A

按题目要求模拟一遍即可

代码：

#include <cstdio>

#include <cstring>

#include <iostream>

#include <algorithm>

using namespace std;

int a[50][50];

int main(){

int n;

scanf("%d",&n);

int cnt=1,x,y;

while(cnt<=n\*n){

if(cnt==1) x=1,y=(n+1)/2,a[x][y]=cnt++;

else if(x!=1&&y==n) y=1,a[--x][y]=cnt++;

else if(x==1&&y!=n) x=n,a[x][++y]=cnt++;

else if(x==1&&y==n) a[++x][y]=cnt++;

else if(x!=1&&y!=n){

if(a[x-1][y+1]==0) a[--x][++y]=cnt++;

else a[++x][y]=cnt++;

}

}

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

for(int j=1;j<=n;j++) printf("%d ",a[i][j]);

printf("\n");

}

}

B

看题意，出现环则停止，暴力把每一个点跑一遍，下一个点的出度则为上一个点的出度+1，（其实我也不太清楚出度的具体概念，在这里我只是把他作为一个计数工具，说法有错也请原谅）则，用成环点的终点出度减去起点出度，其实这两点是同一点，但是上一点的出度即为下一点的入度，这里请用心体会。。我也讲不太清楚

代码：

#include <cstdio>

#include <cstring>

#include <iostream>

#include <algorithm>

using namespace std;

int a[200005],b[200005],cnt[200005];

int main(){

int n;

scanf("%d",&n);

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

int ans=n;

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

if(b[i]) continue;

b[i]=i;

cnt[i]=1;

int pos=i; //记录起始位置

while(b[a[pos]]==0){ 把每一个点跑一边

b[a[pos]]=i;

cnt[a[pos]]=cnt[pos]+1;

pos=a[pos];

}

if(b[a[pos]]==i) ans=min(cnt[pos]-cnt[a[pos]]+1,ans);

//若终点位置等于起点位置则成环

}

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

}

D

二分搜索即可，二分搜索可能的答案，再判断若跳这个距离需要取走的石头数量，若大于给定值，则需要减少距离，反之增加距离

代码：

#include <cstdio>

#include <cstring>

#include <iostream>

#include <algorithm>

using namespace std;

int a[50005],L,n,m;

int judge(int d){

int pos=0,cnt=0;

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

if(a[i]-pos<d) cnt++;

else pos=a[i];

}

if(L-pos<d) cnt++;

return cnt;

}

int main(){

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

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

int l=1,r=L+1,mid=(l+r)>>1;

while(l<r-1){

if(judge(mid)<=m)l=mid;

else r=mid;

mid=(l+r)>>1;

}

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

}

E

E题我想到用dp解决，样例过了很高兴，但是mle，做题的时候没忍住百度了，但是不好意思交。。。dp[i][j][k][0/1],代表a字符串的i位置，与b字符串的j位置分成k段的情况，但是此处有坑，第四维表示的是i位置是否匹配上的情况

dp[i][j][k]=dp[i-1][j-1][k]+dp[i-1][j-1][k-1]分别表示i是不是建立了一个新的子串，但这样的状态是默认了i用了 显然i可以不用 也就是说这样就遗漏了许多状态

则加一维表示i位置是否使用的情况：

if(a[i]==b[j]){

dp[flag][j][l][0]=(dp[!flag][j][l][0]+dp[!flag][j][l][1])%mod;

dp[flag][j][l][1]=(dp[!flag][j-1][l-1][0]+

dp[!flag][j-1][l-1][1]+dp[!flag][j-1][l][1])%mod;

}

else{

dp[flag][j][l][0]=(dp[!flag][j][l][0]+dp[!flag][j][l][1])%mod;

dp[flag][j][l][1]=0;

}

答案则为dp[n][m][k][0]+dp[n][m][k][1];

然而接下来的坑点难住了我，dp[1005][205][205][2]是会mle的

我们发现dp[i][j][k][sign]只与其上一位有关，所以只用到两个状态，

可以引入滚动数组，具体看代码：

#include <cstdio>

#include <cstring>

#include <iostream>

#include <algorithm>

using namespace std;

const int mod=1e9+7;

int dp[2][205][205][2];

char a[1005],b[205];

int main(){

int n,m,k,flag=1;

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

scanf("%s",a+1);

getchar();

scanf("%s",b+1);

//printf("%s %s\n",a+1,b+1);

for(int i=0;i<=1;i++) dp[i][0][0][0]=1;

for(int i=1;i<=n;i++,flag=!flag)

for(int j=1;j<=m;j++)

for(int l=1;l<=k;l++){

if(a[i]==b[j]){

dp[flag][j][l][0]=(dp[!flag][j][l][0]+dp[!flag][j][l][1])%mod;

dp[flag][j][l][1]=(dp[!flag][j-1][l-1][0]+(dp[!flag][j-1][l-1][1]+dp[!flag][j-1][l][1])%mod)%mod;

}

else{

dp[flag][j][l][0]=(dp[!flag][j][l][0]+dp[!flag][j][l][1])%mod;

dp[flag][j][l][1]=0;

}

}

printf("%d\n",(dp[n%2][m][k][0]+dp[n%2][m][k][1])%mod);

}

G

1^2+2^2+3^2+…=?

唯一的限定条件是个数不能大于k，模拟即可

代码：

#include <cstdio>

#include <cstring>

#include <iostream>

#include <algorithm>

using namespace std;

int main(){

long long k,sum=0;

scanf("%lld",&k);

int n=1,i=1;

while(i+n<=k) sum+=n\*n,i+=n,n++;

sum+=n\*(k-i)+n;

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

}

H

遇见雷则更新周围的位置情况，唯一的坑是会影响周围的雷，再开一个数组记录即可

代码：

#include <cstdio>

#include <cstring>

#include <iostream>

#include <algorithm>

using namespace std;

int map[105][105],a[105][105];

int main(){

int n,m;

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

//getchar();

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

getchar();

for(int j=1;j<=m;j++) scanf("%c",&map[i][j]);

}

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

for(int j=1;j<=m;j++)

if(map[i][j]=='\*'){

a[i+1][j]++;

a[i+1][j+1]++;

a[i+1][j-1]++;

a[i][j+1]++;

a[i][j-1]++;

a[i-1][j]++;

a[i-1][j+1]++;

a[i-1][j-1]++;

}

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

for(int j=1;j<=m;j++){

if(map[i][j]=='\*') printf("\*");

else printf("%d",a[i][j]);

}

printf("\n");

}

}