# Problem A. 中位数

## 题目描述

给定个长度为N的非降数列，求数列的中位数，

中位数:当数列的项数N为奇数时，处f中间位置的变量值即为中位数:当N为偶数时，中位数则为处于中间位置的两个数的平均数。

## 输入格式

输入数据第一行是一个整数T(1 <T < 100)，表示测试数据的组数。对三每组测试数据:

第一行是一个正整数N(1<N< 100)，表示数列长度。

第二行有N个整数，数之间用空格限开，所有的整数都不超过105,表示这个数列。输出格式

对于每组测试数据，输出数列的中位数。请不要输出小数点末尾多余的0。

## 输入样例

2  
4  
1 1 2 2  
5  
1 1 2 2 3

## 输出样例

1.5  
2

## AC代码

#include <bits/stdc++.h>  
#define FF(a,b) for(int a=0;a<b;a++)  
#define F(a,b) for(int a=1;a<=b;a++)  
#define LEN 200  
#define INF 100000  
#define bug(x) cout<<#x<<"="<<x<<endl;  
  
using namespace std;  
typedef long long ll;  
const double pi=acos(-1);  
  
int main()  
{  
// freopen("./in","r",stdin);  
 int T,N;  
 scanf("%d",&T);  
  
 while(T--){  
 scanf("%d",&N);  
 F(i,N) scanf("%d",&nums[i]);  
 sort(nums,nums+N);  
 if(N%2){//odd  
 printf("%d\n",nums[N/2+1]);  
 }else{  
 int sum=(nums[N/2]+nums[N/2+1]);  
 if(sum%2)  
 printf("%.1f\n",sum/2.);  
 else  
 printf("%d\n",sum/2);  
 }  
 }  
 return 0;  
}

# Problem B.内存分配

## 题目描述

在操作系统中，内存分配是非常重要的工作。

已知内存空间由N个内存块组成，这些内存块从1到N编号。

进行内存分配时，操作系统将选择一块大小足够的内存全部分配给请求内存的进程。例如，当进程请求10MB的内存时，操作系统必须向该进程分配一个不小于10MB的内存块。**内存块不能重复分配**。

操作系统有三种基本的分配方式，分别为:

* 首次适应:从1号到N号内存块依次查找，自到找到第一 块足够大的月未分配出去的内存块，将其分配绘进程。
* **最佳适应**:找到当前未分配出去且大小足够的内存块中最小的内存块分配给进程。
* 最差适应:找到当前未分配出去且大小足够的内有块中最小的内有块分配给进程。

其中，最佳适应方式是应用最为广泛。现在，操作系统要依次交理M个进程的内存清求，请按照**最佳适应**方式分配内存，并输出相应的内存块的大小。如果，没有大小足够的内存块可以满足兰前清求，则输出“NULL”(不包含引号)，并跳过该诗求

## 输入格式

输入数据的第一行是测试数据组数T(T ≤ 20)每组数据由4行构成:

1. 第一行为一个整数N(1≤N≤100);表示有N个内存块
2. 第二行有N个整数，第i个整数表示第i块内存块的大小
3. 第三行为一个整数M(1≤M≤100)，表示有M个请求
4. 第四行有M个整数，表示进程所请求的内存空间。

## 输出格式

每组数据输出一行，每行有M个数，表示操作系统采月**最住适应**方式，依次分配给进星的内存块大小;

如果没有可用内存块，输出“NULL”(不包含引号)

**不要输山多余的行尾空格**。

## 输入样例

2  
4  
7 5 10 3  
2  
4 6  
4  
3 5 9 10  
3  
5 12 6

## 输出样例

5 7  
5 NULL 9

## AC代码

#include <bits/stdc++.h>  
#define FF(a,b) for(int a=0;a<b;a++)  
#define F(a,b) for(int a=1;a<=b;a++)  
#define LEN 200  
#define INF 100000  
#define bug(x) cout<<#x<<"="<<x<<endl;  
  
using namespace std;  
typedef long long ll;  
const double pi=acos(-1);  
  
int nums[LEN];  
  
int main()  
{  
// freopen("./in","r",stdin);  
 int T,N;  
 scanf("%d",&T);  
 int t,M;  
 while(T--){  
 vector<int> v;  
 scanf("%d",&N);  
 F(i,N){  
 scanf("%d",&t);  
 v.push\_back(t);  
 }  
 sort(v.begin(),v.end());  
 scanf("%d",&M);  
 FF(i,M){  
 scanf("%d",&t);  
 vector<int>::iterator it=lower\_bound(v.begin(),v.end(),t);  
 if(it==v.end())  
 printf("NULL");  
 else{  
 printf("%d",\*it);  
 v.erase(it);  
 }  
 if(i!=M-1) printf(" ");  
 }  
 puts("");  
 }  
 return 0;  
}

# Problem C.图像识别

## 题目描述

在图像识别中，我们经常需要分析特定图像中的些特征，而其中很重要的点就是识别出图像的多个区域。在这个问题中，我们将给定一幅N x M的图像，其中每个1 x 1的点都用一个[0, 255]的值来表示他的RGB颜色。如果两个相邻的像素点颜色差值不超过D.我们就认为这两个像素点属于豆一个区域。对于一个像素点(x,y),以下这8个点(如果存在)是与它相邻的: (x 一1,y一1),(x 一1,y),(x一1,y + 1),(x,y 一1),(x,y + 1),(x + 1,y- 1),(x + 1,y),(x +1,y+ 1)。

你的任务是写一个程序，分辨出给定图像中一共被分为多少个区域。

## 输入格式

输入数据包含多组测试数据。

输入的第一行是一个整数T （T ≤ 100），表示测试效据的组数,

每组测试数据的第一行是三个整数N,M,D(1≤N,M≤100, 0≤D≤255)，意义知上所述。

接下来N行，每行M个整数，表示给定图像的每人像素点颜色。

## 输出格式

对于每组测试数据输出一行，即图像中的区域数量

## 输入样例

2  
3 3 0  
1 1 1  
0 1 0  
0 1 0  
3 4 1  
10 11 12 13  
9 8 7 6  
2 3 4 5

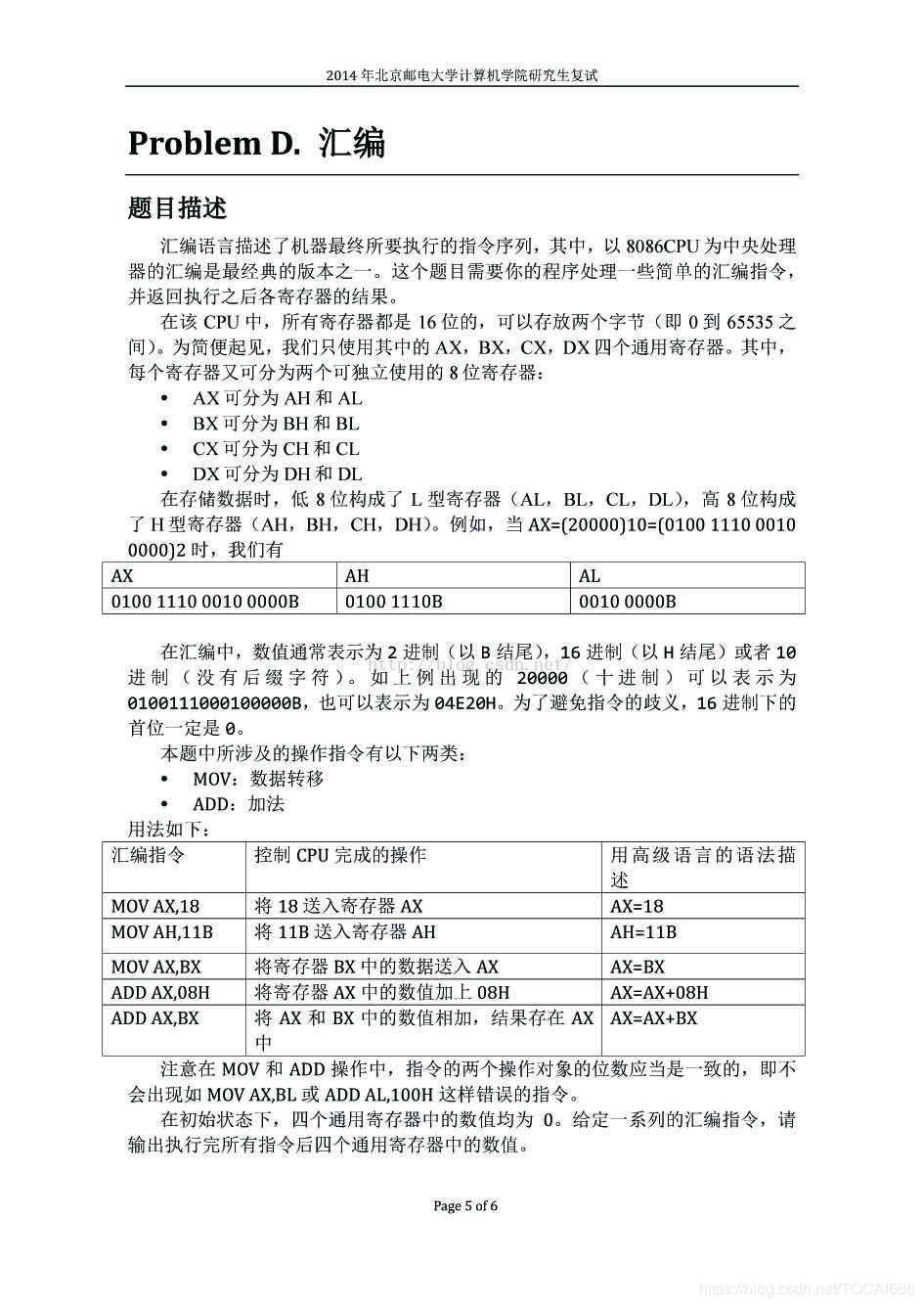
## 输出样例

3  
1

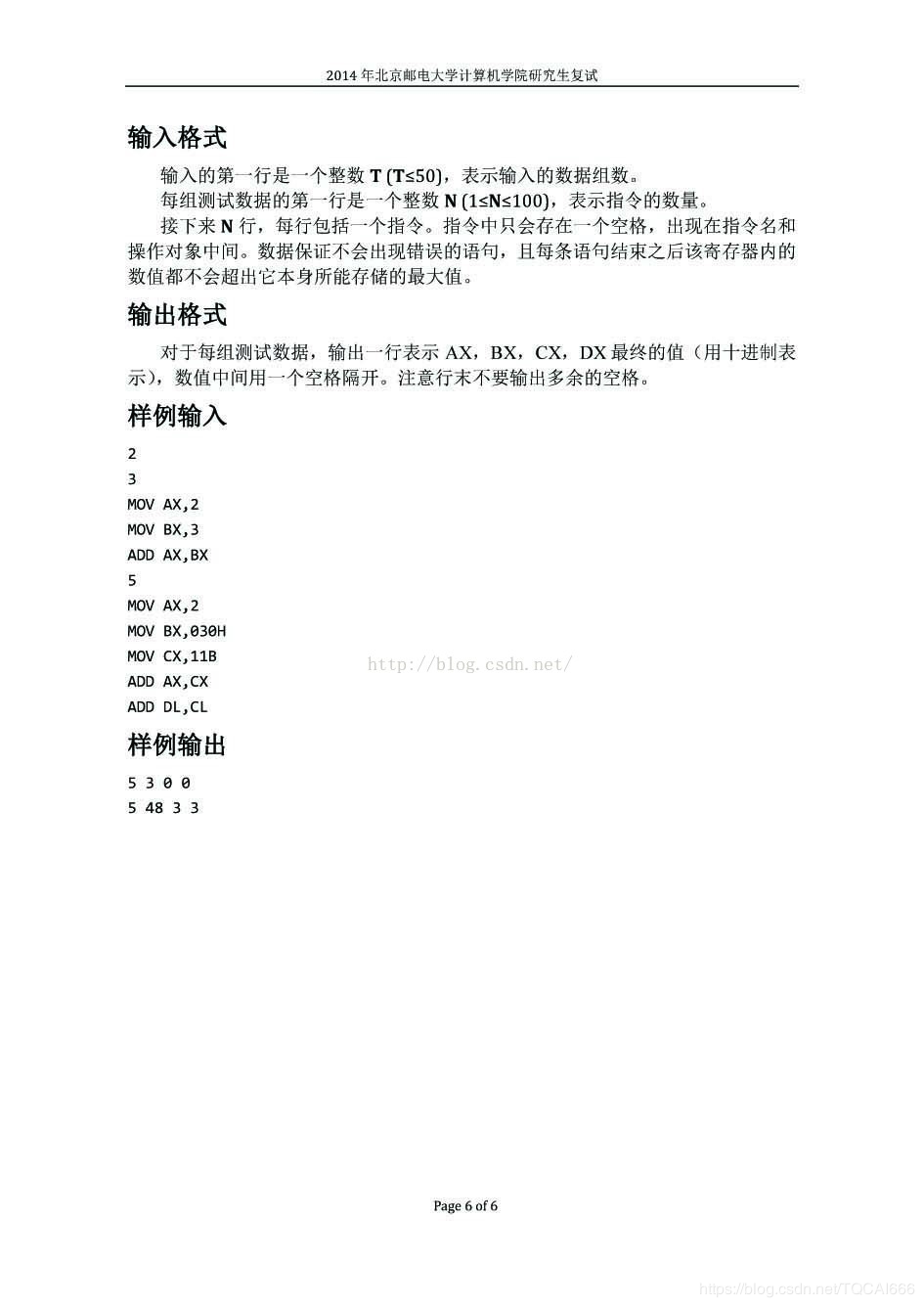
## AC代码

#include <bits/stdc++.h>  
#define FF(a,b) for(int a=0;a<b;a++)  
#define F(a,b) for(int a=1;a<=b;a++)  
#define LEN 200  
#define INF 100000  
#define bug(x) cout<<#x<<"="<<x<<endl;  
  
using namespace std;  
typedef long long ll;  
const double pi=acos(-1);  
  
int g[100][100];  
int vis[100][100];  
  
typedef struct Node{  
 int x,y;  
 Node(int x,int y):x(x),y(y){  
 }  
  
  
}Node;  
  
int T,N,D,M;  
  
bool valid(int i,int j){  
 if(i>=0 && i<N && j>=0 && j<M)  
 return 1;  
 return 0;  
}  
  
int main()  
{  
// freopen("./in","r",stdin);  
 scanf("%d",&T);  
 while(T--){  
 scanf("%d%d%d",&N,&M,&D);  
 memset(vis,0,sizeof vis);  
 FF(i,N) FF(j,M) scanf("%d",&g[i][j]);  
 queue<Node> q;  
 int cnt=0;  
 FF(x,N) FF(y,M) if(vis[x][y]==0){  
 cnt++;  
 q.push(Node(x,y));  
 vis[x][y]=1;  
 while(!q.empty()){  
 Node u=q.front();  
 q.pop();  
 int cx=u.x;int cy=u.y;  
 int v=g[cx][cy];  
 for(int i=cx-1;i<=cx+1;i++)  
 for(int j=cy-1;j<=cy+1;j++)  
 if(!(cx==i && cy==j) && valid(i,j) && vis[i][j]==0){  
 if(abs(v-g[i][j])<=D){  
 vis[i][j]=1;  
 q.push(Node(i,j));  
 }  
 }  
 }  
 }  
 printf("%d\n",cnt);  
 }  
 return 0;  
}

# Problem D. 汇编



在这里插入图片描述



在这里插入图片描述

## 输入数据

2  
3  
MOV AX,2  
MOV BX,3  
ADD AX,BX  
5  
MOV AX,2  
MOV BX,030H  
MOV CX,11B  
ADD AX,CX  
ADD DL,CL

## 输出数据

5 3 0 0  
5 48 3 3

## RE代码

能过样例， 但是RE

#include <bits/stdc++.h>  
#define FF(a,b) for(int a=0;a<b;a++)  
#define F(a,b) for(int a=1;a<=b;a++)  
#define LEN 200  
#define INF 100000  
#define bug(x) cout<<#x<<"="<<x<<endl;  
  
using namespace std;  
typedef long long ll;  
const double pi=acos(-1);  
  
unsigned short R[4];//四个寄存器， 高位H 低位L 整体X  
  
int calcByRadix(const char\* ch,int r){  
 int n=strlen(ch);  
 int ans=0;  
 int base=1;  
 for(int j=n-1; j>=0 ;j--){  
 int v;  
 if(ch[j]>='0'&&ch[j]<='9')  
 v=ch[j]-'0';  
 else  
 v=ch[j]-'A'+10;  
 ans+=v\*base;  
// bug(ans)  
 base\*=r;  
 }  
 return ans;  
}  
  
int getDigitValue(const char\* ch){  
 int n=strlen(ch);  
 int truncate;  
 int radix;  
 switch(ch[n-1]){  
 case 'H':  
 radix=16;  
 truncate=n-1;  
 break;  
 case 'B':  
 radix=2;  
 truncate=n-1;  
 break;  
 default:  
 radix=10;  
 truncate=n;  
 }  
 string obj=string(ch).substr(0,truncate);  
 return calcByRadix(obj.c\_str(),radix);  
}  
  
int getValue(const char\* ch){  
 int tmp;  
 if(ch[0]>='0' && ch[0]<='9'){  
 return getDigitValue(ch);  
 }else{  
 int id=ch[0]-'A';  
 switch(ch[1]){  
 case 'X':  
 return R[id];  
 break;  
 case 'L':  
 tmp=R[id]&0b11111111;  
 return tmp;  
 case 'H':  
 tmp= R[id]&0b1111111100000000;  
 tmp>>=8;  
 return tmp;  
 break;  
 }  
 }  
}  
  
int assignTo(int id,int value,char pattern){  
 switch(pattern){  
 case 'X':  
 R[id]=value;  
 break;  
 case 'L':  
 R[id]=~R[id];  
 R[id]|=0b11111111;  
 R[id]=~R[id];  
 R[id]+=value;  
 break;  
 case 'H':  
 R[id]=~R[id];  
 R[id]|=0b1111111100000000;  
 R[id]=~R[id];  
 value<<=8;  
 R[id]+=value;  
 break;  
 }  
}  
  
int main()  
{  
// freopen("./in","r",stdin);  
 int N,T;  
 scanf("%d",&T);  
 char op[10],fuck[10];  
 while(T--){  
 scanf("%d",&N);  
 while(N--){  
 //scanf("%s ",op);  
 scanf("%s %s",op,fuck);  
// bug(fuck)  
 string S[2];  
 int sp=0;  
 FF(i,strlen(fuck)){  
 if(fuck[i]==','){  
 sp++;  
 }else  
 S[sp]+=fuck[i];  
 }  
 const char \*A=S[0].c\_str();  
 const char \*B=S[1].c\_str();  
 if(op[0]=='M'){  
 int value=getValue(B);  
 int id=A[0]-'A';  
 assignTo(id,value,A[1]);  
 }else{  
 int v1=getValue(A);  
 int v2=getValue(B);  
 int value=v1+v2;  
 int id=A[0]-'A';  
 assignTo(id,value,A[1]);  
 }  
 }  
 FF(i,4){  
 printf("%u",R[i]);  
 if(i!=3)  
 printf(" ");  
 }  
 puts("");  
 }  
 return 0;  
}