 1 1 1 0 0

代码：

#include <map>

#include <math.h>

#include <stdio.h>

#include <string.h>

#include <iostream>

#include <algorithm>

using namespace std;

typedef long long ll;

map<int,int> bit;

int main(){

ll n,k;

scanf("%lld%lld",&n,&k);

for(int i=0;n;n/=2,i++){

bit[i]=n%2;

k-=bit[i];

} //转换二进制

if(k<0) printf("No\n");

//判断是否可行

else{

for(int i=63;;i--){

if(bit[i]>k) break;

k-=bit[i];

bit[i-1]+=bit[i]\*2;

bit[i]=0;

}

//降位操作

int cnt=-64;

while(!bit[cnt]) cnt++;

//记录边界

bit[cnt]--;

printf("Yes\n");

for(int i=63;i>=cnt;i--)

while(bit[i]--) printf("%d ",i);

//根据bit数组的值输出

for(int i=1;i<=k;i++) printf("%d ",cnt-i);

//顺序补位

printf("%d\n",cnt-k);

}

}