# 文件对拍的操作

freopen**(**"in.txt"**,** "r"**,** stdin**);**

fclose**(**stdin**);**

freopen**(**"CON"**,** "r"**,** stdin**);**

int m**;**

cin **>>** m**;**

cout **<<** m**;**

/\*

windows 下对拍

:again

D:\cb-work4\gen\bin\Debug\gen.exe

D:\cb-work4\duiA\bin\Debug\duiA.exe

D:\cb-work4\duiB\bin\Debug\duiB.exe

fc C:\Users\admin\Desktop\duipai\out1.txt C:\Users\admin\Desktop\duipai\out2.txt

if not errorlevel 1 goto again

pause

文件后缀名.bat

linux 下对拍

if diff test.out test.ans;then

echo AC

else

echo WA

exit 0

fi

Done

保存为\*.sh  然后 sh \*.sh就行了

\*/

# RMQ线段树

#include<iostream>

#include<cstdio>

#include<cstring>

#include<limits.h>

**using** **namespace** std**;**

const int MAXN **=** 1 **<<** 18**;** //262144

int n**,** dat**[**2 **\*** MAXN **-** 1**];**

int init**(**int n\_**)** **{**

n **=** 1**;**

**while(**n **<** n\_**)** n **\*=** 2**;**

**for(**int i **=** 0**;** i **<** 2 **\*** n **-** 1**;** i**++)** dat**[**i**]** **=** **-**1**;**

**return** n**;**

**}**

void update**(**int k**,** int a**)** **{**

k **+=** n **-** 1**;**

dat**[**k**]** **=** a**;**

**while(**k **>** 0**)** **{**

k **=** **(**k **-** 1**)** **/** 2**;**

dat**[**k**]** **=** max**(**dat**[**2 **\*** k **+** 1**],** dat**[**2 **\*** k **+** 2**]);**

**}**

**}**

int dfs**(**int k**)** **{** //保证每个点只算一遍

**if** **(**dat**[**k**]** **!=** **-**1**)** **return** dat**[**k**];**

**if** **(**k **<** n **-** 1**)** **{**

dat**[**k**]** **=** max**(**dfs**(**2 **\*** k **+** 1**),** dfs**(**2 **\*** k **+** 2**));**

**return** dat**[**k**];**

**}**

**return** **-**1**;**

**}**

int query**(**int a**,** int b**,** int k**,** int l**,** int r**)** **{** //[a,b) 节点编号 当前查询范围

**if** **(**r **<=** a **||** l **>=** b**)** **{**

**return** **-**1**;**

**}**

**if** **(**a **<=** l **&&** r **<=** b**)** **{**

**return** dat**[**k**];**

**}** **else** **{**

int vl **=** query**(**a**,** b**,** k **\*** 2 **+** 1**,** l**,** **(**l **+** r**)** **/** 2**);**

int vr **=** query**(**a**,** b**,** k **\*** 2 **+** 2**,** **(**l **+** r**)** **/** 2**,** r**);**

**return** max**(**vl**,** vr**);**

**}**

**}**

void solve**()**

**{**

int N**,** Q**;**

**while(**cin **>>** N **>>** Q**)** **{**

int n **=** init**(**N**);** // n必须取整个线段树，否则编号计算会出错

**for(**int i **=** 0**;** i **<** N**;** i**++)** **{**

scanf**(**"%d"**,** dat **+** n **-** 1 **+** i**);** // 先读入数据

**}**

dfs**(**0**);**

**for(**int i **=** 0**;** i **<** Q**;** i**++)** **{**

char s**[**5**];**

int a**,** b**;**

scanf**(**"%s %d%d"**,** s**,** **&**a**,** **&**b**);**

**if** **(**strcmp**(**s**,** "Q"**)** **==** 0**)** **{**

printf**(**"%d\n"**,** query**(**a**-**1**,** b**,** 0**,** 0**,** n**));**

**}** **else** **if** **(**strcmp**(**s**,** "U"**)** **==** 0**)** **{**

update**(**a**-**1**,** b**);**

**}**

**}**

**}**

**}**

int main**()** **{**

solve**();**

**}**

# TSP旅行商问题（状压DP）

#include <iostream>

#include <cstdio>

#include <cstring>

**using** **namespace** std**;**

const int MAXN **=** 16**;**

const int INF **=** 0x3f3f3f3f**;**

int n**;**

int dp**[**1 **<<** MAXN**][**MAXN**];**

int d**[**MAXN**][**MAXN**];**

int rec**(**int s**,** int v**)** **{**

//记忆化

**if** **(**dp**[**s**][**v**]** **>=** 0**)** **{**

**return** dp**[**s**][**v**];**

**}**

//已经访问过所有节点并返回零号节点

**if** **(**s **==** **(**1 **<<** n**)** **-** 1 **&&** v **==** 0**)** **{**

**return** dp**[**s**][**v**]** **=** 0**;**

**}**

int res **=** INF**;**

**for** **(**int u **=** 0**;** u **<** n**;** u**++)** **{**

**if** **(!(**s **>>** u **&** 1**))** **{**

//下一步移动到顶点U

res **=** min**(**res**,** rec**(**s **|** 1 **<<** u**,** u**)** **+** d**[**v**][**u**]);**

**}**

**}**

**return** dp**[**s**][**v**]** **=** res**;**

**}**

int solve**()** **{**

memset**(**dp**,** **-**1**,** **sizeof(**dp**));**

printf**(**"%d\n"**,** rec**(**0**,** 0**));**

**}**

void floyd**()** **{**

**for(**int k **=** 0**;** k **<** n**;** k**++)** **{**

**for(**int i **=** 0**;** i **<** n**;** i**++)** **{**

**for(**int j **=** 0**;** j **<** n**;** j**++)** **{**

**if** **(**i **==** j**)** d**[**i**][**j**]** **=** 0**;**

**else** d**[**i**][**j**]** **=** min**(**d**[**i**][**j**],** d**[**i**][**k**]** **+** d**[**k**][**j**]);**

**}**

**}**

**}**

**}**

int main**()**

**{**

int T**;**

cin **>>** T**;**

**while(**T**--)** **{**

memset**(**d**,** INF**,** **sizeof(**d**));**

int m**;**

cin **>>** n **>>** m**;**

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

int x**,** y**,** v**;**

scanf**(**"%d%d%d"**,** **&**x**,** **&**y**,** **&**v**);**

**if** **(**d**[**x **-** 1**][**y **-** 1**]** **>** v**)** d**[**x **-** 1**][**y **-** 1**]** **=** d**[**y **-** 1**][**x **-** 1**]** **=** v**;**

**}**

//每个点只能走一次就是原本的TSP问题，可以走多次先求各点最短路转化为TSP

//道理比如1 - 3, 1 - 2 那么 1 要走3次， TSP求不出 但是求一波Floyd以后， 3 - 2有条路了，相当于改变了图

floyd**();**

solve**();**

**}**

**return** 0**;**

**}**

**大数对INT取模**

#include<iostream>

#include<cstdio>

#include<string>

#include<cstring>

#include<iomanip>

#include<algorithm>

**using** **namespace** std**;**

char str**[**10000001**];**

int len**;**

int mod**(**const int **&** b**)** //大数对一个int类型的变量进行取模运算

**{**

int i**,**d**=**0**;**

**for** **(**i **=** 0**;** i **<** len**;** i**++)**

**{**

d **=** **((**d **\*** 10**)%** b **+** str**[**i**]** **-** '0'**)%** b**;**

**}**

**return** d**;**

**}**

int main**(**void**)**

**{**

int k **=** 1**;**

**while(**scanf**(**"%s"**,**str**)** **!=** EOF**)**

**{**

len **=** strlen**(**str**);**

printf**(**"Case #%d: "**,** k**++);**

int a **=** 73**,** b **=** 137**;**

//cout << mod(a) << endl;

//cout << mod(b) << endl;

**if** **((**mod**(**a**)** **==** 0**)** **&&** **(**mod**(**b**)** **==** 0**))** printf**(**"YES\n"**);**

**else** printf**(**"NO\n"**);**

**}**

**return** 0**;**

**}**

# 高精度版

#include <iostream>

#include <cstdio>

#include <cstdlib>

#include <cstring>

#include <string>

#include <algorithm>

**using** **namespace** std**;**

const int MAXN **=** 600**;**

char s1**[**MAXN**],** s2**[**MAXN**];**

struct bign

**{**

int len**,** s**[**MAXN**];**

bign **()**

**{**

memset**(**s**,** 0**,** **sizeof(**s**));**

len **=** 1**;**

**}**

bign **(**int num**)** **{** **\*this** **=** num**;** **}**

bign **(**const char **\***num**)** **{** **\*this** **=** num**;** **}**

bign **operator** **=** **(**const int num**)**

**{**

char s**[**MAXN**];**

sprintf**(**s**,** "%d"**,** num**);**

**\*this** **=** s**;**

**return** **\*this;**

**}**

bign **operator** **=** **(**const char **\***num**)**

**{**

**for(**int i **=** 0**;** num**[**i**]** **==** '0'**;** num**++)** **;** //ȥǰռ0

len **=** strlen**(**num**);**

**for(**int i **=** 0**;** i **<** len**;** i**++)** s**[**i**]** **=** num**[**len**-**i**-**1**]** **-** '0'**;**

**return** **\*this;**

**}**

bign **operator** **+** **(**const bign **&**b**)** const //+

**{**

bign c**;**

c**.**len **=** 0**;**

**for(**int i **=** 0**,** g **=** 0**;** g **||** i **<** max**(**len**,** b**.**len**);** i**++)**

**{**

int x **=** g**;**

**if(**i **<** len**)** x **+=** s**[**i**];**

**if(**i **<** b**.**len**)** x **+=** b**.**s**[**i**];**

c**.**s**[**c**.**len**++]** **=** x **%** 10**;**

g **=** x **/** 10**;**

**}**

**return** c**;**

**}**

bign **operator** **+=** **(**const bign **&**b**)**

**{**

**\*this** **=** **\*this** **+** b**;**

**return** **\*this;**

**}**

void clean**()**

**{**

**while(**len **>** 1 **&&** **!**s**[**len**-**1**])** len**--;**

**}**

bign **operator** **\*** **(**const bign **&**b**)** //\*

**{**

bign c**;**

c**.**len **=** len **+** b**.**len**;**

**for(**int i **=** 0**;** i **<** len**;** i**++)**

**{**

**for(**int j **=** 0**;** j **<** b**.**len**;** j**++)**

**{**

c**.**s**[**i**+**j**]** **+=** s**[**i**]** **\*** b**.**s**[**j**];**

**}**

**}**

**for(**int i **=** 0**;** i **<** c**.**len**;** i**++)**

**{**

c**.**s**[**i**+**1**]** **+=** c**.**s**[**i**]/**10**;**

c**.**s**[**i**]** **%=** 10**;**

**}**

c**.**clean**();**

**return** c**;**

**}**

bign **operator** **\*=** **(**const bign **&**b**)**

**{**

**\*this** **=** **\*this** **\*** b**;**

**return** **\*this;**

**}**

bign **operator** **-** **(**const bign **&**b**)**

**{**

bign c**;**

c**.**len **=** 0**;**

**for(**int i **=** 0**,** g **=** 0**;** i **<** len**;** i**++)**

**{**

int x **=** s**[**i**]** **-** g**;**

**if(**i **<** b**.**len**)** x **-=** b**.**s**[**i**];**

**if(**x **>=** 0**)** g **=** 0**;**

**else**

**{**

g **=** 1**;**

x **+=** 10**;**

**}**

c**.**s**[**c**.**len**++]** **=** x**;**

**}**

c**.**clean**();**

**return** c**;**

**}**

bign **operator** **-=** **(**const bign **&**b**)**

**{**

**\*this** **=** **\*this** **-** b**;**

**return** **\*this;**

**}**

bign **operator** **/** **(**const bign **&**b**)**

**{**

bign c**,** f **=** 0**;**

**for(**int i **=** len**-**1**;** i **>=** 0**;** i**--)**

**{**

f **=** f**\***10**;**

f**.**s**[**0**]** **=** s**[**i**];**

**while(**f **>** b **||** f **==** b**)**

**{**

f **-=** b**;**

c**.**s**[**i**]++;**

**}**

**}**

c**.**len **=** len**;**

c**.**clean**();**

**return** c**;**

**}**

bign **operator** **/=** **(**const bign **&**b**)**

**{**

**\*this** **=** **\*this** **/** b**;**

**return** **\*this;**

**}**

bign **operator** **%** **(**const bign **&**b**)**

**{**

bign r **=** **\*this** **/** b**;**

r **=** **\*this** **-** r**\***b**;**

**return** r**;**

**}**

bign **operator** **%=** **(**const bign **&**b**)**

**{**

**\*this** **=** **\*this** **%** b**;**

**return** **\*this;**

**}**

bool **operator** **<** **(**const bign **&**b**)**

**{**

**if(**len **!=** b**.**len**)** **return** len **<** b**.**len**;**

**for(**int i **=** len**-**1**;** i **>=** 0**;** i**--)**

**{**

**if(**s**[**i**]** **!=** b**.**s**[**i**])** **return** s**[**i**]** **<** b**.**s**[**i**];**

**}**

**return** **false;**

**}**

bool **operator** **>** **(**const bign **&**b**)**

**{**

**if(**len **!=** b**.**len**)** **return** len **>** b**.**len**;**

**for(**int i **=** len**-**1**;** i **>=** 0**;** i**--)**

**{**

**if(**s**[**i**]** **!=** b**.**s**[**i**])** **return** s**[**i**]** **>** b**.**s**[**i**];**

**}**

**return** **false;**

**}**

bool **operator** **==** **(**const bign **&**b**)**

**{**

**return** **!(\*this** **>** b**)** **&&** **!(\*this** **<** b**);**

**}**

string str**()** const

**{**

string res **=** ""**;**

**for(**int i **=** 0**;** i **<** len**;** i**++)** res **=** char**(**s**[**i**]+**'0'**)** **+** res**;**

**return** res**;**

**}**

**};**

int main**()**

**{**

bign a**,** b**,** c**;**

**while(**scanf**(**"%s %s"**,** s1**,** s2**)** **!=** EOF**)**

**{**

a **=** bign**(**s1**);**

b **=** bign**(**s2**);**

c **=** a **/** b**;**

cout **<<** c**.**str**()** **<<** endl**;**

**}**

**return** 0**;**

**}**

# 矩阵快速幂：

#include <iostream>

#include <cstdio>

#include <cstring>

#include <vector>

**using** **namespace** std**;**

//用二维vector表示矩阵

**typedef** vector**<**int**>** vec**;**

**typedef** vector**<**vec**>** mat**;**

**typedef** long long LL**;**

const int N **=** 2**;**

const int MOD **=** 10000**;**

int m**;**

mat mul **(**mat**&** A**,** mat**&** B**)** **{**

mat C**(**A**.**size**(),** vec**(**B**[**0**].**size**()));**

**for(**int k **=** 0**;** k **<** B**.**size**();** **++**k**)** **{**

**for(**int i **=** 0**;** i **<** A**.**size**();** **++**i**)** **{**

**for(**int j **=** 0**;** j **<** B**[**0**].**size**();** **++**j**)** **{**

C**[**i**][**j**]** **=** **(**C**[**i**][**j**]** **+** A**[**i**][**k**]** **\*** B**[**k**][**j**])** **%** m**;**

**}**

**}**

**}**

**return** C**;**

**}**

mat pow **(**mat**&** A**,** LL k**)** **{**

mat B**(**A**.**size**(),** vec**(**A**.**size**()));**

**for(**int i **=** 0**;** i **<** A**.**size**();** **++**i**)**

B**[**i**][**i**]** **=** 1**;** //初始化为单位矩阵

**while(**k **>** 0**)** **{**

**if** **(**k **&** 1**)** B **=** mul**(**B**,** A**);**

A **=** mul**(**A**,** A**);**

k **>>=** 1**;**

**}**

**return** B**;**

**}**

int main**()**

**{**

int k**,** a**[**10**];**

**while(**cin **>>** k **>>** m**)** **{**

**for(**int i **=** 0**;** i **<** 10**;** i**++)** **{**

scanf**(**"%d"**,** **&**a**[**i**]);**

**}**

mat T**(**10**,** vec**(**10**));**

**for(**int i **=** 0**;** i **<** 10**;** i**++)** **{**

**for(**int j **=** 0**;** j **<** 10**;** j**++)** **{**

**if** **(**i **==** 0**)** T**[**i**][**j**]** **=** a**[**j**];**

**else** T**[**i**][**j**]** **=** i **==** j **+** 1**;**

**}**

**}**

mat I**(**10**,** vec**(**1**));**

**for(**int i **=** 0**;** i **<** 10**;** i**++)** **{**

I**[**i**][**0**]** **=** 9 **-** i**;**

**}**

**if** **(**k **>=** 10**)** **{**

T **=** pow**(**T**,** k **-** 9**);**

I **=** mul**(**T**,** I**);**

printf**(**"%d\n"**,** I**[**0**][**0**]);**

**}** **else** **{**

printf**(**"%d\n"**,** k **%** m**);**

**}**

**}**

**return** 0**;**

**}**

# 矩阵快速幂二：

//矩阵快速幂 +++

#include <iostream>

#include <cstdio>

#include <cstring>

**using** **namespace** std**;**

**typedef** long long LL**;**

struct mat//矩阵结构体，a表示内容，size大小

**{**

LL a**[**2**][**2**],**size**;**

mat**()**

**{**

size**=**0**;**

memset**(**a**,**0**,sizeof(**a**));**

**}**

**};**

void prLL**(**mat m**)**//输出矩阵信息，debug用

**{**

LL i**,**j**;**

printf**(**"%lld\n"**,**m**.**size**);**

**for(**i**=**0**;**i**<**m**.**size**;**i**++)**

**{**

**for(**j**=**0**;**j**<**m**.**size**;**j**++)**printf**(**"%lld "**,**m**.**a**[**i**][**j**]);**

printf**(**"\n"**);**

**}**

**}**

LL mult**(**LL a**,**LL b**,**LL mod**)**//自己写大数乘法，利用位运算有效避免LL64或者高精度，这里a,b,mod只要不超过2^30就不会出问题

**{**

LL exp**=**a**%**mod**,**res**=**0**;**

**while(**b**)**

**{**

**if(**b**&**1**)**

**{**

res**+=**exp**;**

**if(**res**>=**mod**)**res**-=**mod**;**

**}**

exp**<<=**1**;**

**if(**exp**>=**mod**)**exp**-=**mod**;**

b**>>=**1**;**

**}**

**return** res**;**

**}**

mat mul**(**mat **&**m1**,**mat **&**m2**,**LL mod**)**//两个相等矩阵的乘法，对于稀疏矩阵，有0处不用运算的优化

**{**

mat ans**=**mat**();**

LL i**,**j**,**k**;**

**for(**i**=**0**;**i**<**m1**.**size**;**i**++)**

**for(**j**=**0**;**j**<**m2**.**size**;**j**++)**

**if(**m1**.**a**[**i**][**j**])**//稀疏矩阵优化

**for(**k**=**0**;**k**<**m1**.**size**;**k**++)**

**{**

ans**.**a**[**i**][**k**]** **+=** m1**.**a**[**i**][**j**]** **\*** m2**.**a**[**j**][**k**];**//mult(m1.a[i][j],m2.a[j][k],mod);

ans**.**a**[**i**][**k**]** **%=** mod**;**

**}**

ans**.**size**=**m1**.**size**;**

**return** ans**;**

**}**

mat pow**(**mat m**,**LL n**,**LL mod**)**//二分快速幂

**{**

mat ans**=**mat**();**

LL i**;**

**for(**i**=**0**;**i**<**m**.**size**;**i**++)**ans**.**a**[**i**][**i**]=**1**;**

ans**.**size**=**m**.**size**;**

**while(**n **>** 0**)**

**{**

**if(**n**&**1**)**ans **=** mul**(**m**,**ans**,**mod**);**

m **=** mul**(**m**,**m**,**mod**);**

n **>>=** 1**;**

//cout << n << endl;

**}**

**return** ans**;**

**}**

const LL MOD1 **=** 26880699 **-** 3**;**

const LL MOD2 **=** 20160519**;**

int main**()**

**{**

LL T**;**

cin **>>** T**;**

**while(**T**--)** **{**

LL n**;**

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

mat A**;**

A**.**size **=** 2**;**

A**.**a**[**0**][**0**]** **=** 1**;** A**.**a**[**0**][**1**]** **=** 1**;**

A**.**a**[**1**][**0**]** **=** 1**;** A**.**a**[**1**][**1**]** **=** 0**;**

A **=** pow**(**A**,** n**,** MOD1**);**

LL ans **=** A**.**a**[**1**][**0**];**

//cout << ans << endl;

A**.**a**[**0**][**0**]** **=** 1**;** A**.**a**[**0**][**1**]** **=** 1**;**

A**.**a**[**1**][**0**]** **=** 1**;** A**.**a**[**1**][**1**]** **=** 0**;**

A **=** pow**(**A**,** ans**,** MOD2**);**

ans **=** A**.**a**[**1**][**0**];**

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

**}**

**return** 0**;**

**}**

//563

//26880699

# 考新郎（组合数加错排公式）：

#include <iostream>

#include <cstdio>

**using** **namespace** std**;**

**typedef** long long LL**;**

const int MAXN **=** 25**;**

const int N **=** 25**;**

LL D**[**MAXN**];**

LL C**[**MAXN**][**MAXN**];**

void dp**()** **{**

D**[**1**]** **=** 0**;**

D**[**2**]** **=** 1**;**

**for(**int i **=** 3**;** i **<** 25**;** i**++)** **{**

D**[**i**]** **=** **(**i **-** 1**)** **\*** **(**D**[**i **-** 1**]** **+** D**[**i **-** 2**]);**

**}**

**}**

LL combination**()** **{**

C**[**1**][**0**]** **=** C**[**1**][**1**]** **=** 1**;**

**for** **(**int i **=** 2**;** i **<** N**;** i**++){**

C**[**i**][**0**]** **=** 1**;**

**for** **(**int j **=** 1**;** j **<** N**;** j**++)**

C**[**i**][**j**]** **=** **(**C**[**i **-** 1**][**j**]** **+** C**[**i **-** 1**][**j **-** 1**]);**

**}**

**}**

void solve**(**int N**,** int M**)** **{**

printf**(**"%I64d\n"**,** C**[**N**][**M**]** **\*** D**[**M**]);**

**}**

# 区间第K大的数——归并树实现

const int ST\_SIZE **=** **(**1 **<<** 18**)** **-** 1**;**

const int MAXN **=** 1E5 **+** 5**;**

const int MAXM **=** 5005**;**

int N**,** M**;**

int A**[**MAXN**];** // 要处理的数组

int I**[**MAXM**],** J**[**MAXM**],** K**[**MAXM**];** //存查询

int nums**[**MAXN**];** //对A排序后的数组

vector**<**int**>** dat**[**ST\_SIZE**];** //线段树节点数组

//构建线段树

// k是节点编号， 和区间[l, r)对应

void init**(**int k**,** int l**,** int r**)** **{**

**if** **(**r **-** l **==** 1**)** **{**

dat**[**k**].**push\_back**(**A**[**l**]);**

**}** **else** **{**

int lch **=** k **\*** 2 **+** 1**,** rch **=** 2 **\*** k **+** 2**;**

init**(**lch**,** l**,** **(**l **+** r**)** **/** 2**);**

init**(**rch**,** **(**l **+** r**)** **/** 2**,** r**);**

dat**[**k**].**resize**(**r **-** l**);** //也不知道有什么用，估计是缩小空间吧

//利用STL的merge函数把两个儿子的数列合并

merge**(**dat**[**lch**].**begin**(),** dat**[**lch**].**end**(),** dat**[**rch**].**begin**(),** dat**[**rch**].**end**(),** dat**[**k**].**begin**());**

**}**

**}**

//计算[i, j)中不超过x的数的个数

//k是节点编号， 和区间[l, r)对应 (一开始 k l r 0 0 N）

int query**(**int i**,** int j**,** int x**,** int k**,** int l**,** int r**)** **{**

**if** **(**j **<=** l **||** r **<=** i**)** **{**

//完全不相交

**return** 0**;**

**}** **else** **if** **(**i **<=** l **&&** r **<=** j**)** **{**

//完全包含

**return** upper\_bound**(**dat**[**k**].**begin**(),** dat**[**k**].**end**(),** x**)** **-** dat**[**k**].**begin**();**

**}** **else** **{**

//对儿子们递归计算

int lc **=** query**(**i**,** j**,** x**,** k **\*** 2 **+** 1**,** l**,** **(**l **+** r**)** **/** 2**);**

int rc **=** query**(**i**,** j**,** x**,** k **\*** 2 **+** 2**,** **(**l **+** r**)** **/** 2**,** r**);**

**return** lc **+** rc**;**

**}**

**}**

void solve**()** **{**

**for(**int i **=** 0**;** i **<** N**;** i**++)** nums**[**i**]** **=** A**[**i**];**

sort**(**nums**,** nums **+** N**);**

init**(**0**,** 0**,** N**);**

**for** **(**int i **=** 0**;** i **<** M**;** i**++)** **{**

//[l, r)

int l **=** I**[**i**]** **-** 1**,** r **=** J**[**i**],** k **=** K**[**i**];**

int lb **=** **-**1**,** ub **=** N **-** 1**;** //(-1, N-1]

**while(**ub **-** lb **>** 1**)** **{**

int md **=** **(**ub **+** lb**)** **/** 2**;**

int ans **=** query**(**l**,** r**,** nums**[**md**],** 0**,** 0**,** N**);**

**if** **(**ans **>=** k**)** ub **=** md**;**

**else** lb **=** md**;**

**}**

printf**(**"%d\n"**,** nums**[**ub**]);**

**}**

**}**

int main**() {**

cin **>>** N **>>** M**;**

**for** **(**int i **=** 0**;** i **<** N**;** i**++)** **{**

scanf**(**"%d"**,** A **+** i**);**

**}**

**for** **(**int i **=** 0**;** i **<** M**;** i**++)** **{**

scanf**(**"%d%d%d"**,** I **+** i**,** J **+** i**,** K **+** i**);**

**}**

solve**();**

**return** 0**;**

**}**

# 欧拉筛：

#include <cstring>

**using** **namespace** std**;**

int prime**[**1100000**],**primesize**,**phi**[**11000000**];**

bool isprime**[**11000000**];**

void getlist**(**int listsize**)**

**{**

memset**(**isprime**,**1**,sizeof(**isprime**));**

isprime**[**1**]=false;**

**for(**int i**=**2**;**i**<=**listsize**;**i**++)**

**{**

**if(**isprime**[**i**])**prime**[++**primesize**]=**i**;**

**for(**int j**=**1**;**j**<=**primesize**&&**i**\***prime**[**j**]<=**listsize**;**j**++)**

**{**

isprime**[**i**\***prime**[**j**]]=false;**

**if(**i**%**prime**[**j**]==**0**)break;**

**}**

**}**

**}**

# 输入挂：

inline bool scan\_d**(**int **&**num**)**

**{**

char in**;**bool IsN**=false;**

in**=**getchar**();**

**if(**in**==**EOF**)** **return** **false;**

**while(**in**!=**'-'**&&(**in**<**'0'**||**in**>**'9'**))** in**=**getchar**();**

**if(**in**==**'-'**){** IsN**=true;**num**=**0**;}**

**else** num**=**in**-**'0'**;**

**while(**in**=**getchar**(),**in**>=**'0'**&&**in**<=**'9'**){**

num**\*=**10**,**num**+=**in**-**'0'**;**

**}**

**if(**IsN**)** num**=-**num**;**

**return** **true;**

**}**

# 区间第K大的数——平方分割法实现

const int B **=** 1000**;**

const int MAXN **=** 1e5 **+** 5**;**

const int MAXM **=** 5005**;**

int N**,** M**;**

int A**[**MAXN**];** // 要处理的数组

int I**[**MAXM**],** J**[**MAXM**],** K**[**MAXM**];** //存查询

int nums**[**MAXN**];** //对A排序后的数组

vector**<**int**>** bucket**[**MAXN **/** B**];**

void solve**()** **{**

//初始化

**for(**int i **=** 0**;** i **<** N**;** i**++)** **{**

bucket**[**i **/** B**].**push\_back**(**A**[**i**]);**

nums**[**i**]** **=** A**[**i**];**

**}**

**for(**int i **=** 0**;** i **<** N **/** B**;** i**++)** **{** // 桶内排序

sort**(**bucket**[**i**].**begin**(),** bucket**[**i**].**end**());**

**}**

sort**(**nums**,** nums **+** N**);**

**for** **(**int i **=** 0**;** i **<** M**;** i**++)** **{**

//求区间[l, r)中的第k个数

int l **=** I**[**i**]** **-** 1**,** r **=** J**[**i**],** k **=** K**[**i**];**

int lb **=** **-**1**,** ub **=** N **-** 1**;** //二分下标(-1, N-1]

**while(**ub **-** lb **>** 1**)** **{**

int md **=** **(**lb **+** ub**)** **>>** 1**;**

int x **=** nums**[**md**];**

int c **=** 0**,** tl **=** l**,** tr **=** r**;**

// 区间多余部分

**while(**tl **<** tr **&&** tl **%** B **!=** 0**)** **if** **(**A**[**tl**++]** **<=** x**)** c**++;** //我也不知道为什么反正就是《=

**while(**tl **<** tr **&&** tr **%** B **!=** 0**)** **if** **(**A**[--**tr**]** **<=** x**)** c**++;**

// 桶内计算

**while(**tl **<** tr**)** **{**

c **+=** upper\_bound**(**bucket**[**tl **/** B**].**begin**(),** bucket**[**tl **/** B**].**end**(),** x**)** **-** bucket**[**tl **/** B**].**begin**();**//我也不知道为什么反正就是upper\_bound

tl **+=** B**;**

**}**

**if** **(**c **>=** k**)** ub **=** md**;**

**else** lb **=** md**;**

**}**

printf**(**"%d\n"**,** nums**[**ub**]);**

**}**

**}**

int main**()**

**{**

cin **>>** N **>>** M**;**

**for** **(**int i **=** 0**;** i **<** N**;** i**++)** **{**

scanf**(**"%d"**,** A **+** i**);**

**}**

**for** **(**int i **=** 0**;** i **<** M**;** i**++)** **{**

scanf**(**"%d%d%d"**,** I **+** i**,** J **+** i**,** K **+** i**);**

**}**

solve**();**

**return** 0**;**

**}**

# 区间求和线段树

const int MAXN **=** 1 **<<** 17**;** //131072

int n**,** dat**[**2 **\*** MAXN **-** 1**];**

int init**(**int n\_**)** **{**

n **=** 1**;**

**while(**n **<** n\_**)** n **\*=** 2**;**

**for(**int i **=** 0**;** i **<** 2 **\*** n **-** 1**;** i**++)** dat**[**i**]** **=** 0**;**

**return** n**;**

**}**

void add**(**int k**,** int a**)** **{**

k **+=** n **-** 1**;**

dat**[**k**]** **+=** a**;**

**while(**k **>** 0**)** **{**

k **=** **(**k **-** 1**)** **/** 2**;**

dat**[**k**]** **=** dat**[**2 **\*** k **+** 1**]** **+** dat**[**2 **\*** k **+** 2**];**

**}**

**}**

void sub**(**int k**,** int a**)** **{**

k **+=** n **-** 1**;**

dat**[**k**]** **-=** a**;**

**while(**k **>** 0**)** **{**

k **=** **(**k **-** 1**)** **/** 2**;**

dat**[**k**]** **=** dat**[**2 **\*** k **+** 1**]** **+** dat**[**2 **\*** k **+** 2**];**

**}**

**}**

int query**(**int a**,** int b**,** int k**,** int l**,** int r**)** **{** //[a,b) 节点编号 当前查询范围

**if** **(**r **<=** a **||** l **>=** b**)** **{**

**return** 0**;**

**}**

**if** **(**a **<=** l **&&** r **<=** b**)** **{**

**return** dat**[**k**];**

**}** **else** **{**

int vl **=** query**(**a**,** b**,** k **\*** 2 **+** 1**,** l**,** **(**l **+** r**)** **/** 2**);**

int vr **=** query**(**a**,** b**,** k **\*** 2 **+** 2**,** **(**l **+** r**)** **/** 2**,** r**);**

**return** vl **+** vr**;**

**}**

**}**

void solve**()**

**{**

int T**;**

cin **>>** T**;**

**for(**int k **=** 1**;** k **<=** T**;** k**++)** **{**

int N**;**

cin **>>** N**;**

int n **=** init**(**N**);**

**for(**int i **=** 0**;** i **<** N**;** i**++)** **{**

int t**;**

scanf**(**"%d"**,** **&**t**);**

add**(**i**,** t**);**

**}**

char s**[**50**];**

int num**,** val**;**

printf**(**"Case %d:\n"**,** k**);**

**while(**scanf**(**"%s"**,** s**)** **!=** EOF**)** **{**

**if** **(**strcmp**(**s**,**"End"**)** **==** 0**)** **{**

**break;**

**}** **else** **if** **(**strcmp**(**s**,** "Query"**)** **==** 0**)** **{**

int a**,** b**;**

scanf**(**"%d%d"**,** **&**a**,** **&**b**);**

a**--;**

cout **<<** query**(**a**,** b**,** 0**,** 0**,** n**)** **<<** endl**;**

**}** **else** **if** **(**strcmp**(**s**,** "Sub"**)** **==** 0**)** **{**

int k**,** a**;**

scanf**(**"%d%d"**,** **&**k**,** **&**a**);**

k**--;**

sub**(**k**,** a**);**

**}** **else** **if** **(**strcmp**(**s**,** "Add"**)** **==** 0**)** **{**

int k**,** a**;**

scanf**(**"%d%d"**,** **&**k**,** **&**a**);**

k**--;**

add**(**k**,** a**);**

**}**

**}**

**}**

# 树形DP

**using** **namespace** std**;**

const int MAXN **=** 6e3 **+** 5**;**

int n**;**

int dp**[**MAXN**][**2**]** **=** **{**0**},** father**[**MAXN**]** **=** **{**0**};**//dp[i][0]0表示不去，dp[i][1]1表示去了

bool vis**[**MAXN**]** **=** **{**0**};**

void tree\_dp**(**int node**)** **{**

vis**[**node**]** **=** 1**;**

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

**if** **(!**vis**[**i**]** **&&** father**[**i**]** **==** node**)** **{**

tree\_dp**(**i**);**//递归调用孩子结点，从叶子结点开始dp

//关键

dp**[**node**][**1**]** **+=** dp**[**i**][**0**];**//上司来,下属不来

dp**[**node**][**0**]** **+=** max**(**dp**[**i**][**1**],**dp**[**i**][**0**]);**//上司不来，下属来、不来

**}**

**}**

**}**

int main**()** **{**

cin **>>** n**;**

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

scanf**(**"%d"**,** **&**dp**[**i**][**1**]);**

**}**

int x**,** y**;**

**while(**scanf**(**"%d%d"**,** **&**x**,** **&**y**),** x **+** y**)** **{**

father**[**x**]** **=** y**;**

**}**

int root **=** 1**;**

**while(**father**[**root**])** **{**

root **=** father**[**root**];**

**}**

tree\_dp**(**root**);**

printf**(**"%d\n"**,** max**(**dp**[**root**][**0**],** dp**[**root**][**1**]));**

**}**

# 树状数组：

const int MAXN **=** 100000 **+** 5**;**

**typedef** long long LL**;**

int bit**[**MAXN **+** 1**],** a**[**MAXN **+** 1**],** b**[**MAXN **+** 1**],** n**;** //[1, n]

int sum**(**int i**)** **{**

int s **=** 0**;**

**while(**i **>** 0**)** **{**

s **+=** bit**[**i**];**

i **-=** i **&** **-**i**;**

**}**

**return** s**;**

**}**

void add**(**int i**,** int x**)** **{**

**while(**i **<=** n**)** **{**

bit**[**i**]** **+=** x**;**

i **+=** i **&** **-**i**;**

**}**

**}**

int main**()**

**{**

**for(**int k **=** 1**;** k **<=** T**;** k**++)** **{**

cin **>>** n**;**

fill**(**bit**,** bit **+** n **+** 1**,** 0**);**

printf**(**"Case #%d: "**,** k**);**

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

scanf**(**"%d"**,** a **+** i**);**

add**(**a**[**i**],** 1**);**

b**[**a**[**i**]]** **=** i **+** a**[**i**]** **-** 1 **-** sum**(**a**[**i**]** **-** 1**)** **-** min**(**i**,** a**[**i**]);**

**}**

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

printf**(**"%d%c"**,** b**[**i**],** **(**i **==** n **?** '\n' **:** ' '**));**

**}**

**}**

**}**

# 头文件：

#include <iostream>

#pragma comment(linker, "/STACK:1024000000,1024000000")

#include <stdio.h>

#include <fstream>

#include <iomanip>

#include <cmath>

#include <string>

#include <string.h>

#include <sstream>

#include <cctype>

#include <climits>

#include <set>

#include <map>

#include <deque>

#include <queue>

#include <vector>

#include <iterator>

#include <algorithm>

#include <stack>

#include <functional>

/\*int类型最大值INT\_MAX，short最大值为SHORT\_MAX

long long最大值为LONG\_LONG\_MAX\*/

//cout << "OK" << endl;

#define \_clr(x,y) memset(x,y,sizeof(x))

#define \_inf(x) memset(x,0x3f,sizeof(x))

#define pb push\_back

#define mp make\_pair

#define FORD(i,a,b) for (int i=(a); i<=(b); i++)

#define FORP(i,a,b) for (int i=(a); i>=(b); i--)

#define REP(i,n) for (int i=0; i<(n); i++)

**using** **namespace** std**;**

const int INF **=** 0x3f3f3f3f**;**

const double eps **=** 1e-8**;**

const double EULER **=** 0.577215664901532860**;**

const double PI **=** 3.1415926535897932384626**;**

const double E **=** 2.71828182845904523536028**;**

**typedef** long long LL**;**

# 树状数组求逆序对

#define INT\_MAX 2147483647

**using** **namespace** std**;**

const int MAXN **=** 1 **<<** 13**;** //1024 \* 8

int n**,** dat**[**2 **\*** MAXN **-** 1**],** num**[**MAXN**];** // 维护dat表示大小为i的数有几个 ， num记录输入数据

int init**(**int n\_**)** **{**

n **=** 1**;**

**while(**n **<** n\_**)** n **\*=** 2**;**

**for(**int i **=** 0**;** i **<** 2 **\*** n **-** 1**;** i**++)** dat**[**i**]** **=** 0**;**

**for(**int i **=** 0**;** i **<** n**;** i**++)** num**[**i**]** **=** **-**1**;**

**return** n**;**

**}**

void add**(**int k**,** int a**)** **{**

k **+=** n **-** 1**;**

dat**[**k**]** **+=** a**;**

**while(**k **>** 0**)** **{**

k **=** **(**k **-** 1**)** **/** 2**;**

dat**[**k**]** **=** dat**[**2 **\*** k **+** 1**]** **+** dat**[**2 **\*** k **+** 2**];**

**}**

**}**

int query**(**int a**,** int b**,** int k**,** int l**,** int r**)** **{** //[a,b) 节点编号 当前查询范围

**if** **(**r **<=** a **||** l **>=** b**)** **{**

**return** 0**;**

**}**

**if** **(**a **<=** l **&&** r **<=** b**)** **{**

**return** dat**[**k**];**

**}** **else** **{**

int vl **=** query**(**a**,** b**,** k **\*** 2 **+** 1**,** l**,** **(**l **+** r**)** **/** 2**);**

int vr **=** query**(**a**,** b**,** k **\*** 2 **+** 2**,** **(**l **+** r**)** **/** 2**,** r**);**

**return** vl **+** vr**;**

**}**

**}**

void solve**()**

**{**

int N**;**

**while(**cin **>>** N**)** **{**

int n **=** init**(**N**+**1**);**

int sum **=** 0**;**

**for(**int i **=** 0**;** i **<** N**;** i**++)** **{**

int a**;**

scanf**(**"%d"**,** **&**a**);**

num**[**i**]** **=** a**;**

add**(**a**,** 1**);**

sum **+=** query**(**a**+**1**,** N**+**1**,** 0**,** 0**,** n**);** //求比它大的数的个数和 logN

**}**

int ans **=** sum**;**

**for(**int i **=** 0**;** i **<** N **-** 1**;** i**++)** **{**

sum **=** sum **-** 2 **\*** num**[**i**]** **+** N **-** 1**;** // 循环移1位

ans **=** min**(**ans**,** sum**);**

**}**

cout **<<** ans **<<** endl**;**

**}**

**}**

int main**()** **{**

solve**();**

**}**