The black aker

目录

[Data\_Structure 1](#_Toc514385902)

[cdq 1](#_Toc514385903)

[chain\_div 2](#_Toc514385904)

[leftTree 4](#_Toc514385905)

[linkCutTree 4](#_Toc514385906)

[mergingTree 5](#_Toc514385907)

[partitionTree 6](#_Toc514385908)

[undo\_dsu 7](#_Toc514385909)

[Geometry 10](#_Toc514385910)

[rotateCalip 10](#_Toc514385911)

[Graph 11](#_Toc514385912)

[costFlow\_dijkstra 11](#_Toc514385913)

[dijkstra\_heap 12](#_Toc514385914)

[dinic & isap 12](#_Toc514385915)

[dominatorTree 13](#_Toc514385916)

[euler 14](#_Toc514385917)

[floyd 15](#_Toc514385918)

[hungarian 15](#_Toc514385919)

[kosaraju 16](#_Toc514385920)

[kruskal 16](#_Toc514385921)

[lca\_doubling 17](#_Toc514385922)

[matrixTree 17](#_Toc514385923)

[maximumDensitySubgraph 17](#_Toc514385924)

[maximumWeightClosureGraph 19](#_Toc514385925)

[mixedEuler 20](#_Toc514385926)

[mstCounting 21](#_Toc514385927)

[pnt\_d&c 22](#_Toc514385928)

[prim\_heap 23](#_Toc514385929)

[tarjan 24](#_Toc514385930)

[threeMemberedRing 24](#_Toc514385931)

[zhuliu 25](#_Toc514385932)

[Math 26](#_Toc514385933)

[combination\_pre 26](#_Toc514385934)

[FFT 26](#_Toc514385935)

[NTT 27](#_Toc514385936)

[xor linear basis 28](#_Toc514385937)

[Search 29](#_Toc514385938)

[exactCover 29](#_Toc514385939)

[multiCover 30](#_Toc514385940)

[sudoku 31](#_Toc514385941)

[String 33](#_Toc514385942)

[KMP 33](#_Toc514385943)

[manacher 33](#_Toc514385944)

[ahoCorasickAutomaton 33](#_Toc514385945)

[palindromeAutomaton 34](#_Toc514385946)

[suffixAutomaton 35](#_Toc514385947)

[smallestRepresentation 36](#_Toc514385948)

[Others 37](#_Toc514385949)

[BM 37](#_Toc514385950)

[contor 38](#_Toc514385951)

[input\_plug 38](#_Toc514385952)

[java 40](#_Toc514385953)

[vimrc\_long 42](#_Toc514385954)

[vimrc\_short 43](#_Toc514385955)

# Data\_Structure

## cdq

/\*\*\*\*\*\*\*\*\*\*\*\*三维偏序问题\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

const int maxn **=** 1e5**+**100**;**

int t**,**n**,**bit**[**maxn**],**ans**[**maxn**];**

struct que

**{**

int a**,**b**,**c**,**idx**;**

**}** q**[**maxn**],** tmp**[**maxn**];**

int cmp**(**que a**,** que b**){**

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

**else** **if(**a**.**b **!=** b**.**b**)** **return** a**.**b **<** b**.**b**;**

**else** **return** a**.**c **<** b**.**c**;**

**}**

int cmp1**(**que a**,** que b**){**

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

**else** **return** a**.**c **<** b**.**c**;**

**}**

int inline lowbit**(**int x**){**

**return** x**&-**x**;**

**}**

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

**for(**int i **=** x**;** i **<** maxn**;** i **+=** lowbit**(**i**)){**

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

**}**

**}**

void reset**(**int p**){**

**while(**p**<=**100000**){**

bit**[**p**]=**0**;**

p**+=**lowbit**(**p**);**

**}**

**}**

int sum**(**int x**){**

int res **=** 0**;**

**for(**int i **=** x**;** i **>** 0**;** i **-=** lowbit**(**i**)){**

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

**}**

**return** res**;**

**}**

void cdq**(**int l**,** int r**){**

/\*l,r为第一维,分治中对第二维进行排序，第三维bit维护\*/

**if(**l **==** r**)** **return;**

int m **=** **(**l**+**r**)/**2**;**

cdq**(**l**,**m**);**

cdq**(**m**+**1**,**r**);**

**for(**int i **=** l**;** i **<=** r**;** i**++)** tmp**[**i**]** **=** q**[**i**];**

sort**(**tmp**+**l**,**tmp**+**m**+**1**,**cmp1**);**

sort**(**tmp**+**m**+**1**,**tmp**+**r**+**1**,**cmp1**);**

int j **=** l**;**

/\*处理[l,m]之中的修改对[m+1,r]中的询问的影响\*/

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

**while(**j**<=**m**&&**tmp**[**j**].**b**<=**tmp**[**i**].**b**){**

add**(**tmp**[**j**++].**c**,**1**);**

**}**

ans**[**tmp**[**i**].**idx**]+=**sum**(**tmp**[**i**].**c**);**

**}**

/\*reset bit\*/

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

reset**(**tmp**[**i**].**c**);**

**}**

**}**

int main**(){**

sc**(**t**);**

**while(**t**--){**

sc**(**n**);**

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

scanf**(**"%d%d%d"**,** **&**q**[**i**].**a**,** **&**q**[**i**].**b**,** **&**q**[**i**].**c**);**

q**[**i**].**idx **=** i**;**

**}**

sort**(**q**,** q**+**n**,**cmp**);**

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

cdq**(**0**,** n**-**1**);**

**for(**int i **=** n**-**2**;** i **>=** 0**;** i**--){**

**if(**q**[**i**].**a **==** q**[**i**+**1**].**a **&&** q**[**i**].**b **==** q**[**i**+**1**].**b **&&** q**[**i**].**c **==** q**[**i**+**1**].**c**)**

ans**[**q**[**i**].**idx**]** **=** ans**[**q**[**i**+**1**].**idx**];**

**}**

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

printf**(**"%d\n"**,** ans**[**i**]);**

**}**

**return** 0**;**

**}**

## chain\_div

///暂时只更新(点权修改,链上点权值和/最大值询问),例题bzoj1036

const int maxn **=** 1e5**+**10**;**

///树链剖分模板

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int val\_p**[**maxn**],** val\_e**[**maxn**],** ///点权与边权

to**[**maxn**\***2**],** nxt**[**maxn**\***2**],**head**[**maxn**\***2**],** tot**;**

void tree\_init**(){**

tot **=** 0**;**

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

**}**

void add\_edg**(**int u**,** int v**,** int w **=** 0**){**

tot**++;** val\_e**[**tot**]** **=** w**;** to**[**tot**]** **=** v**;**

nxt**[**tot**]** **=** head**[**u**];** head**[**u**]** **=** tot**;**

**}**

void read\_edg**(**int op **=** 1**){**///op=1 --> 双向边; op=0 --> 单向边

int u**,** v**,** w**;**

scanf**(**"%d%d%d"**,** **&**u**,** **&**v**,** **&**w**);**

add\_edg**(**u**,** v**,** w**);**

**if(**op**)** add\_edg**(**v**,** u**,** w**);**

**}**

int tid**[**maxn**],** ///dfs序/tid[i]-->节点i在线段树中的位置

rk**[**maxn**],** ///rk[i]-->dfs序第i个在树中的节点编号

top**[**maxn**],** ///所在链顶点

sz**[**maxn**],** ///子树size

son**[**maxn**],** ///重儿子

dep**[**maxn**],** ///深度

fa**[**maxn**],** ///父节点

idx**;** ///dfs序计数器

void dfs1**(**int u**,** int f**){**

///更新dep[u], fa[u], sz[u], son[u]

dep**[**u**]** **=** dep**[**f**]+**1**;**

fa**[**u**]** **=** f**;**

sz**[**u**]** **=** 1**;**

**for(**int i **=** head**[**u**];** i**;** i **=** nxt**[**i**]){**

int v **=** to**[**i**];**

**if(**v **==** f**)** **continue;**

dfs1**(**v**,** u**);**

sz**[**u**]** **+=** sz**[**v**];**

**if(**son**[**u**]** **<** 0 **||** sz**[**v**]** **>** sz**[**son**[**u**]])**

son**[**u**]** **=** v**;**

**}**

**}**

void dfs2**(**int u**,** int tp**){**

///更新top[u], tid[u], rk[u]

top**[**u**]** **=** tp**;**

tid**[**u**]** **=** idx**;**

rk**[**idx**++]** **=** u**;**

**if(**son**[**u**]** **>=** 0**)** dfs2**(**son**[**u**],** tp**);**

**for(**int i **=** head**[**u**];** i**;** i **=** nxt**[**i**]){**

int v **=** to**[**i**];**

**if(**v **==** fa**[**u**]** **||** v **==** son**[**u**])** **continue;**

dfs2**(**v**,** v**);**

**}**

**}**

void chain\_div**(**int \_n**,** int rt **=** 1**){**///rt取决于题目是否为有根树

memset**(**son**,** **-**1**,** **sizeof(**int**)\*(**\_n**+**5**));**

dep**[**rt**]** **=** idx **=** 0**;**

dfs1**(**rt**,** rt**);**

dfs2**(**rt**,** rt**);**

**}**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

///线段树和调用部分仅作参考，赛场上自行发挥

int mx**[**maxn**\***3**],** val**[**maxn**],** n**,** sum**[**maxn**\***3**];**

void pick\_up**(**int rt**){**

mx**[**rt**]** **=** max**(**mx**[**lson**],** mx**[**rson**]);**

sum**[**rt**]** **=** sum**[**lson**]** **+** sum**[**rson**];**

**}**

void build**(**int rt **=** 0**,** int l **=** 0**,** int r **=** n**-**1**){**

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

mx**[**rt**]** **=** sum**[**rt**]** **=** val**[**l**];**

**return;**

**}**

build**(**lson**,** l**,** mid**);**

build**(**rson**,** mid**+**1**,** r**);**

pick\_up**(**rt**);**

**}**

void update**(**int x**,** int v**,** int rt **=** 0**,** int l **=** 0**,** int r **=** n**-**1**){**

**if(**l **>** x **||** r **<** x**)** **return;**

**if(**l **==** x **&&** r **==** x**){**

mx**[**rt**]** **=** sum**[**rt**]** **=** v**;**

**return;**

**}**

update**(**x**,** v**,** lson**,** l**,** mid**);**

update**(**x**,** v**,** rson**,** mid**+**1**,** r**);**

pick\_up**(**rt**);**

**}**

inline pii add**(**pii a**,** pii b**){**

**return** pii**(**max**(**a**.**fi**,**b**.**fi**),** a**.**se**+**b**.**se**);**

**}**

pii query**(**int x**,** int y**,** int rt **=** 0**,** int l **=** 0**,** int r **=** n**-**1**){**

**if(**l **>** y **||** r **<** x**)** **return** mk**(-**INF**,** 0**);**

**if(**l **>=** x **&&** r **<=** y**){**

**return** mk**(**mx**[**rt**],** sum**[**rt**]);**

**}**

pii a **=** query**(**x**,** y**,** lson**,** l**,** mid**),**

b **=** query**(**x**,** y**,** rson**,** mid**+**1**,** r**);**

**return** add**(**a**,**b**);**

**}**

pii solve**(**int u**,** int v**){**

pii ans **=** mk**(-**INF**,** 0**);**

**while(**top**[**u**]** **!=** top**[**v**]){**

**if(**dep**[**top**[**u**]]** **<** dep**[**top**[**v**]])** swap**(**u**,**v**);**

ans **=** add**(**ans**,** query**(**tid**[**top**[**u**]],** tid**[**u**]));**

u **=** fa**[**top**[**u**]];**

**}**

**if(**dep**[**u**]** **>** dep**[**v**])** swap**(**u**,** v**);**

ans **=** add**(**ans**,** query**(**tid**[**u**],** tid**[**v**]));**

**return** ans**;**

**}**

int main**(){**

tree\_init**();**

sc**(**n**);**

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

int u**,** v**;** sc**(**u**);** sc**(**v**);**

add\_edg**(**u**,**v**);**

add\_edg**(**v**,**u**);**

**}**

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

sc**(**val\_p**[**i**]);**

chain\_div**(**n**);**

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

val**[**tid**[**i**]]** **=** val\_p**[**i**];**

build**();**

int q**;** sc**(**q**);**

**while(**q**--){**

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

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

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

**if(**s**[**1**]** **==** 'M'**){**

printf**(**"%d\n"**,** solve**(**x**,**y**).**fi**);**

**}else** **if(**s**[**1**]** **==** 'H'**){**

update**(**tid**[**x**],** y**);**

**}else{**

printf**(**"%d\n"**,** solve**(**x**,**y**).**se**);**

**}**

**}**

**return** 0**;**

**}**

## leftTree

/\*\*\*\*\*\*\*左偏树模板,pop,push等功能自行实现\*\*\*\*\*\*\*\*\*\*/

struct Heap**{**

int l**,** r**,** v**,** d**;**

**}** heap**[**maxn**];**

int Merge**(**int x**,** int y**){**

**if(!**x **||** **!**y**)** **return** x**+**y**;**

///大根堆

///if(heap[x].v < heap[y].v) swap(x,y);

///小根堆

///if(headp[x].v > heap[y].v) swap(x,y);

heap**[**x**].**r **=** Merge**(**heap**[**x**].**r**,** y**);**

///保持左端深度更大

**if(**heap**[**heap**[**x**].**l**].**d **<** heap**[**heap**[**x**].**r**].**d**)**

swap**(**heap**[**x**].**l**,** heap**[**x**].**r**);**

heap**[**x**].**d **=** heap**[**heap**[**x**].**r**].**d **+** 1**;**

**return** x**;**

**}**

## linkCutTree

#define rev(x) (swap(l[x],r[x]),b[x]^=1)

#define judge(x) (l[f[x]]==x||r[f[x]]==x)

#define root(x) (access(x),modify(x),rev(x)) //LCA换根

int l**[**maxn**],**r**[**maxn**],**f**[**maxn**],**b**[**maxn**],**mx**[**maxn**],**v**[**maxn**];**

struct edge **{**int st**,**ed**,**a**,**b**;}** x**[**maxm**];**

bool find**(**int x**,**int y**)** //判断是否在同一棵树

**{**

**while(**f**[**x**])**x**=**f**[**x**];**

**while(**f**[**y**])**y**=**f**[**y**];**

**return** x**==**y**;**

**}**

void update**(**int t**)** //更新splay上维护的信息

**{**

mx**[**t**]=**max**(**mx**[**l**[**t**]],**mx**[**r**[**t**]]);**

mx**[**t**]=**max**(**mx**[**t**],**v**[**t**]);**

**}**

void rotate**(**int x**,**int l**[],**int r**[])** //splay旋转

**{**

int t**=**f**[**x**];**

f**[**l**[**t**]=**r**[**x**]]=**t**;**f**[**x**]=**f**[**t**];**

**if(**judge**(**t**))**l**[**f**[**t**]]==**t**?**l**[**f**[**t**]]=**x**:**r**[**f**[**t**]]=**x**;**

f**[**r**[**x**]=**t**]=**x**;**update**(**r**[**x**]);**

**}**

void deal**(**int x**)** //splay标记下传

**{**

**if(**judge**(**x**))**deal**(**f**[**x**]);**

**if(**b**[**x**])**b**[**x**]^=**1**,**rev**(**l**[**x**]),**rev**(**r**[**x**]);**

**}**

void modify**(**int x**)** //splay换根

**{**

deal**(**x**);**

**while(**judge**(**x**))if(**l**[**f**[**x**]]==**x**)**

**{**

**if(**judge**(**f**[**x**])&&**l**[**f**[**f**[**x**]]]==**f**[**x**])**

rotate**(**f**[**x**],**l**,**r**);**

rotate**(**x**,**l**,**r**);**

}

else

{

if(judge(f[x])&&r[f[f[x]]]==f[x])

rotate(f[x],r,l);

rotate(x,r,l);

}

update(x);

}

void access(int x) //路径变重边

{

for(int y=0;x;y=x,x=f[x])

modify(x),l[x]=y,update(x);

}

void link(int x,int y) //加边

{

root(x);

f[x]=y;

}

void cut(int x,int y) //删边

{

root(x);access(y);

modify(y);

f[x]=r[y]=0;

}

int quiry(int z,int y) //查询两点之间权最大的点

{

if(!find(z,y))return 2e9;

root(z);

access(y);modify(y);

while(v[y]!=mx[y])

y=(mx[l[y]]==mx[y]?l[y]:r[y]);

return y;

}

## mergingTree

#include <vector>

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

#define pb push\_back

#define ALL(x) x.begin(), x.end()

int n**,** m**;**

const int maxn **=** 1e5 **+** 10**;**

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

vector**<**int**>** dat**[**ST\_SIZE**];**

int A**[**maxn**];**

int nums**[**maxn**];**

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

**{**

**if(**r**-**l **==** 1**)**dat**[**k**].**pb**(**A**[**l**]);**

**else{**

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

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

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

dat**[**k**].**resize**(**r**-**l**);**

copy**(**ALL**(**dat**[**lch**]),** dat**[**k**].**begin**());**

copy**(**ALL**(**dat**[**rch**]),** dat**[**k**].**begin**()+**dat**[**lch**].**size**());**

inplace\_merge**(**dat**[**k**].**begin**(),** dat**[**k**].**begin**()+**dat**[**lch**].**size**(),** dat**[**k**].**end**());**

**}**

**}**

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

**{**

**if(**j **<=** l **||** r **<=** i**)return** 0**;**

**else** **if(**i **<=** l **&&** r **<=** j**)return** upper\_bound**(**ALL**(**dat**[**k**]),** x**)** **-** dat**[**k**].**begin**();**

**else{**

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**()**

**{**

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

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

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

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

**}**

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

init**(**0**,** 0**,** n**);**

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

int l, r, k;

scanf("%d%d%d", &l, &r, &k);

l--;

int ls = 0, rs = n - 1, ans = 0;

while(ls <= rs){

int mid = (ls + rs) / 2;

int c = query(l, r, nums[mid], 0, 0, n);

if(c >= k){

rs = mid - 1;

ans = mid;

}

else ls = mid + 1;

}

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

}

}

## partitionTree

#include <cstdio>

#include <algorithm>

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

const int maxn **=** 1e5 **+** 10**;**

const int maxd **=** 18**;**

**typedef** struct

**{**

int num**[**maxn**];**

int cnt**[**maxn**];**

**}**partition\_tree**;**

partition\_tree tree**[**maxd**];**

int sorted**[**maxn**];**

void build**(**int d**,** int l**,** int r**)**

**{**

**if(**l **==** r**)return;**

int m **=** **(**l **+** r**)** **>>** 1**;**

int need **=** m **-** l **+** 1**;**

**for(**int i **=** l**;** i **<=** r**;** i**++)if(**tree**[**d**].**num**[**i**]** **<** sorted**[**m**])**need**--;**

int p **=** l**,** q **=** m **+** 1**;**

**for(**int i **=** l**,** cnt **=** 0**;** i **<=** r**;** i**++){**

int num **=** tree**[**d**].**num**[**i**];**

**if(**num **<** sorted**[**m**]** **||** **(**num **==** sorted**[**m**]** **&&** need**)){**

**if(**num **==** sorted**[**m**])**need**--;**

cnt**++;**

tree**[**d **+** 1**].**num**[**p**++]** **=** num**;**

**}**

**else** tree**[**d **+** 1**].**num**[**q**++]** **=** num**;**

tree**[**d**].**cnt**[**i**]** **=** cnt**;**

**}**

build**(**d **+** 1**,** l**,** m**);**

build**(**d **+** 1**,** m **+** 1**,** r**);**

**}**

int query**(**int d**,** int l**,** int r**,** int ql**,** int qr**,** int k**)**

**{**

**if(**l **==** r**)return** tree**[**d**].**num**[**l**];**

int ly**;**

**if(**l **==** ql**)**ly **=** 0**;**

**else** ly **=** tree**[**d**].**cnt**[**ql**-**1**];**

int tot **=** tree**[**d**].**cnt**[**qr**]** **-** ly**;**

int newl**,** newr**,** mid **=** **(**l **+** r**)** **/** 2**;**

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

newl **=** l **+** ly**;**

newr = newl + tot - 1;

return query(d + 1, l, mid, newl, newr, k);

}

else{

newl = mid + 1 + (ql - l - ly);

newr = newl + ((qr-ql+1)-tot) - 1;

return query(d + 1, mid + 1, r, newl, newr, k - tot);

}

}

void solve()

{

int n, m;

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

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

scanf("%d", &tree[0].num[i]);

sorted[i] = tree[0].num[i];

}

sort(sorted+1, sorted+n+1);

build(0, 1, n);

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

int l, r, k;

scanf("%d%d%d", &l, &r, &k);

int ans = query(0, 1, n, l, r, k);

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

}

}

## undo\_dsu

/\*\*\*四种操作:1.连x,y; 2.删掉边x,y; 3.询问x所在连通块的size; 4.询问x,y是否连通\*\*\*/

/\*\*\*大致思路:记录每个连边的持续时间\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*solve(l,r)碰到持续时间和l,r一样的操作就添加,回溯时撤销\*\*\*\*\*\*\*\*\*\*\*\*/

#include <bits/stdc++.h>

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

const int N **=** 2e5**+**5**;**

**typedef** pair**<**int**,**int**>** pii**;**

class jobs**{**

public**:**

int type**;**

int x**,**y**,**l**,**r**;**

jobs**()** **{}**

jobs**(**int a**,**int b**,**int c**,**int d**,**int e**)** **:** type**(**a**),**x**(**b**),**y**(**c**),**l**(**d**),**r**(**e**)** **{}**

**};**

int n**,**m**,**x**,**y**,**op**,**ans**[**N**],**siz**[**N**],**f**[**N**];**

unordered\_map **<**long long**,**int**>** pre**;**

vector**<**jobs**>** job**;**

int find**(**int x**,**vector**<**pii**>** **&**mem**){**

**if(**x **==** f**[**x**])** **return** x**;**

mem**.**push\_back**(**make\_pair**(**x**,**f**[**x**]));**

f**[**x**]** **=** find**(**f**[**x**],**mem**);**

**return** f**[**x**];**

**}**

void deal**(**int l**,**int r**,**vector**<**jobs**>** **&**q**){**

vector**<**pii**>** mem**,**mem2**;**

**for(**jobs **&**p **:** q**)**

**if(**p**.**l **==** l **&&** p**.**r **==** r **&&** p**.**type **==** 1**){**

int x **=** p**.**x**,**y **=** p**.**y**;**

**if(**find**(**x**,**mem**)** **!=** find**(**y**,**mem**)){**

mem**.**push\_back**(**make\_pair**(**f**[**x**],**f**[**f**[**x**]]));**

mem2**.**push\_back**(**make\_pair**(**f**[**y**],**siz**[**f**[**y**]]));**

siz**[**f**[**y**]]** **+=** siz**[**f**[**x**]];**

f**[**f**[**x**]]** **=** f**[**y**];**

**}**

**}**

**for(**jobs **&**p **:** q**)**

**if(**p**.**l **==** l **&&** p**.**r **==** r **&&** p**.**type **!=** 1**){**

int x **=** p**.**x**,**y **=** p**.**y**;**

**if(**p**.**type **==** 2**)** ans**[**l**]** **=** siz**[**find**(**x**,**mem**)];**

**else** ans**[**l**]** **=** N **+** **(**find**(**x**,**mem**)** **==** find**(**y**,**mem**));**

**}**

**if(**l **!=** r**){**

vector**<**jobs**>** q1**,**q2**;**

int mid **=** **(**l**+**r**)>>**1**;**

**for(**jobs **&**p **:** q**)**

**if(**p**.**r **<=** mid**)** q1**.**push\_back**(**p**);**

**else**

**if(**p**.**l **<=** mid**){**

**if(**p**.**l **==** l **&&** p**.**r **==** r**)** **continue;**

q1**.**push\_back**(**jobs**(**1**,**p**.**x**,**p**.**y**,**p**.**l**,**mid**));**

q2**.**push\_back**(**jobs**(**1**,**p**.**x**,**p**.**y**,**mid**+**1**,**p**.**r**));**

**}**

**else** q2**.**push\_back**(**p**);**

q**.**clear**();**

deal**(**l**,**mid**,**q1**);**

deal**(**mid**+**1**,**r**,**q2**);**

**}**

**for(**int i **=** mem**.**size**()-**1**;**i **>=** 0**;**i**--)** f**[**mem**[**i**].**first**]** **=** mem**[**i**].**second**;**

**for(**int i **=** mem2**.**size**()-**1**;**i **>=** 0**;**i**--)** siz**[**mem2**[**i**].**first**]** **=** mem2**[**i**].**second**;**

**}**

int main**(){**

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

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

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

**if(**op **==** 1**){**

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

**if(**x **<** y**)** swap**(**x**,**y**);**

pre**[**1ll**\***x**\***N**+**y**]** **=** i**;**

**}**

**if(**op **==** 2**){**

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

**if(**x **<** y**)** swap**(**x**,**y**);**

job**.**push\_back**(**jobs**(**1**,**x**,**y**,**pre**[**1ll**\***x**\***N**+**y**],**i**-**1**));**

pre**.**erase**(**pre**.**find**(**1ll**\***x**\***N**+**y**));**

**}**

**if(**op **==** 3**){**

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

job**.**push\_back**(**jobs**(**2**,**x**,**0**,**i**,**i**));**

**}**

**if(**op **==** 4**){**

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

job**.**push\_back**(**jobs**(**3**,**x**,**y**,**i**,**i**));**

**}**

**}**

**for(**pair**<**long long**,**int**>** v **:** pre**)**

job**.**push\_back**(**jobs**(**1**,**v**.**first**/**N**,**v**.**first **%** N**,**v**.**second**,**m**));**

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

f**[**i**]** **=** i**,**siz**[**i**]** **=** 1**;**

deal**(**1**,**m**,**job**);**

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

**if(!**ans**[**i**])** **continue;**

**if(**ans**[**i**]** **<** N**)** cout**<<**ans**[**i**]<<**endl**;**

**else{**

**if(**ans**[**i**]** **==** N**)** cout**<<**"No"**<<**endl**;**

**else** cout**<<**"Yes,cap"**<<**endl**;**

**}**

**}**

**}**

# Geometry

## rotateCalip

/\*旋转卡壳求凸包中最远点对\*/

double Dis**(**Point a**,** Point b**){return** Length**(**a**-**b**);}**

double Area**(**Point a**,** Point b**,** Point c**){return** Cross**(**b**-**a**,**c**-**a**);}**

double rot\_calip**(**Point**\*** p**,** int n**){**

double res **=** 0**;**

**for(**int i **=** 0**,** j **=** 1**;** i **<** n**;** i**++){**

**while(**Area**(**p**[**i**],**p**[**i**+**1**],**p**[**j**+**1**])** **>** Area**(**p**[**i**],**p**[**i**+**1**],**p**[**j**]))** j**=(**j**+**1**)%**n**;**

res **=** max**(**res**,** Dis**(**p**[**i**],** p**[**j**]));**

res **=** max**(**res**,** Dis**(**p**[**i**+**1**],** p**[**j**]));**

**}**

**return** res**;**

**}**

# Graph

## costFlow\_dijkstra

#include <vector>

#include <queue>

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

**typedef** pair**<**int**,** int**>** pii**;**

#define fi first

#define se second

#define pb push\_back

#define mk make\_pair

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

const int maxn **=** 110**;**

const int maxv **=** 10100**;**

int n**,** m**,** ans**;**

struct es

**{**

int to**,** cap**,** rev**,** cost**;**

es**(**int to**,** int cap**,** int rev**,** int cost**):**to**(**to**),**cap**(**cap**),**rev**(**rev**),**cost**(**cost**){}**

**};**

vector**<**es**>** G**[**maxv**];**

bool vis**[**maxv**];**

int dist**[**maxv**];**

void add\_edge**(**int from**,** int to**,** int cap**,** int cost**)**

**{**

G**[**from**].**pb**(**es**(**to**,** cap**,** **(**int**)**G**[**to**].**size**(),** cost**));**

G**[**to**].**pb**(**es**(**from**,** 0**,** **(**int**)**G**[**from**].**size**()-**1**,** **-**cost**));**

**}**

bool dijkstra**(**int s**,** int t**)**

**{**

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

dist**[**t**]** **=** 0**;**

priority\_queue**<**pii**,** vector**<**pii**>,** greater**<**pii**>** **>** pq**;**

pq**.**push**(**mk**(**0**,** t**));**

**while(!**pq**.**empty**()){**

pii u **=** pq**.**top**();** pq**.**pop**();**

**if(**dist**[**u**.**se**]** **<** u**.**fi**)continue;**

**for(**int i **=** 0**;** i **<** **(**int**)**G**[**u**.**se**].**size**();** i**++){**

es **&**e **=** G**[**u**.**se**][**i**];**

**if(**G**[**e**.**to**][**e**.**rev**].**cap **&&** dist**[**e**.**to**]** **>** dist**[**u**.**se**]-**e**.**cost**){**

dist**[**e**.**to**]** **=** dist**[**u**.**se**]-**e**.**cost**;**

pq**.**push**(**mk**(**dist**[**e**.**to**],** e**.**to**));**

**}**

**}**

**}**

**return** dist**[**s**]** **<** INF**;**

**}**

int dfs**(**int v**,** int t**,** int f**)**

**{**

vis**[**v**]** **=** 1**;**

**if(**v **==** t**)return** f**;**

int ret **=** 0**;**

**for(**int i **=** 0**;** i **<** **(**int**)**G**[**v**].**size**();** i**++){**

es **&**e **=** G**[**v**][**i**];**

**if(!**vis**[**e**.**to**]** **&&** e**.**cap **&&** dist**[**v**]** **-** e**.**cost **==** dist**[**e**.**to**]){**

int d **=** dfs**(**e**.**to**,** t**,** min**(**e**.**cap**,** f**-**ret**));**

**if(**d**){**

ans **+=** e**.**cost**\***d**;**

e**.**cap **-=** d**;**

G**[**e**.**to**][**e**.**rev**].**cap **+=** d**;**

ret **+=** d**;**

**if(**ret **==** f**)break;**

**}**

**}**

**}**

**return** ret**;**

**}**

int costflow**(**int s**,** int t**)**

**{**

int flow **=** 0**;**

**while(**dijkstra**(**s**,** t**)){**

vis**[**t**]** **=** 1**;**

**while(**vis**[**t**]){**

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

flow **+=** dfs**(**s**,** t**,** INF**);**

**}**

**}**

**return** flow**;**

**}**

## dijkstra\_heap

#include <queue>

#include <cstring>

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

**typedef** pair**<**int**,** int**>** pii**;**

#define mk make\_pair

#define fi first

#define se second

const int maxn **=** 1e5**+**10**;**

vector**<**pii**>** G**[**maxn**];**

int cost**[**maxn**];**

void dijkstra**(**int s**)**

**{**

memset**(**cost**,** 0x3f**,** **sizeof(**cost**));**

cost**[**s**]** **=** 0**;**

priority\_queue**<**pii**,** vector**<**pii**>,** greater**<**pii**>** **>**pq**;**

pq**.**push**(**mk**(**0**,** s**));**

**while(!**pq**.**empty**()){**

pii p **=** pq**.**top**();** pq**.**pop**();**

**if(**cost**[**p**.**se**]** **<** p**.**fi**)continue;**

**for(**int i **=** 0**;** i **<** **(**int**)**G**[**p**.**se**].**size**();** i**++){**

pii v **=** G**[**p**.**se**][**i**];**

**if(**cost**[**v**.**fi**]** **>** cost**[**p**.**se**]+**v**.**se**){**

cost**[**v**.**fi**]** **=** cost**[**p**.**se**]+**v**.**se**;**

pq**.**push**(**mk**(**cost**[**v**.**fi**],** v**.**fi**));**

**}**

**}**

**}**

**}**

## dinic & isap

#include<bits/stdc++.h>

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

struct edge**{**int st**,**ed**,**fl**,**ne**;}**x**[**maxm**];**

queue**<**int**>**q**;**

int d**[**maxn**],**g**[**maxn**],**bj**[**maxn**],**now**[**maxn**];**

/\*Dinic\*/

bool bfs**(){**

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

q**.**push**(**1**);**d**[**1**]=**1**;**

**while(!**q**.**empty**()){**

int t**=**q**.**front**();**q**.**pop**();**

**for(**int i**=**bj**[**t**];**i**!=-**1**;**i**=**x**[**i**].**ne**)**

**if(!**d**[**x**[**i**].**ed**]&&**x**[**i**].**fl**)**

q**.**push**(**x**[**i**].**ed**),**d**[**x**[**i**].**ed**]=**d**[**t**]+**1**;**

**}**

**return** d**[**n**]!=**0**;**

**}**

int dfs**(**int t**,**int pre**){**

**if(**t**==**n**)return** pre**;**

**for(**int **&**i**=**now**[**t**];**i**!=-**1**;**i**=**x**[**i**].**ne**){**

**if(!**x**[**i**].**fl**||**d**[**x**[**i**].**ed**]!=**d**[**t**]+**1**||!(**k**=**dfs**(**x**[**i**].**ed**,**min**(**x**[**i**].**fl**,**pre**))))continue;**

x**[**i**].**fl**-=**k**,**x**[**i**^**1**].**fl**+=**k**;**

**return** k**;**

**}**

**return** 0**;**

**}**

int main**(){**

**...**

**while(**bfs**()){**

**for(**int i**=**1**;**i**<=**n**;**i**++)**now**[**i**]=**bj**[**i**];**

**do** ans**+=(**t**=**dfs**(**1**,**inf**));**

**while(**t**!=**0**);**

**}**

**return** 0**;**

**}**

/\*ISAP

int dfs(int t,int pre){

if(t==n)return pre;

int mn=n-1,res=pre,k;

for(int i=bj[t];i!=-1;i=x[i].ne){

if(!x[i].fl)continue;

if(d[t]==d[x[i].ed]+1){

k=dfs(x[i].ed,min(res,x[i].fl));

x[i].fl-=k;

x[i^1].fl+=k;

res-=k;

if(d[1]>=n)return pre-res;

if(!res)break;

}

mn=min(mn,d[x[i].ed]);

}

if(res==pre){

if(!(--g[d[t]]))d[1]=n;

g[d[t]=mn+1]++;

}

return pre-res;

}

int main(){

...

g[0]=n;

while(d[1]<n)ans-=dfs(1,inf);

return 0;

}

\*/

## dominatorTree

#include <cstdio>

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

**typedef** long long ll**;**

const int MaxN **=** 100000 **+** 10**,** MaxE **=** **(**5 **\*** 100000**)** **\*** 2 **+** MaxN**;**

int head**[**MaxN**],** pre**[**MaxN**],** dom**[**MaxN**],** to**[**MaxE**],** nxt**[**MaxE**],** top**;**

int bcj**[**MaxN**],** semi**[**MaxN**],** idom**[**MaxN**],** best**[**MaxN**],** dfn**[**MaxN**],** id**[**MaxN**],** fa**[**MaxN**],** dfs\_clock**;**

int sons**[**MaxN**];**

int n**,** m**;**

ll ans**;**

void addedge**(**int **\***h**,**int fr**,**int tt**)**

**{**

top **++;**

nxt**[**top**]** **=** h**[**fr**];**

to**[**top**]** **=** tt**;**

h**[**fr**]** **=** top**;**

**}**

void init**()**

**{**

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

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

**for(**int i **=** 1**;** i **<=** m**;** **++**i**)**

**{**

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

addedge**(**head**,** a**,** b**);**

addedge**(**pre**,** b**,** a**);**

**}**

**}**

int push**(**int v**)**

**{**

**if(**v **==** bcj**[**v**])** **return** v**;**

int y **=** push**(**bcj**[**v**]);**

**if(**dfn**[**semi**[**best**[**bcj**[**v**]]]]** **<** dfn**[**semi**[**best**[**v**]]])** best**[**v**]** **=** best**[**bcj**[**v**]];**

**return** bcj**[**v**]** **=** y**;**

**}**//带权并查集路径压缩

void dfs**(**int rt**)**

**{**

dfn**[**rt**]** **=** **++**dfs\_clock**;**

id**[**dfs\_clock**]** **=** rt**;**

**for(**int i **=** head**[**rt**];** i**;** i **=** nxt**[**i**])**

**if(!**dfn**[**to**[**i**]])**

**{**

dfs**(**to**[**i**]);**

fa**[**to**[**i**]]** **=** rt**;**

**}**

**}**//求出dfs序

void tarjan**()**

**{**

**for(**int i **=** dfs\_clock**,** u**;** i **>=** 2**;** **--**i**)**

**{**//按dfs序从大到小计算

u **=** id**[**i**];**

**for(**int j **=** pre**[**u**];** j**;** j **=** nxt**[**j**])**//semi

**{**

**if(!**dfn**[**to**[**j**]])** **continue;**

push**(**to**[**j**]);**

**if(**dfn**[**semi**[**best**[**to**[**j**]]]]** **<** dfn**[**semi**[**u**]])** semi**[**u**]** **=** semi**[**best**[**to**[**j**]]];**

**}**

addedge**(**dom**,** semi**[**u**],** u**);**

bcj**[**u**]** **=** fa**[**u**];**u **=** id**[**i **-** 1**];**

**for(**int j **=** dom**[**u**];** j**;** j **=** nxt**[**j**])**//idom

**{**

push**(**to**[**j**]);**

**if(**semi**[**best**[**to**[**j**]]]** **==** u**)** idom**[**to**[**j**]]** **=** u**;**

**else** idom**[**to**[**j**]]** **=** best**[**to**[**j**]];**

**}**

**}**

**for(**int i **=** 2**,** u**;** i **<=** dfs\_clock**;** **++**i**)**

**{**

u **=** id**[**i**];**

**if(**idom**[**u**]** **!=** semi**[**u**])** idom**[**u**]** **=** idom**[**idom**[**u**]];**

**}**

**}**

void calc\_son**()**

**{**

**for(**int i **=** dfs\_clock**,** u**;** i **>=** 2**;** **--**i**)**

**{**

u **=** id**[**i**];**

**++** sons**[**u**];**

**if(**idom**[**u**]** **!=** 1**)** sons**[**idom**[**u**]]** **+=** sons**[**u**];**

**else** ans **-=** **((**ll**)**sons**[**u**]** **\*** **(**ll**)(**sons**[**u**]** **-** 1**))** **/** 2ll**;**

**}**

**}**

void solve**()**

**{**

**for(**int i **=** 1**;** i **<=** n**;** **++**i**)** bcj**[**i**]** **=** semi**[**i**]** **=** best**[**i**]** **=** i**;**

dfs\_clock **=** 0**;**

dfs**(**1**);**

tarjan**();**

ans **=** **((**ll**)**dfs\_clock **\*** **(**ll**)(**dfs\_clock **-** 1**))** **/** 2ll**;**

calc\_son**();**

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

**}**

## euler

#include <stack>

#include <vector>

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

#define pb push\_back

int n**,** m**;**

const int maxn **=** 1e5**+**10**;**

vector**<**int**>** G**[**maxn**];**

stack**<**int**>** sk**;**

vector**<**int**>** ans**;**

void euler**(**int st**)**//调用后会修改原图

**{**

sk**.**push**(**st**);**

**while(!**sk**.**empty**()){**

int u **=** sk**.**top**();**

bool flg **=** 0**;**

**for(**int i **=** 0**;** i **<** **(**int**)**G**[**u**].**size**();** i**++){**

int v **=** G**[**u**][**i**];**

**if(**v **!=** **-**1**){**

flg **=** 1**;**

G**[**u**][**i**]** **=** **-**1**;**

sk**.**push**(**v**);**

**break;**

**}**

**}**

**if(!**flg**){**

ans**.**pb**(**u**);**

sk**.**pop**();**

**}**

**}**

**}**

## floyd

#include <algorithm>

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

const int maxn **=** 1e3**+**10**;**

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

int mp**[**maxn**][**maxn**];**//原始距离

int dis**[**maxn**][**maxn**];**//最短路

int n**;**

int floyd**()**

**{**

int ret **=** INF**;**

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

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

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

ret **=** min**(**ret**,** dis**[**i**][**j**]+**mp**[**i**][**k**]+**mp**[**k**][**j**]);**

**}**

**}**

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

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

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

**}**

**}**

**}**

**return** ret**;**

**}**

## hungarian

#include <vector>

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

const int maxn **=** 1e3**+**10**;**

vector**<**int**>** G**[**maxn**];**

int mat**[**maxn**];**

bool vis**[**maxn**];**

int n**;**

bool dfs**(**int x**){**

**for** **(**int i **=** 0**;** i **<** **(**int**)**G**[**x**].**size**();** i**++){**

int v **=** G**[**x**][**i**];**

**if(!**vis**[**v**]){**

vis**[**v**]** **=** 1**;**

**if** **(**mat**[**v**]** **==** 0 **||** dfs**(**mat**[**v**])){**

mat**[**v**]** **=** x**;**

**return** **true;**

**}**

**}**

**}**

**return** **false;**

**}**

int solve**()**

**{**

int ans **=** 0**;**

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

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

**if(**dfs**(**i**))**ans**++;**

**}**

**return** ans**;**

**}**

## kosaraju

#include <vector>

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

const int maxn **=** 1e5**+**10**;**

vector**<**int**>** G**[**maxn**],** G2**[**maxn**];**

vector**<**int**>** S**;**

int sccno**[**maxn**],** scc\_cnt**,** n**,** m**;**

bool vis**[**maxn**];**

void dfs1**(**int u**)**

**{**

**if(**vis**[**u**])return;**

vis**[**u**]** **=** 1**;**

**for(**int i **=** 0**;** i **<** **(**int**)**G**[**u**].**size**();** i**++)**dfs1**(**G**[**u**][**i**]);**

S**.**push\_back**(**u**);**

**}**

void dfs2**(**int u**)**

**{**

**if(**sccno**[**u**])return;**

sccno**[**u**]** **=** scc\_cnt**;**

**for(**int i **=** 0**;** i **<** **(**int**)**G2**[**u**].**size**();** i**++)**dfs2**(**G2**[**u**][**i**]);**

**}**

void Korasaju**()**

**{**

scc\_cnt **=** 0**;**

S**.**clear**();**

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

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

**for(**int i **=** 0**;** i **<** n**;** i**++)**dfs1**(**i**);**

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

**if(!**sccno**[**S**[**i**]]){**

**++**scc\_cnt**;**

dfs2**(**S**[**i**]);**

**}**

**}**

**}**

## kruskal

#include <algorithm>

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

const int maxn **=** 1e5**+**10**;**

struct edge

**{**

int u**,** v**,** w**;**

bool **operator** **<** **(**const edge **&**rhs**)**const

**{**

**return** w **<** rhs**.**w**;**

**}**

**}**es**[**maxn**];**

int par**[**maxn**];**

int n**,** m**;**

void init**()**

**{**

**for(**int i **=** 0**;** i **<=** n**;** i**++)**par**[**i**]** **=** i**;**

**}**

int find**(**int x**)**

**{**

**return** par**[**x**]==**x**?**x**:**par**[**x**]=**find**(**par**[**x**]);**

**}**

bool unite**(**int x**,** int y**)**

**{**

x **=** find**(**x**),** y **=** find**(**y**);**

**if(**x **!=** y**){**

par**[**x**]** **=** y**;**

**return** 1**;**

**}**

**return** 0**;**

**}**

int kruskal**()**

**{**

sort**(**es**,** es**+**m**);**

int ret **=** 0**;**

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

**if(**unite**(**es**[**i**].**u**,** es**[**i**].**v**))**ret **+=** es**[**i**].**w**;**

**}**

**return** ret**;**

**}**

## lca\_doubling

#include <algorithm>

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

const int maxn **=** 1e6**+**10**;**

const int maxd **=** 21**;**

int anc**[**maxn**][**maxd**];**//anc[i][0]->par[i]

int dep**[**maxn**];**

int n**;**

void init\_lca**()**

**{**

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

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

anc**[**j**][**i**]** **=** anc**[**anc**[**j**][**i**-**1**]][**i**-**1**];**

**}**

**}**

**}**

int lca**(**int u**,** int v**)**

**{**

**if(**dep**[**u**]** **<** dep**[**v**])**swap**(**u**,** v**);**

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

**if(**dep**[**anc**[**u**][**i**]]** **>=** dep**[**v**])**u **=** anc**[**u**][**i**];**

**}**

**if(**u **==** v**)return** u**;**

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

**if(**anc**[**u**][**i**]** **!=** anc**[**v**][**i**])**u **=** anc**[**u**][**i**],** v **=** anc**[**v**][**i**];**

**}**

**return** anc**[**u**][**0**];**

**}**

## matrixTree

#include <vector>

using namespace std;

typedef long long ll;

#define ABS(x) ((x)>=0?(x):(-(x)))

const int maxn = 110;

vector<int> G[maxn];

int n;

ll det()

{

ll ret = 1;

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

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

while(G[j][i]){

ll t = G[i][i]/G[j][i];

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

G[i][k] -= G[j][k]\*t;

}

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

swap(G[i][k], G[j][k]);

}

ret = -ret;

}

}

if(!G[i][i])return 0;

ret \*= G[i][i];

}

return ABS(ret);

}

## maximumDensitySubgraph

#include <cstring>

#include <queue>

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

**typedef** pair**<**int**,** int**>** pii**;**

#define mk make\_pair

#define pb push\_back

#define fi first

#define se second

const double eps **=** 1e-10**;**

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

const int maxn **=** 1e4**+**10**;**

const int maxm **=** 1e4**+**10**;**

struct edge

**{**

int to**,** nxt**;**

long double cap**;**

**}**es**[**maxm**];**

int head**[**maxn**],** iter**[**maxn**],** lev**[**maxn**];**

pii p**[**maxm**];**

bool vis**[**maxn**];**

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

void init**()**

**{**

cnt **=** 0**;**

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

**}**

void add\_edge**(**int u**,** int v**,** long double cap**)**

**{**

es**[**cnt**].**to **=** v**,** es**[**cnt**].**cap **=** cap**,** es**[**cnt**].**nxt **=** head**[**u**];**

head**[**u**]** **=** cnt**++;**

es**[**cnt**].**to **=** u**,** es**[**cnt**].**cap **=** 0**,** es**[**cnt**].**nxt **=** head**[**v**];**

head**[**v**]** **=** cnt**++;**

**}**

bool bfs**(**int s**,** int t**)**

**{**

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

lev**[**s**]** **=** 1**;**

queue**<**int**>** q**;**

q**.**push**(**s**);**

**while(!**q**.**empty**()){**

int u **=** q**.**front**();** q**.**pop**();**

**for(**int i **=** head**[**u**];** **~**i**;** i **=** es**[**i**].**nxt**){**

edge **&**e **=** es**[**i**];**

**if(**e**.**cap **>** eps **&&** lev**[**e**.**to**]** **<** 0**){**

lev**[**e**.**to**]** **=** lev**[**u**]+**1**;**

**if(**e**.**to **==** t**)return** 1**;**

q**.**push**(**e**.**to**);**

**}**

**}**

**}**

**return** 0**;**

**}**

long double dfs**(**int v**,** int t**,** long double f**)**

**{**

**if(**v **==** t**)return** f**;**

long double ret **=** 0**;**

**for(**int **&**i **=** iter**[**v**];** **~**i**;** i **=** es**[**i**].**nxt**){**

edge **&**e **=** es**[**i**];**

**if(**e**.**cap **>** eps **&&** lev**[**e**.**to**]** **==** lev**[**v**]+**1**){**

long double d **=** dfs**(**e**.**to**,** t**,** min**(**e**.**cap**,** f**-**ret**));**

**if(**d **>** eps**){**

e**.**cap **-=** d**;**

es**[**i**^**1**].**cap **+=** d**;**

ret **+=** d**;**

**if(**ret**+**eps **>=** f**)break;**

**}**

**}**

**}**

**if(**ret **<** eps**)**lev**[**v**]** **=** 0**;**

**return** ret**;**

**}**

long double max\_flow**(**int s**,** int t**)**

**{**

long double ret **=** 0**;**

**while(**bfs**(**s**,** t**)){**

memcpy**(**iter**,** head**,** **sizeof(**head**));**

ret **+=** dfs**(**s**,** t**,** INF**);**

**}**

**return** ret**;**

**}**

bool maximumDensitySubgraph**(**long double m**)**

**{**

init**();**

int src **=** N**+**M**+**1**,** dst **=** src**+**1**;**

long double sum **=** 0**;**

**for(**int i **=** 1**;** i **<=** N**;** i**++)**add\_edge**(**i**,** dst**,** m**);**

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

add\_edge**(**src**,** i**+**N**,** 1**),** sum **+=** 1**;**

add\_edge**(**i**+**N**,** p**[**i**].**fi**,** INF**);**

add\_edge**(**i**+**N**,** p**[**i**].**se**,** INF**);**

**}**

long double ret **=** sum**-**max\_flow**(**src**,** dst**);**

**return** ret **>** eps**;**

**}**

void dfs**(**int u**)**

**{**

vis**[**u**]** **=** 1**;**

**for(**int i **=** head**[**u**];** **~**i**;** i **=** es**[**i**].**nxt**){**

edge **&**e **=** es**[**i**];**

**if(!**vis**[**e**.**to**]** **&&** e**.**cap **>** eps**)**dfs**(**e**.**to**);**

**}**

**}**

void solve**()**

**{**

scanf**(**"%d%d"**,** **&**N**,** **&**M**);**

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

int u**,** v**;**

scanf**(**"%d%d"**,** **&**u**,** **&**v**);**

p**[**i**]** **=** mk**(**u**,** v**);**

**}**

long double l **=** 0**,** r **=** M**,** ans **=** 0**;**

**while(**l**+**1.0**/(**N**\***N**)** **<=** r**){**

long double m **=** **(**l**+**r**)/**2**;**

**if(**maximumDensitySubgraph**(**m**)){**

ans **=** m**;**

l **=** m**;**

**}**

**else** r **=** m**;**

**}**

maximumDensitySubgraph**(**ans**);**

int src **=** N**+**M**+**1**;**

dfs**(**src**);**

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

**for(**int i **=** 1**;** i **<=** N**;** i**++)if(**vis**[**i**])**vec**.**pb**(**i**);**

**if(**vec**.**empty**())**vec**.**pb**(**1**);**

printf**(**"%d\n"**,** **(**int**)**vec**.**size**());**

**for(**int i **=** 0**;** i **<** **(**int**)**vec**.**size**();** i**++)**printf**(**"%d\n"**,** vec**[**i**]);**

**}**

## maximumWeightClosureGraph

#include <cstdio>

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

**typedef** long long ll**;**

const int maxn **=** 5e3**+**10**;**

const int maxm **=** 2e5**+**10**;**

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

struct edge

**{**

int to**,** cap**,** nxt**;**

**}**es**[**maxm**];**

int head**[**maxn**];**

bool vis**[**maxn**];**

int cnt**;**

void add\_edge**(**int u**,** int v**,** int cap**)**

**{**

es**[**cnt**].**to **=** v**,** es**[**cnt**].**cap **=** cap**,** es**[**cnt**].**nxt **=** head**[**u**];**

head**[**u**]** **=** cnt**++;**

es**[**cnt**].**to **=** u**,** es**[**cnt**].**cap **=** 0**,** es**[**cnt**].**nxt **=** head**[**v**];**

head**[**v**]** **=** cnt**++;**

**}**

void dfs**(**int u**)**

**{**

vis**[**u**]** **=** 1**;**

**for(**int i **=** head**[**u**];** **~**i**;** i **=** es**[**i**].**nxt**){**

edge e **=** es**[**i**];**

**if(**e**.**cap **&&** **!**vis**[**e**.**to**])**dfs**(**e**.**to**);**

**}**

**}**

int solve**()**

**{**

int n**,** m**;**

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

int src **=** n**+**1**,** dst **=** src**+**1**;**

ll sum **=** 0**;**

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

int w**;** scanf**(**"%d"**,** **&**w**);**

**if(**w **>** 0**)**add\_edge**(**src**,** i**,** w**),** sum **+=** w**;**

**else** **if(**w **<** 0**)**add\_edge**(**i**,** dst**,** **-**w**);**

**}**

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

int u**,** v**;**

scanf**(**"%d%d"**,** **&**u**,** **&**v**);**

add\_edge**(**u**,** v**,** INF**);**

**}**

ll ans **=** sum**-**max\_flow**(**src**,** dst**);**

dfs**(**src**);**

int num **=** 0**;**

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

**if(**vis**[**i**])**num**++;**

**}**

**return** ans**;**

**}**

## mixedEuler

#include <stack>

#include <vector>

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

**typedef** pair**<**int**,** int**>** pii**;**

#define pb push\_back

#define mk make\_pair

#define fi first

#define se second

const int maxn **=** 1e3**+**10**;**

const int maxm **=** 2e4**+**10**;**

struct edge

**{**

int to**,** cap**,** nxt**;**

**}**es**[**maxm**];**

int head**[**maxn**],** iter**[**maxn**],** lev**[**maxn**];**

int deg**[**maxn**];**

vector**<**int**>** G**[**maxn**];**

stack**<**int**>** S**;**

pii p**[**maxm**];**

int n**,** m**,** cnt**;**

void add\_edge**(**int u**,** int v**,** int cap**)**

**{**

es**[**cnt**].**to **=** v**,** es**[**cnt**].**cap **=** cap**,** es**[**cnt**].**nxt **=** head**[**u**];**

head**[**u**]** **=** cnt**++;**

es**[**cnt**].**to **=** u**,** es**[**cnt**].**cap **=** 0**,** es**[**cnt**].**nxt **=** head**[**v**];**

head**[**v**]** **=** cnt**++;**

**}**

bool solve**()**

**{**

int num **=** 0**;**

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

int u**,** v**,** type**;**

scanf**(**"%d%d%d"**,** **&**u**,** **&**v**,** **&**type**);**

deg**[**u**]++,** deg**[**v**]--;**

**if(**type **==** 1**)**p**[**num**++]** **=** mk**(**u**,** v**);**//undirected

**else** G**[**u**].**pb**(**v**);**//directed

**}**

bool flg **=** 1**;**

int src **=** n**+**1**,** dst **=** src**+**1**;**

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

**if(**deg**[**i**]%**2 **!=** 0**){**

flg **=** 0**;**

**break;**

**}**

**if(**deg**[**i**]** **>** 0**)**add\_edge**(**src**,** i**,** deg**[**i**]/**2**);**

**else** **if(**deg**[**i**]** **<** 0**)**add\_edge**(**i**,** dst**,** **-**deg**[**i**]/**2**);**

**}**

**if(!**flg**)return** 0**;**//没有合法欧拉路

**for(**int i **=** 0**;** i **<** num**;** i**++)**add\_edge**(**p**[**i**].**fi**,** p**[**i**].**se**,** 1**);**

max\_flow**(**src**,** dst**);**

**for(**int i **=** head**[**src**];** **~**i**;** i **=** es**[**i**].**nxt**){**

**if(**es**[**i**].**cap**){**

flg **=** 0**;**

**break;**

**}**

**}**

**if(!**flg**)return** 0**;**

**else{**

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

**for(**int j **=** head**[**i**];** **~**j**;** j **=** es**[**j**].**nxt**){**

int u **=** i**,** v **=** es**[**j**].**to**;**

**if(**v **<=** n **&&** es**[**j**].**cap**)**G**[**u**].**pb**(**v**);**

**}**

**}**

euler**();**

**}**

**return** 1**;**

**}**

## mstCounting

#include <vector>

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

**typedef** long long ll**;**

#define pb push\_back

const int maxn **=** 110**;**

const int maxe **=** 1010**;**

struct edge

**{**

int u**,** v**,** w**;**

bool **operator** **<** **(**const edge **&**rbs**)**const

**{**

**return** w**<**rbs**.**w**;**

**}**

**}**es**[**maxe**];**

ll mat**[**maxn**][**maxn**];**

int link**[**maxn**][**maxn**];**

int fa**[**maxn**];**

int par**[**maxn**];**

bool vis**[**maxn**];**

vector**<**int**>** vec**[**maxn**];**

int n**,** m**;**

ll mod**;**

void init**()**

**{**

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

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

**for(**int i **=** 0**;** i **<=** n**;** i**++)**fa**[**i**]** **=** i**,** par**[**i**]** **=** i**;**

**}**

int find**(**int x**,** int arr**[])**

**{**

**return** x**==**arr**[**x**]?**x**:**arr**[**x**]=**find**(**arr**[**x**],** arr**);**

**}**

ll det**(**int sz**)**

**{**

ll ret **=** 1**;**

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

**for(**int j **=** 0**;** j **<** sz**;** j**++)**mat**[**i**][**j**]** **=** **(**mat**[**i**][**j**]%**mod**+**mod**)%**mod**;**

**}**

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

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

**while(**mat**[**j**][**i**]){**

ll t **=** mat**[**i**][**i**]/**mat**[**j**][**i**];**

**for(**int k **=** i**;** k **<** sz**;** k**++){**

mat**[**i**][**k**]** **=** **(**mat**[**i**][**k**]-**mat**[**j**][**k**]\***t**)%**mod**;**

swap**(**mat**[**i**][**k**],** mat**[**j**][**k**]);**

**}**

ret **=** **(-**ret**+**mod**)%**mod**;**

**}**

**}**

**if(!**mat**[**i**][**i**])return** 0**;**

ret **=** ret**\***mat**[**i**][**i**]%**mod**;**

**}**

**return** **(**ret**+**mod**)%**mod**;**

**}**

ll solve**()**

**{**

sort**(**es**,** es**+**m**);**

ll ret **=** 1**;**

int pre **=** **-**1**;**

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

**if(**es**[**i**].**w **!=** pre **||** i **==** m**){**

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

**if(**vis**[**j**]){**

int fj **=** find**(**j**,** par**);**

vec**[**fj**].**pb**(**j**);**

fa**[**j**]** **=** fj**;**

vis**[**j**]** **=** 0**;**

**}**

**}**

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

int sz **=** vec**[**j**].**size**();**

**if(**sz **<=** 1**)continue;**

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

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

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

int u **=** vec**[**j**][**k**],** v **=** vec**[**j**][**h**];**

mat**[**k**][**h**]** **=** mat**[**h**][**k**]** **=** **-**link**[**u**][**v**];**

mat**[**k**][**k**]** **+=** link**[**u**][**v**];**

mat**[**h**][**h**]** **+=** link**[**v**][**u**];**

**}**

**}**

ret **=** ret**\***det**(**sz**)%**mod**;**

**}**

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

vec**[**j**].**clear**();**

**}**

**if(**i **==** m**)break;**

**}**

int u **=** es**[**i**].**u**,** v **=** es**[**i**].**v**;**

u **=** find**(**u**,** fa**);** v **=** find**(**v**,** fa**);**

**if(**u **==** v**)continue;**

vis**[**u**]** **=** vis**[**v**]** **=** 1**;**

par**[**find**(**v**,** par**)]** **=** find**(**u**,** par**);**

link**[**u**][**v**]++;** link**[**v**][**u**]++;**

pre **=** es**[**i**].**w**;**

**}**

int flg **=** find**(**0**,** fa**);**

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

**if(**find**(**i**,** fa**)** **!=** flg**)return** 0**;**

**}**

**return** **(**ret**+**mod**)%**mod**;**

**}**

## pnt\_d&c

const int maxn **=** 1e4**+**10**;**

int n**,** head**[**maxn**],** val**[**maxn**\***2**],** nxt**[**maxn**\***2**],**

to**[**maxn**\***2**],** tot**;**

void add\_edg**(**int u**,** int v**,** int w**){**

tot**++;** nxt**[**tot**]** **=** head**[**u**];** to**[**tot**]** **=** v**;**

val**[**tot**]** **=** w**;** head**[**u**]** **=** tot**;**

**}**

int root**,** mx**[**maxn**],** sz**[**maxn**],** done**[**maxn**];**

void get\_root**(**int u**,** int f**,** int tree\_size**){**

mx**[**u**]** **=** 0**;** sz**[**u**]** **=** 1**;**

**for(**int i **=** head**[**u**];** i**;** i **=** nxt**[**i**]){**

int v **=** to**[**i**];**

**if(**v **==** f **||** done**[**v**])** **continue;**

get\_root**(**v**,** u**,** tree\_size**);**

sz**[**u**]** **+=** sz**[**v**];**

mx**[**u**]** **=** max**(**mx**[**u**],** sz**[**v**]);**

**}**

mx**[**u**]** **=** max**(**mx**[**u**],** tree\_size**-**sz**[**u**]);**

**if(**mx**[**u**]** **<** mx**[**root**])** root **=** u**;**

**}**

void solve**(**int u**,** int tree\_size**){**

root **=** 0**;**

mx**[**root**]** **=** n**;**

get\_root**(**u**,** 0**,** tree\_size**);**

u **=** root**;**

done**[**u**]** **=** 1**;**

/\*\*\*\*\*\*\*\*\*/

///TODO

/\*\*\*\*\*\*\*\*\*/

**for(**int i **=** head**[**u**];** i**;** i **=** nxt**[**i**]){**

int v **=** to**[**i**];**

**if(**done**[**v**])** **continue;**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

///TODO

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

solve**(**v**,** sz**[**v**]);**

**}**

**}**

void init**(){**

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

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

tot **=** 0**;**

///TODO

**}**

## prim\_heap

#include <cstring>

#include <queue>

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

**typedef** pair**<**int**,** int**>** pii**;**

#define fi first

#define se second

#define mk make\_pair

const int maxn **=** 1e6**+**10**;**

vector**<**pii**>** G**[**maxn**];**

int cost**[**maxn**];**

bool vis**[**maxn**];**

int prim**()**

**{**

priority\_queue**<**pii**,** vector**<**pii**>,** greater**<**pii**>** **>** pq**;**

memset**(**cost**,** 0x3f**,** **sizeof(**cost**));**

cost**[**1**]** **=** 0**;**

pq**.**push**(**mk**(**0**,** 1**));**

int ret **=** 0**;**

**while(!**pq**.**empty**()){**

pii p **=** pq**.**top**();** pq**.**pop**();**

**if(**vis**[**p**.**se**])continue;**

vis**[**p**.**se**]** **=** 1**;**

ret **+=** cost**[**p**.**se**];**

**for(**int i **=** 0**;** i **<** **(**int**)**G**[**p**.**se**].**size**();** i**++){**

pii v **=** G**[**p**.**se**][**i**];**

**if(**cost**[**v**.**fi**]** **>** v**.**se**){**

cost**[**v**.**fi**]** **=** v**.**se**;**

pq**.**push**(**mk**(**cost**[**v**.**fi**],** v**.**fi**));**

**}**

**}**

**}**

**return** ret**;**

**}**

## tarjan

#include <algorithm>

#include <vector>

#include <stack>

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

const int maxn **=** 1e5**+**10**;**

vector**<**int**>** G**[**maxn**];**

stack**<**int**>** S**;**

int low**[**maxn**],** dfn**[**maxn**],** scc**[**maxn**];**

int n**,** cnt**,** idx**;**

void dfs**(**int u**,** int f**)**

**{**

low**[**u**]** **=** dfn**[**u**]** **=** **++**idx**;**

S**.**push**(**u**);**

**for(**int i **=** 0**;** i **<** **(**int**)**G**[**u**].**size**();** i**++){**

int v **=** G**[**u**][**i**];**

**if(**v **==** f**)continue;**

**if(!**dfn**[**v**]){**

dfs**(**v**,** u**);**

low**[**u**]** **=** min**(**low**[**u**],** low**[**v**]);**

**}**

**else** **if(!**scc**[**v**]){**

low**[**u**]** **=** min**(**low**[**u**],** dfn**[**v**]);**

**}**

**}**

**if(**low**[**u**]** **==** dfn**[**u**]){**

cnt**++;**

**while(**1**){**

int x **=** S**.**top**();** S**.**pop**();**

scc**[**x**]** **=** cnt**;**

**if(**x **==** u**)break;**

**}**

**}**

**}**

void tarjan**()**

**{**

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

**if(!**dfn**[**i**])**dfs**(**i**,** 0**);**

**}**

**}**

## threeMemberedRing

#include <vector>

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

#define pb push\_back

const int maxn **=** 2e5**+**10**;**

vector**<**int**>** G**[**maxn**];**

int deg**[**maxn**];**

struct edge

**{**

int u**,** v**;**

**}**es**[**maxn**];**

int vis**[**maxn**];**

void init**(**int n**)**

**{**

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

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

**for(**int i **=** 0**;** i **<=** n**;** i**++)**G**[**i**].**clear**();**

**}**

int Counting**()**

**{**

int n**,** m**,** ans **=** 0**;**

**while(**scanf**(**"%d%d"**,** **&**n**,** **&**m**)** **!=** EOF**){**

init**(**n**);**

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

int u**,** v**;**

scanf**(**"%d%d"**,** **&**u**,** **&**v**);**

es**[**i**].**u **=** u**,** es**[**i**].**v **=** v**;**

deg**[**u**]++,** deg**[**v**]++;**

**}**

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

int u **=** es**[**i**].**u**,** v **=** es**[**i**].**v**;**

**if(**deg**[**u**]** **<** deg**[**v**]** **||** **(**deg**[**u**]** **==** deg**[**v**]** **&&** u **<** v**))**G**[**u**].**pb**(**v**);**

**else** G**[**v**].**pb**(**u**);**

**}**

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

int u **=** es**[**i**].**u**,** v **=** es**[**i**].**v**;**

**for(**auto w **:** G**[**u**])**vis**[**w**]** **=** i**;**

**for(**auto w **:** G**[**v**])if(**vis**[**w**]** **==** i**)**ans**++;**

**}**

**}**

**return** ans**;**

**}**

## zhuliu

//O(nm)

#include <cstring>

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

const int maxn **=** 1010**;**

const int maxe **=** 40100**;**

struct edge

**{**

int u**,** v**,** w**;**

**}**es**[**maxe**];**

int vis**[**maxn**],** id**[**maxn**],** pre**[**maxn**],** in**[**maxn**];**

int n**,** m**;**

int zhuliu**(**int root**)**

**{**

int ret **=** 0**;**

**while(**1**){**

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

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

memset**(**in**,** 0x3f**,** **sizeof(**in**));**

**for(**int i **=** 0**;** i **<** m**;** i**++)if(**es**[**i**].**u **!=** es**[**i**].**v **&&** in**[**es**[**i**].**v**]** **>** es**[**i**].**w**){**

in**[**es**[**i**].**v**]** **=** es**[**i**].**w**;**

pre**[**es**[**i**].**v**]** **=** es**[**i**].**u**;**

**}**

**for(**int i **=** 0**;** i **<** n**;** i**++)if(**i **!=** root **&&** in**[**i**]** **==** INF**)return** INF**;**//不联通

in**[**root**]** **=** 0**;**

int loop **=** 0**;**

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

ret **+=** in**[**i**];**

int v **=** i**;**

**while(**v **!=** root **&&** id**[**v**]** **==** **-**1 **&&** vis**[**v**]** **!=** i**){**

vis**[**v**]** **=** i**;**

v **=** pre**[**v**];**

**}**

**if(**vis**[**v**]** **==** i**){**

**for(**int j **=** pre**[**v**];** j **!=** v**;** j **=** pre**[**j**])**id**[**j**]** **=** loop**;**

id**[**v**]** **=** loop**++;**

**}**

**}**

**if(!**loop**)break;**

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

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

int v **=** es**[**i**].**v**;**

es**[**i**].**u **=** id**[**es**[**i**].**u**];**

es**[**i**].**v **=** id**[**es**[**i**].**v**];**

**if(**es**[**i**].**u **!=** es**[**i**].**v**)**es**[**i**].**w **-=** in**[**v**];**

**}**

root **=** id**[**root**];**

n **=** loop**;**

**}**

**return** ret**;**

**}**

# Math

## combination\_pre

///简单递推

int c**[**maxn**][**maxn**];**

void init**(){**

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

c**[**0**][**0**]** **=** c**[**1**][**0**]** **=** c**[**1**][**1**]** **=** 1**;**

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

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

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

c**[**i**][**j**]** **=** **(**c**[**i**-**1**][**j**]** **+** c**[**i**-**1**][**j**-**1**])%**mod**;**

**}**

**}**

///预处理阶乘+逆元

const int maxn **=** 1e6**+**10**;**

ll fac**[**maxn**],** iv**[**maxn**];**

void extgcd**(**ll aa**,**ll bb**,**ll**&** dd**,**ll**&** xx**,**ll**&** yy**){**

**if(!**bb**){**

dd **=** aa**;**

xx **=** 1**;**

yy **=** 0**;**

**}else{**

extgcd**(**bb**,** aa**%**bb**,** dd**,** yy**,** xx**);**

yy **-=** xx**\*(**aa**/**bb**);**

**}**

**}**

ll inv**(**ll aa**,**ll mm**){**

ll dd**,** xx**,** yy**;**

extgcd**(**aa**,** mm**,** dd**,** xx**,** yy**);**

**return** dd**==**1**?(**xx**+**mm**)%**mm**:-**1**;**

**}**

void init**(){**

fac**[**0**]** **=** 1**;**

iv**[**0**]** **=** inv**(**fac**[**0**],** mod**);**

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

fac**[**i**]** **=** **(**i**\***fac**[**i**-**1**])%**mod**;**

iv**[**i**]** **=** inv**(**fac**[**i**],** mod**);**

**}**

**}**

ll comb**(**int x**,** int y**){**

**if(**x **<** y**)** swap**(**x**,**y**);**

ll res **=** **(**fac**[**x**]\***iv**[**x**-**y**])%**mod**;**

res **=** **(**res**\***iv**[**y**])%**mod**;**

**return** res**;**

**}**

## FFT

struct Complex**{**

double a**,**b**;**

Complex**(**double aa **=** 0**,** double bb **=** 0**){**a **=** aa**,** b **=** bb**;}**

Complex **operator** **+** **(**const Complex**&** rbs**)** const**{**

**return** Complex**(**a**+**rbs**.**a**,** b**+**rbs**.**b**);**

**}**

Complex **operator** **-** **(**const Complex**&** rbs**)** const**{**

**return** Complex**(**a**-**rbs**.**a**,** b**-**rbs**.**b**);**

**}**

Complex **operator** **\*** **(**const Complex**&** rbs**)** const**{**

**return** Complex**(**a**\***rbs**.**a**-**b**\***rbs**.**b**,** a**\***rbs**.**b**+**b**\***rbs**.**a**);**

**}**

**};**

void change**(**Complex y**[],** int len**){**

**for(**int i **=** 1**,** j **=** len**/**2**;** i **<** len**-**1**;** i**++){**

**if(**i **<** j**)** swap**(**y**[**i**],** y**[**j**]);**

int k **=** len**>>**1**;**

**while(**1**){**

j **^=** k**;**

**if(**j**&**k**)** **break;**

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

**}**

**}**

**}**

void fft**(**Complex y**[],** int len**,** int o**){**

change**(**y**,** len**);**

**for(**int h **=** 2**;** h **<=** len**;** h **<<=** 1**){**

Complex wn**(**cos**(-**o**\***2**\***PI**/**h**),** sin**(-**o**\***2**\***PI**/**h**));**

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

Complex w**(**1**,**0**);**

**for(**int j **=** i**;** j **<** i**+**h**/**2**;** j**++){**

Complex l **=** y**[**j**],** r **=** w**\***y**[**j**+**h**/**2**];**

y**[**j**]** **=** l**+**r**;** y**[**j**+**h**/**2**]** **=** l**-**r**;**

w **=** w**\***wn**;**

**}**

**}**

**}**

**if(**o **==** **-**1**)** **for(**int i **=** 0**;** i **<** len**;** i**++)** y**[**i**].**a **/=** len**;**

**}**

## NTT

/\*\*\*\*\*\*

\* NTT Algorithm

\* Time Complexity: O(nlog n)

\* available data range rely on the value of g and modn

\* default range:

\* \* 0 <= ai < 1004535809 (a number slightly bigger than 1e9)

\* \* n<= 2^21

\* prototype problem: HDU1402

\*\*\*\*\*\*/

#include <cstdio>

#include <cstring>

#include <algorithm>

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

**typedef** long long ll**;**

const ll g **=** 3**,** modn **=** **(**479**<<**21**)+**1**;**

const int maxn **=** 1**<<**17**;**

ll a**[**maxn**],** b**[**maxn**];**

char s**[**maxn**];**

ll quickpow**(**ll base**,** ll n**,** const ll **&**P**){**

ll ans **=** 1**;**

**while(**n**){**

**if(**n**&**1**)** ans **=** ans**\***base**%**P**;**

base **=** base**\***base**%**P**;**

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

**}**

**return** ans**;**

**}**

inline ll multi**(**ll x**,**ll y**,**ll P**){**

ll tmp**=(**x**\***y**-(**ll**)((**long double**)**x**/**P**\***y**+**1.0e-8**)\***P**);**

**return** tmp**<**0 **?** tmp**+**P **:** tmp**;**

**}**

inline int nor**(**int x**)** **{** **return** x**<**0**?**x**+**modn**:**x**<**modn**?**x**:**x**-**modn**;** **}**

/// len: must be 2^k and not smaller than length of x

/// note that len of NTT must be equal to len of INTT

void ntt**(**ll x**[],** const int **&**len**,** const int **&**on**){**

**for(**int i**=**1**,** j**=**len**/**2**;** i**<**len**-**1**;** i**++)** **{**

**if(**i**<**j**)** swap**(**x**[**i**],** x**[**j**]);**

int k **=** len**/**2**;**

**while(**j**>=**k**)** **{**

j **-=** k**;**

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

**}**

**if(**j**<**k**)** j **+=** k**;**

**}**

**for(**int k**=**1**;** k**<**len**;** k**<<=**1**)** **{**

// wm[pos] = quickpow(g, modn>>(pos+1), modn);

ll wm **=** quickpow**(**g**,** modn**/**2**/**k**,** modn**);**

**for(**int i**=**0**;** i**<**len**;** i**+=**2**\***k**)** **{**

ll w **=** 1**;**

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

ll u **=** x**[**i**+**j**];**

ll t **=** w**\***x**[**i**+**j**+**k**]%**modn**;**

x**[**i**+**j**]** **=** nor**(**u**+**t**);**

x**[**i**+**j**+**k**]** **=** nor**(**u**-**t**);**

w **=** w**\***wm**%**modn**;**

**}**

**}**

**}**

**if(**on **==** **-**1**)** **{**

reverse**(**x**+**1**,** x**+**len**);**

long long inv **=** quickpow**(**len**,** modn**-**2**,** modn**);**

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

x**[**i**]** **=** x**[**i**]\***inv**%**modn**;**

**}**

**}**

/\*\*

prime number:

g r k modn

3 1 16 1<<16|1 = 65537

3 7 26 7<<26|1 = 469762049

3 119 23 119<<23|1 = 998244353

3 479 21 479<<21|1 = 1004535809

31 15 27 15<<27|1 = 2013265921

3 17 27 17<<27|1 = 2281701377

5 27 56 27<<56|1 = 1945555039024054273

\*\*/

## xor linear basis

const int BITNUM **=** 63**;**

ll b**[**BITNUM**];**

void Insert**(**ll x**){**

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

**if(!(**x**>>**i**&**1**))** **continue;**

**if(**b**[**i**])** x **^=** b**[**i**];**

**else{**

b**[**i**]** **=** x**;**

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

**if(**b**[**j**]** **&&** **(**b**[**i**]>>**j**&**1**))** b**[**i**]** **^=** b**[**j**];**

**for(**int j **=** i**+**1**;** j **<** BITNUM**;** j**++)**

**if(**b**[**j**]>>**i**&**1**)** b**[**j**]** **^=** b**[**i**];**

**break;**

**}**

**}**

**}**

# Search

## exactCover

const int maxnode **=** 1e5**+**10**;**

const int maxn **=** 1e3**+**10**;**

struct DLX

**{**

int n**,** m**,** size**;**

int U**[**maxnode**],** D**[**maxnode**],** L**[**maxnode**],** R**[**maxnode**],** col**[**maxnode**],** row**[**maxnode**];**

int H**[**maxn**],** S**[**maxn**];**

int ansd**,** ans**[**maxn**];**

void init**(**int \_n**,** int \_m**)**

**{**

n **=** \_n**,** m **=** \_m**;**

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

S**[**i**]** **=** 0**,** U**[**i**]** **=** D**[**i**]** **=** i**,** L**[**i**]** **=** i **-** 1**,** R**[**i**]** **=** i **+** 1**;**

L**[**0**]** **=** m**,** R**[**m**]** **=** 0**;**

size **=** m**;**

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

**}**

void push**(**int r**,** int c**)**

**{**

**++**S**[**col**[++**size**]** **=** c**];**

row**[**size**]** **=** r**;**

D**[**size**]** **=** D**[**c**];**

U**[**D**[**c**]]** **=** size**;**

U**[**size**]** **=** c**;**

D**[**c**]** **=** size**;**

**if(**H**[**r**]** **<** 0**)**H**[**r**]** **=** L**[**size**]** **=** R**[**size**]** **=** size**;**

**else{**

R**[**size**]** **=** R**[**H**[**r**]];**

L**[**R**[**H**[**r**]]]** **=** size**;**

L**[**size**]** **=** H**[**r**];**

R**[**H**[**r**]]** **=** size**;**

**}**

**}**

void del**(**int c**)**

**{**

L**[**R**[**c**]]** **=** L**[**c**];**

R**[**L**[**c**]]** **=** R**[**c**];**

**for(**int i **=** D**[**c**];** i **!=** c**;** i **=** D**[**i**]){**

**for(**int j **=** R**[**i**];** j **!=** i**;** j **=** R**[**j**]){**

U**[**D**[**j**]]** **=** U**[**j**];**

D**[**U**[**j**]]** **=** D**[**j**];**

**--**S**[**col**[**j**]];**

**}**

**}**

**}**

void reback**(**int c**)**

**{**

**for(**int i **=** U**[**c**];** i **!=** c**;** i **=** U**[**i**])**

**for(**int j **=** L**[**i**];** j **!=** i**;** j **=** L**[**j**])**

**++**S**[**col**[**U**[**D**[**j**]]** **=** D**[**U**[**j**]]** **=** j**]];**

L**[**R**[**c**]]** **=** R**[**L**[**c**]]** **=** c**;**

**}**

bool dancing**(**int dep**)**

**{**

**if(**R**[**0**]** **==** 0**){**

ansd **=** dep**;**

**return** **true;**

**}**

int c **=** R**[**0**];**

**for(**int i **=** R**[**0**];** i **!=** 0**;** i **=** R**[**i**]){**

**if(**S**[**i**]** **<** S**[**c**])**c **=** i**;**

**}**

del**(**c**);**

**for(**int i **=** D**[**c**];** i **!=** c**;** i **=** D**[**i**]){**

ans**[**dep**]** **=** row**[**i**];**

**for(**int j **=** R**[**i**];** j **!=** i**;** j **=** R**[**j**])**del**(**col**[**j**]);**

**if(**dancing**(**dep **+** 1**))return** **true;**

**for(**int j **=** L**[**i**];** j **!=** i**;** j **=** L**[**j**])**reback**(**col**[**j**]);**

**}**

reback**(**c**);**

**return** **false;**

**}**

**}**dlx**;**

## multiCover

const int maxn **=** 1e3**+**10**;**

const int maxnode **=** 1e6**+**10**;**

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

struct DLX

**{**

int n**,**m**,**size**;**

int U**[**maxnode**],**D**[**maxnode**],**R**[**maxnode**],**L**[**maxnode**],**Row**[**maxnode**],**Col**[**maxnode**];**

int H**[**maxn**],**S**[**maxn**];**

int ansd**;**

void init**(**int \_n**,**int \_m**)**

**{**

ansd **=** INF**;**

n **=** \_n**,** m **=** \_m**;**

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

**{**

S**[**i**]** **=** 0**;**

U**[**i**]** **=** D**[**i**]** **=** i**;**

L**[**i**]** **=** i**-**1**;**

R**[**i**]** **=** i**+**1**;**

**}**

R**[**m**]** **=** 0**;** L**[**0**]** **=** m**;**

size **=** m**;**

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

**}**

void push**(**int r**,**int c**)**

**{**

**++**S**[**Col**[++**size**]=**c**];**

Row**[**size**]** **=** r**;**

D**[**size**]** **=** D**[**c**];**

U**[**D**[**c**]]** **=** size**;**

U**[**size**]** **=** c**;**

D**[**c**]** **=** size**;**

**if(**H**[**r**]** **<** 0**)**H**[**r**]** **=** L**[**size**]** **=** R**[**size**]** **=** size**;**

**else**

**{**

R**[**size**]** **=** R**[**H**[**r**]];**

L**[**R**[**H**[**r**]]]** **=** size**;**

L**[**size**]** **=** H**[**r**];**

R**[**H**[**r**]]** **=** size**;**

**}**

**}**

void del**(**int c**)**

**{**

**for(**int i **=** D**[**c**];**i **!=** c**;**i **=** D**[**i**])**

L**[**R**[**i**]]** **=** L**[**i**],** R**[**L**[**i**]]** **=** R**[**i**];**

**}**

void reback**(**int c**)**

**{**

**for(**int i **=** U**[**c**];**i **!=** c**;**i **=** U**[**i**])**

L**[**R**[**i**]]** **=** R**[**L**[**i**]]** **=** i**;**

**}**

bool v**[**maxn**];**

int f**()**

**{**

int ret **=** 0**;**

**for(**int c **=** R**[**0**];** c **!=** 0**;**c **=** R**[**c**])**v**[**c**]** **=** **true;**

**for(**int c **=** R**[**0**];** c **!=** 0**;**c **=** R**[**c**]){**

**if(**v**[**c**])**

**{**

ret**++;**

v**[**c**]** **=** **false;**

**for(**int i **=** D**[**c**];**i **!=** c**;**i **=** D**[**i**])**

**for(**int j **=** R**[**i**];**j **!=** i**;**j **=** R**[**j**])**

v**[**Col**[**j**]]** **=** **false;**

**}**

**}**

**return** ret**;**

**}**

void dancing**(**int d**)**

**{**

**if(**d **+** f**()** **>=** ansd**)return;**

**if(**R**[**0**]** **==** 0**)**

**{**

**if(**d **<** ansd**)**ansd **=** d**;**

**return;**

**}**

int c **=** R**[**0**];**

**for(**int i **=** R**[**0**];**i **!=** 0**;**i **=** R**[**i**])**

**if(**S**[**i**]** **<** S**[**c**])**

c **=** i**;**

**for(**int i **=** D**[**c**];**i **!=** c**;**i **=** D**[**i**])**

**{**

del**(**i**);**

**for(**int j **=** R**[**i**];**j **!=** i**;**j **=** R**[**j**])**del**(**j**),** **--**S**[**Col**[**j**]];**

dancing**(**d**+**1**);**

**for(**int j **=** L**[**i**];**j **!=** i**;**j **=** L**[**j**])**reback**(**j**),** **++**S**[**Col**[**j**]];**

reback**(**i**);**

**}**

**}**

**}**dlx**;**

## sudoku

#include <cstdio>

#include <cstring>

const int maxn **=** 1e3**+**10**;**

const int maxnode **=** 5e5**+**10**;**

char s**[**100**];**

struct DLX

**{**

int n**,** m**,** size**;**

int U**[**maxnode**],** D**[**maxnode**],** L**[**maxnode**],** R**[**maxnode**],** col**[**maxnode**],** row**[**maxnode**];**

int H**[**maxn**],** S**[**maxn**];**

int ansd**,** ans**[**maxn**];**

void init**(**int \_n**,** int \_m**)**

**{**

n **=** \_n**;**

m **=** \_m**;**

**for(**int i **=** 0**;** i **<=** m**;** i**++)**S**[**i**]** **=** 0**,** U**[**i**]** **=** D**[**i**]** **=** i**,** L**[**i**]** **=** i **-** 1**,** R**[**i**]** **=** i **+** 1**;**

L**[**0**]** **=** m**,** R**[**m**]** **=** 0**;**

size **=** m**;**

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

**}**

void push**(**int r**,** int c**)**

**{**

**++**S**[**col**[++**size**]** **=** c**];**

row**[**size**]** **=** r**;**

D**[**size**]** **=** D**[**c**];**

U**[**D**[**c**]]** **=** size**;**

U**[**size**]** **=** c**;**

D**[**c**]** **=** size**;**

**if(**H**[**r**]** **<** 0**)**H**[**r**]** **=** L**[**size**]** **=** R**[**size**]** **=** size**;**

**else{**

R**[**size**]** **=** R**[**H**[**r**]];**

L**[**R**[**H**[**r**]]]** **=** size**;**

L**[**size**]** **=** H**[**r**];**

R**[**H**[**r**]]** **=** size**;**

**}**

**}**

void del**(**int c**)**

**{**

L**[**R**[**c**]]** **=** L**[**c**];**

R**[**L**[**c**]]** **=** R**[**c**];**

**for(**int i **=** D**[**c**];** i **!=** c**;** i **=** D**[**i**]){**

**for(**int j **=** R**[**i**];** j **!=** i**;** j **=** R**[**j**]){**

U**[**D**[**j**]]** **=** U**[**j**];**

D**[**U**[**j**]]** **=** D**[**j**];**

**--**S**[**col**[**j**]];**

**}**

**}**

**}**

void reback**(**int c**)**

**{**

**for(**int i **=** U**[**c**];** i **!=** c**;** i **=** U**[**i**]){**

**for(**int j **=** L**[**i**];** j **!=** i**;** j **=** L**[**j**]){**

**++**S**[**col**[**U**[**D**[**j**]]** **=** D**[**U**[**j**]]** **=** j**]];**

**}**

**}**

L**[**R**[**c**]]** **=** R**[**L**[**c**]]** **=** c**;**

**}**

bool dancing**(**int dep**)**

**{**

**if(**R**[**0**]** **==** 0**){**

**for(**int i **=** 0**;** i **<** dep**;** i**++)**s**[(**ans**[**i**]-**1**)/**9**]** **=** **(**ans**[**i**]-**1**)%**9**+**'1'**;**

**for(**int i **=** 0**;** i **<** 81**;** i**++)**printf**(**"%c"**,** s**[**i**]);**

puts**(**""**);**

**return** **true;**

**}**

int c **=** R**[**0**];**

**for(**int i **=** R**[**0**];** i **!=** 0**;** i **=** R**[**i**])if(**S**[**i**]** **<** S**[**c**])**c **=** i**;**

del**(**c**);**

**for(**int i **=** D**[**c**];** i **!=** c**;** i **=** D**[**i**]){**

ans**[**dep**]** **=** row**[**i**];**

**for(**int j **=** R**[**i**];** j **!=** i**;** j **=** R**[**j**])**del**(**col**[**j**]);**

**if(**dancing**(**dep **+** 1**))return** **true;**

**for(**int j **=** L**[**i**];** j **!=** i**;** j **=** L**[**j**])**reback**(**col**[**j**]);**

**}**

reback**(**c**);**

**return** **false;**

**}**

**}**dlx**;**

void place**(**int **&**r**,** int **&**c1**,** int **&**c2**,** int **&**c3**,** int **&**c4**,** int i**,** int j**,** int k**)**

**{**

r **=** **(**i**\***9**+**j**)\***9**+**k**;**

c1 **=** i**\***9**+**j**+**1**;**

c2 **=** 9**\***9**+**i**\***9**+**k**;**

c3 **=** 9**\***9**\***2**+**j**\***9**+**k**;**

c4 **=** 9**\***9**\***3**+((**i**/**3**)\***3**+(**j**/**3**))\***9**+**k**;**

**}**

void solve**()**

**{**

dlx**.**init**(**9**\***9**\***9**,** 9**\***9**\***4**);**

int r**,** c1**,** c2**,** c3**,** c4**;**

**for(**int i **=** 0**;** i **<** 9**;** i**++){**

**for(**int j **=** 0**;** j **<** 9**;** j**++){**

**for(**int k **=** 1**;** k **<=** 9**;** k**++){**

**if(**s**[**i**\***9**+**j**]** **==** '.' **||** s**[**i**\***9**+**j**]-**'0' **==** k**){**

place**(**r**,** c1**,** c2**,** c3**,** c4**,** i**,** j**,** k**);**

dlx**.**push**(**r**,** c1**);**

dlx**.**push**(**r**,** c2**);**

dlx**.**push**(**r**,** c3**);**

dlx**.**push**(**r**,** c4**);**

**}**

**}**

**}**

**}**

dlx**.**dancing**(**0**);**

**}**

# String

## KMP

#include <cstring>

const int maxn **=** 1e6**+**10**;**

char s**[**maxn**];**//模式串

char t**[**maxn**];**//匹配串

int nxt**[**maxn**];**

int n**,** m**;**

void get\_next**()**

**{**

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

n **=** strlen**(**s**);**

nxt**[**0**]** **=** **-**1**;**

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

**while(**k **>=** 0 **&&** s**[**i**]** **!=** s**[**k**+**1**])**k **=** nxt**[**k**];**

**if(**s**[**i**]** **==** s**[**k**+**1**])**k**++;**

nxt**[**i**]** **=** k**;**

**}**

//int cyc = n-1-nxt[n-1];

//cyc = n%cyc?1:cyc;

**}**

bool match**()**

**{**

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

m **=** strlen**(**t**);**

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

**while(**k **>=** 0 **&&** t**[**i**]** **!=** s**[**k**+**1**])**k **=** nxt**[**k**];**

**if(**t**[**i**]** **==** s**[**k**+**1**])**k**++;**

**if(**k **==** n**-**1**)return** 1**;**

**}**

**return** 0**;**

**}**

## manacher

#include <cstring>

#include <algorithm>

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

const int maxn **=** 1e6**+**10**;**

char s**[**maxn**];**//原串

char t**[**maxn**];**//新串

int ma**[**maxn**];**

int n**;**//串长

int manacher**()**

**{**

t**[**0**]** **=** '$'**,** t**[**1**]** **=** '#'**;**

n **=** strlen**(**s**);**

**for(**int i **=** 0**;** i **<** n**;** i**++)**t**[**i**\***2**+**2**]** **=** s**[**i**],** t**[**i**\***2**+**3**]** **=** '#'**;**

int mx **=** **-**1**,** ctr **=** **-**1**,** ret **=** 1**;**

**for(**int i **=** 1**;** i **<** 2**\***n**+**2**;** i**++){**

**if(**mx **>** i**)**ma**[**i**]** **=** min**(**ma**[**2**\***ctr**-**i**],** mx**-**i**);**

**else** ma**[**i**]** **=** 1**;**

**while(**t**[**i**-**ma**[**i**]]** **==** t**[**i**+**ma**[**i**]])**ma**[**i**]++;**

**if(**i**+**ma**[**i**]-**1 **>** mx**){**

mx **=** i**+**ma**[**i**]-**1**;**

ctr **=** i**;**

**}**

ret **=** max**(**ret**,** ma**[**i**]-**1**);**

**}**

**return** ret**;**

**}**

## ahoCorasickAutomaton

#include <cstring>

#include <queue>

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

const int maxn **=** 1e6**+**10**;**

const int sigma **=** 26**;**

char s**[**maxn**];**

struct Node

**{**

int nxt**[**maxn**][**sigma**],** fail**[**maxn**],** last**[**maxn**],** idx**;**

int newNode**()**

**{**

int x **=** **++**idx**;**

memset**(**nxt**[**x**],** **-**1**,** **sizeof(**nxt**[**x**]));**

fail**[**x**]** **=** last**[**x**]** **=** 0**;**

**return** x**;**

**}**

void init**()**

**{**

idx **=** 0**;**

memset**(**nxt**[**0**],** **-**1**,** **sizeof(**nxt**[**0**]));**

fail**[**0**]** **=** last**[**0**]** **=** 0**;**

**}**

void insert**()**

**{**

int rt **=** 0**,** n **=** strlen**(**s**);**

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

int u **=** s**[**i**]-**'a'**;**

**if(**nxt**[**rt**][**u**]** **==** **-**1**)**nxt**[**rt**][**u**]** **=** newNode**();**

rt **=** nxt**[**rt**][**u**];**

**}**

last**[**rt**]++;**

**}**

void get\_fail**()**

**{**

queue**<**int**>** q**;**

int rt **=** 0**;**

fail**[**rt**]** **=** rt**;**

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

int **&**u **=** nxt**[**rt**][**i**];**

**if(**u **==** **-**1**)**u **=** rt**;**

**else{**

fail**[**u**]** **=** rt**;**

q**.**push**(**u**);**

**}**

**}**

**while(!**q**.**empty**()){**

int u **=** q**.**front**();** q**.**pop**();**

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

int **&**v **=** nxt**[**u**][**i**];**

int w **=** fail**[**u**];**

**if(**v **==** **-**1**)**v **=** nxt**[**w**][**i**];**

**else{**

fail**[**v**]** **=** nxt**[**w**][**i**];**

q**.**push**(**v**);**

**}**

**}**

**}**

**}**

int get\_ans**()**

**{**

int rt **=** 0**,** n **=** strlen**(**s**),** ret **=** 0**;**

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

int u **=** s**[**i**]-**'a'**;**

rt **=** nxt**[**rt**][**u**];**

int tmp **=** rt**;**

**while(**tmp**){**

**if(**last**[**tmp**])**ret **+=** last**[**tmp**],** last**[**tmp**]** **=** 0**;**

tmp **=** fail**[**tmp**];**

**}**

**}**

**return** ret**;**

**}**

**}**ACAM**;**

## palindromeAutomaton

#include <algorithm>

#include <cstring>

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

**typedef** long long ll**;**

const int maxn **=** 1e6**+**10**;**

const int sigma **=** 26**;**

char s**[**maxn**];**

struct Node

**{**

int idx**,** state**,** nxt**[**maxn**][**sigma**],** len**[**maxn**],** sz**[**maxn**],** fail**[**maxn**];**

void init**()**

**{**

idx **=** 1**,** state **=** 0**;**

fail**[**0**]** **=** fail**[**1**]** **=** 1**;** len**[**1**]** **=** **-**1**;**

memset**(**nxt**[**0**],** 0**,** **sizeof(**nxt**[**0**]));**

memset**(**nxt**[**1**],** 0**,** **sizeof(**nxt**[**1**]));**

**}**

int newNode**()**

**{**

int p **=** **++**idx**;**

memset**(**nxt**[**p**],** 0**,** **sizeof(**nxt**[**p**]));**

len**[**p**]** **=** sz**[**p**]** **=** fail**[**p**]** **=** 0**;**

**return** p**;**

**}**

void insert**(**int x**,** int pos**)**

**{**

int rt **=** state**;**

**while(**s**[**pos**-**len**[**rt**]-**1**]** **!=** s**[**pos**])**rt **=** fail**[**rt**];**

**if(!**nxt**[**rt**][**x**]){**

int v **=** newNode**(),** q **=** fail**[**rt**];**

len**[**v**]** **=** len**[**rt**]+**2**;**

**while(**s**[**pos**-**len**[**q**]-**1**]** **!=** s**[**pos**])**q **=** fail**[**q**];**

fail**[**v**]** **=** nxt**[**q**][**x**];**

nxt**[**rt**][**x**]** **=** v**;**

**}**

state **=** nxt**[**rt**][**x**],** sz**[**state**]++;**

**}**

ll count**()**

**{**

ll ans **=** 0**;**

**for(**int i **=** idx**;** i **>** 0**;** i**--){**

sz**[**fail**[**i**]]** **+=** sz**[**i**];**

ans **=** max**(**ans**,** 1LL**\***len**[**i**]\***sz**[**i**]);**

**}**

**return** ans**;**

**}**

**}**PAM**;**

## suffixAutomaton

#include <cstring>

#include <algorithm>

#include <vector>

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

#define pb push\_back

const int maxn **=** 1e6**+**10**;**

const int sigma **=** 26**;**

int ans**[**maxn**];**

char s**[**maxn**];**

struct Trie

**{**

int nxt**[**maxn**][**sigma**],** par**[**maxn**],** len**[**maxn**],** right**[**maxn**];**

vector**<**int**>** son**[**maxn**];**

int lst**,** idx**;**

int newNode**()**

**{**

int u **=** **++**idx**;**

memset**(**nxt**[**u**],** 0**,** **sizeof(**nxt**[**u**]));**

par**[**u**]** **=** len**[**u**]** **=** right**[**u**]** **=** 0**;**

**return** u**;**

**}**

void init**()**

**{**

idx **=** 0**;**

lst **=** newNode**();**

**}**

void extend**(**int c**)**

**{**

int rt **=** lst**,** np **=** newNode**();**

right**[**np**]** **=** 1**;**

len**[**np**]** **=** len**[**rt**]+**1**;**

**for(;** rt **&&** **!**nxt**[**rt**][**c**];** rt **=** par**[**rt**])**nxt**[**rt**][**c**]** **=** np**;**

**if(!**rt**)**par**[**np**]** **=** 1**;**

**else{**

int q **=** nxt**[**rt**][**c**];**

**if(**len**[**q**]** **==** len**[**rt**]+**1**)**par**[**np**]** **=** q**;**

**else{**

int nq **=** **++**idx**;**

right**[**nq**]** **=** 0**;**

par**[**nq**]** **=** par**[**q**];**

len**[**nq**]** **=** len**[**rt**]+**1**;**

memcpy**(**nxt**[**nq**],** nxt**[**q**],** **sizeof(**nxt**[**q**]));**

par**[**np**]** **=** par**[**q**]** **=** nq**;**

**for(;** rt **&&** nxt**[**rt**][**c**]** **==** q**;** rt **=** par**[**rt**])**nxt**[**rt**][**c**]** **=** nq**;**

**}**

**}**

lst **=** np**;**

**}**

void dfs**(**int u**)**

**{**

**for(**auto v **:** son**[**u**]){**

dfs**(**v**);**

right**[**u**]** **+=** right**[**v**];**

**}**

**}**

void solve**()**

**{**

**for(**int i **=** 2**;** i **<=** idx**;** i**++)**son**[**par**[**i**]].**pb**(**i**);**

dfs**(**1**);**

**for(**int i **=** 2**;** i **<=** idx**;** i**++)**ans**[**len**[**i**]]** **=** max**(**ans**[**len**[**i**]],** right**[**i**]);**

**}**

**}**SAM**;**

## smallestRepresentation

#include <cstring>

#include <algorithm>

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

const int maxn **=** 1e6**+**10**;**

char s**[**maxn**];**

int n**;**

int smallestRepresentation**()**

**{**

n **=** strlen**(**s**);**

int i **=** 0**,** j **=** 1**,** k **=** 0**;**

**while(**i **<** n **&&** j **<** n **&&** k **<** n**){**

**if(**s**[(**i**+**k**)%**n**]** **==** s**[(**j**+**k**)%**n**])**k**++;**

**else{**

**if(**s**[(**i**+**k**)%**n**]** **>** s**[(**j**+**k**)%**n**])**i **+=** k**+**1**;**

**else** j **+=** k**+**1**;**

**if(**i **==** j**)**j**++;**

k **=** 0**;**

**}**

**}**

**return** min**(**i**,** j**);**

**}**

# Others

## BM

#include <cstdio>

#include <vector>

#include <cassert>

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

#define sc(x) scanf("%d", &x)

#define rep(i,a,n) for (int i=a;i<n;i++)

#define pb push\_back

#define SZ(x) ((int)(x).size())

**typedef** vector**<**int**>** VI**;**

**typedef** long long ll**;**

const ll mod**=**1e9**+**7**;**

ll powmod**(**ll a**,**ll b**)** **{**ll res**=**1**;**a**%=**mod**;** assert**(**b**>=**0**);** **for(;**b**;**b**>>=**1**){if(**b**&**1**)**res**=**res**\***a**%**mod**;**a**=**a**\***a**%**mod**;}return** res**;}**

**namespace** linear\_seq **{**

const int maxn**=**1e3**+**10**;**

ll res**[**maxn**],**base**[**maxn**],**\_c**[**maxn**],**\_md**[**maxn**];**

vector**<**int**>** Md**;**

void mul**(**ll **\***a**,**ll **\***b**,**int k**)** **{**

rep**(**i**,**0**,**k**+**k**)** \_c**[**i**]=**0**;**

rep**(**i**,**0**,**k**)** **if** **(**a**[**i**])** rep**(**j**,**0**,**k**)** \_c**[**i**+**j**]=(**\_c**[**i**+**j**]+**a**[**i**]\***b**[**j**])%**mod**;**

**for** **(**int i**=**k**+**k**-**1**;**i**>=**k**;**i**--)** **if** **(**\_c**[**i**])**

rep**(**j**,**0**,**SZ**(**Md**))** \_c**[**i**-**k**+**Md**[**j**]]=(**\_c**[**i**-**k**+**Md**[**j**]]-**\_c**[**i**]\***\_md**[**Md**[**j**]])%**mod**;**

rep**(**i**,**0**,**k**)** a**[**i**]=**\_c**[**i**];**

**}**

int solve**(**ll n**,**VI a**,**VI b**)** **{** // a 系数 b 初值 b[n+1]=a[0]\*b[n]+...

ll ans**=**0**,**pnt**=**0**;**

int k**=**SZ**(**a**);**

assert**(**SZ**(**a**)==**SZ**(**b**));**

rep**(**i**,**0**,**k**)** \_md**[**k**-**1**-**i**]=-**a**[**i**];**\_md**[**k**]=**1**;**

Md**.**clear**();**

rep**(**i**,**0**,**k**)** **if** **(**\_md**[**i**]!=**0**)** Md**.**push\_back**(**i**);**

rep**(**i**,**0**,**k**)** res**[**i**]=**base**[**i**]=**0**;**

res**[**0**]=**1**;**

**while** **((**1ll**<<**pnt**)<=**n**)** pnt**++;**

**for** **(**int p**=**pnt**;**p**>=**0**;**p**--)** **{**

mul**(**res**,**res**,**k**);**

**if** **((**n**>>**p**)&**1**)** **{**

**for** **(**int i**=**k**-**1**;**i**>=**0**;**i**--)** res**[**i**+**1**]=**res**[**i**];**res**[**0**]=**0**;**

rep**(**j**,**0**,**SZ**(**Md**))** res**[**Md**[**j**]]=(**res**[**Md**[**j**]]-**res**[**k**]\***\_md**[**Md**[**j**]])%**mod**;**

**}**

**}**

rep**(**i**,**0**,**k**)** ans**=(**ans**+**res**[**i**]\***b**[**i**])%**mod**;**

**if** **(**ans**<**0**)** ans**+=**mod**;**

**return** ans**;**

**}**

VI BM**(**VI s**)** **{**

VI C**(**1**,**1**),**B**(**1**,**1**);**

int L**=**0**,**m**=**1**,**b**=**1**;**

rep**(**n**,**0**,**SZ**(**s**))** **{**

ll d**=**0**;**

rep**(**i**,**0**,**L**+**1**)** d**=(**d**+(**ll**)**C**[**i**]\***s**[**n**-**i**])%**mod**;**

**if** **(**d**==**0**)** **++**m**;**

**else** **if** **(**2**\***L**<=**n**)** **{**

VI T**=**C**;**

ll c**=**mod**-**d**\***powmod**(**b**,**mod**-**2**)%**mod**;**

**while** **(**SZ**(**C**)<**SZ**(**B**)+**m**)** C**.**pb**(**0**);**

rep**(**i**,**0**,**SZ**(**B**))** C**[**i**+**m**]=(**C**[**i**+**m**]+**c**\***B**[**i**])%**mod**;**

L**=**n**+**1**-**L**;** B**=**T**;** b**=**d**;** m**=**1**;**

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

ll c**=**mod**-**d**\***powmod**(**b**,**mod**-**2**)%**mod**;**

**while** **(**SZ**(**C**)<**SZ**(**B**)+**m**)** C**.**pb**(**0**);**

rep**(**i**,**0**,**SZ**(**B**))** C**[**i**+**m**]=(**C**[**i**+**m**]+**c**\***B**[**i**])%**mod**;**

**++**m**;**

**}**

**}**

**return** C**;**

**}**

int gao**(**VI a**,**ll n**)** **{**

VI c**=**BM**(**a**);**

c**.**erase**(**c**.**begin**());**

rep**(**i**,**0**,**SZ**(**c**))** c**[**i**]=(**mod**-**c**[**i**])%**mod**;**

**return** solve**(**n**,**c**,**VI**(**a**.**begin**(),**a**.**begin**()+**SZ**(**c**)));**

**}**

**};**

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

int T**;**

sc**(**T**);**

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

ll n**;**

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

printf**(**"%d\n"**,**linear\_seq**::**gao**(**VI**{**2**,**3**,**4**,**6**,**9**,**13**},**n**-**1**));**

**}**

**}**

## contor

#include <cstring>

const int maxn **=** 13**;**

int fac**[**maxn**];**

int s**[**maxn**];**

bool tmp**[**maxn**];**

int n**;**

void init**()**

**{**

fac**[**0**]** **=** 1**;**

**for(**int i **=** 1**;** i **<** maxn**;** i**++)**fac**[**i**]** **=** fac**[**i**-**1**]\***i**;**

**}**

int cantor**()**

**{**

int num**=**0**;**

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

**{**

int tmp **=** 0**;**

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

**if(**s**[**j**]** **<** s**[**i**])**tmp**++;**

num **+=** fac**[**n**-**i**-**1**]\***tmp**;**

**}**

**return** num**;**

**}**

void \_cantor**(**int x**)**

**{**

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

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

**{**

int k **=** x**/**fac**[**i**];** x **%=** fac**[**i**];**

int j **=** 1**;**

**for(**int sum **=** 0**;** sum **<** k **||** tmp**[**j**];** j**++)**

**if(!**tmp**[**j**])**sum**++;**

s**[**n**-**1**-**i**]=**j**;**

tmp**[**j**]=**1**;**

**}**

**}**

## input\_plug

//fread版本

**namespace** IO **{**

const int MT **=** 27 **\*** 1024 **\*** 1024**;** /// 10MB 请注意输入数据的大小！！！

char IO\_BUF**[**MT**];**

int IO\_PTR**,** IO\_SZ**;**

/// 要记得把这一行添加到main函数第一行！！！

void begin**()** **{**

IO\_PTR **=** 0**;**

IO\_SZ **=** fread **(**IO\_BUF**,** 1**,** MT**,** stdin**);**

**}**

template**<**typename T**>**

inline bool scan\_d **(**T **&** t**)** **{**

**while** **(**IO\_PTR **<** IO\_SZ **&&** IO\_BUF**[**IO\_PTR**]** **!=** '-' **&&** **(**IO\_BUF**[**IO\_PTR**]** **<** '0' **||** IO\_BUF**[**IO\_PTR**]** **>** '9'**))**

IO\_PTR **++;**

**if** **(**IO\_PTR **>=** IO\_SZ**)** **return** **false;**

bool sgn **=** **false;**

**if** **(**IO\_BUF**[**IO\_PTR**]** **==** '-'**)** sgn **=** **true,** IO\_PTR **++;**

**for** **(**t **=** 0**;** IO\_PTR **<** IO\_SZ **&&** '0' **<=** IO\_BUF**[**IO\_PTR**]** **&&** IO\_BUF**[**IO\_PTR**]** **<=** '9'**;** IO\_PTR **++)**

t **=** t **\*** 10 **+** IO\_BUF**[**IO\_PTR**]** **-** '0'**;**

**if** **(**sgn**)** t **=** **-**t**;**

**return** **true;**

**}**

inline bool scan\_s **(**char s**[])** **{**

**while** **(**IO\_PTR **<** IO\_SZ **&&** **(**IO\_BUF**[**IO\_PTR**]** **==** ' ' **||** IO\_BUF**[**IO\_PTR**]** **==** '\n'**)** **)** IO\_PTR **++;**

**if** **(**IO\_PTR **>=** IO\_SZ**)** **return** **false;**

int len **=** 0**;**

**while** **(**IO\_PTR **<** IO\_SZ **&&** IO\_BUF**[**IO\_PTR**]** **!=** ' ' **&&** IO\_BUF**[**IO\_PTR**]** **!=** '\n'**)**

s**[**len **++]** **=** IO\_BUF**[**IO\_PTR**],** IO\_PTR **++;**

s**[**len**]** **=** '\0'**;**

**return** **true;**

**}**

template**<**typename T**>**

void print**(**T x**)** **{**

static char s**[**33**],** **\***s1**;** s1 **=** s**;**

**if** **(!**x**)** **\***s1**++** **=** '0'**;**

**if** **(**x **<** 0**)** putchar**(**'-'**),** x **=** **-**x**;**

**while(**x**)** **\***s1**++** **=** **(**x **%** 10 **+** '0'**),** x **/=** 10**;**

**while(**s1**--** **!=** s**)** putchar**(\***s1**);**

**}**

template**<**typename T**>**

void println**(**T x**)** **{**

print**(**x**);** putchar**(**'\n'**);**

**}**

**};**

//getchar版本

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;**

**}**

inline bool scan\_lf**(**double **&**num**){**

char in**;**double Dec**=**0.1**;**

bool IsN**=false,**IsD**=false;**

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

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

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

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

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

**else** **if(**in**==**'.'**){**IsD**=true;**num**=**0**;}**

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

**if(!**IsD**){**

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

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

**}**

**if(**in**!=**'.'**){**

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

**return** **true;**

**}else{**

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

num**+=**Dec**\*(**in**-**'0'**);**Dec**\*=**0.1**;**

**}**

**}**

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

**return** **true;**

**}**

## java

1. 输入输出

格式为：Scanner cin **=** **new** Scanner **(new**

BufferedInputStream**(**System**.**in**));**

或者、Scanner cin **=** **new** Scanner **(**System**.**in**);**

函数：System**.**out**.**print**();** System**.**out**.**println**();**

System**.**out**.**printf**();**

System**.**out**.**print**();** // cout << …;

System**.**out**.**println**();** // cout << … << endl;

System**.**out**.**printf**();** // 与C 中的printf 用法类似.

例程：

**import** java**.**io**.\*;**

**import** java**.**math**.\*;**

**import** java**.**util**.\*;**

**import** java**.**text**.\*;**

public class Main**{**

public static void main**(**String**[]** args**)**

**{**

Scanner cin **=** **new** Scanner **(new**

BufferedInputStream**(**System**.**in**));**

int a**;** double b**;** BigInteger c**;** String st**;**

a **=** cin**.**nextInt**();** b **=** cin**.**nextDouble**();** c **=**cin**.**nextBigInteger**();** d **=** cin**.**nextLine**();**

// 每种类型都有相应的输入函数.

a **=** 12345**;** b **=** 1.234567**;**

System**.**out**.**println**(**a **+** " " **+** b**);**

System**.**out**.**printf**(**"%d %10.5f\n"**,** a**,** b**);**

// 输入b 为字宽为10，右对齐，保留小数点后5 位，四舍五入.

**}**

**}**

2. 字符串处理

java 中字符串String 是不可以修改的，要修改只能转换为字符数组**.**

例程：

**{**

int i**;**

Scanner cin **=** **new** Scanner **(new**

BufferedInputStream**(**System**.**in**));**

String st **=** "abcdefg"**;**

System**.**out**.**println**(**st**.**charAt**(**0**));** // st.charAt(i)就相当于st[i].

char **[]** ch **=** st**.**toCharArray**();** // 字符串转换为字符数组.

**for** **(**i **=** 0**;** i **<** ch**.**length**;** i**++)** ch**[**i**]** **+=** 1**;**

System**.**out**.**println**(**ch**);** // 输入为“bcdefgh”.

**if** **(**st**.**startsWith**(**"a"**))** // 如果字符串以'0'开头.

**{**

st **=** st**.**substring**(**1**);** // 则从第1 位开始copy(开头为第0 位).

**}**

**}**

3. 排序

函数：Arrays**.**sort**();**

例程：

Scanner cin **=** **new** Scanner **(new** BufferedInputStream**(**System**.**in**));**

int n **=** cin**.**nextInt**();**

int a**[]** **=** **new** int **[**n**];**

**for** **(**int i **=** 0**;** i **<** n**;** i**++)** a**[**i**]** **=** cin**.**nextInt**();**

Arrays**.**sort**(**a**);**

**for** **(**int i **=** 0**;** i **<** n**;** i**++)** System**.**out**.**print**(**a**[**i**]** **+** " "**);**

4. 结构体排序

例子：一个结构体有两个元素String x，int y，排序，如果x 相等y 升序，否者x 升序。

Comparator

强行对某个对象collection 进行整体排序的比较函数，可以将

Comparator 传递给Collections**.**sort 或Arrays**.**sort。

接口方法：这里也给出了两种方法。

**import** java**.**util**.\*;**

class structSort**{**

String x**;**

int y**;**

**}**

class cmp **implements** Comparator**<**structSort**>{**

public int compare**(**structSort o1**,** structSort o2**)** **{**

**if(**o1**.**x**.**compareTo**(**o2**.**x**)** **==** 0**){**//这个相当于c/c++中strcmp（o1.x , o2,x）

**return** o1**.**y **-** o2**.**y**;**

**}**

**return** o1**.**x**.**compareTo**(**o2**.**x**);**

**}**

**}**

public class Main **{**

public static void main**(**String**[]** args**)** **{**

Comparator**<**structSort**>** comparator **=** **new** Comparator**<**structSort**>(){**

public int compare**(**structSort o1**,** structSort o2**)** **{**

**if(**o1**.**x**.**compareTo**(**o2**.**x**)** **==** 0**){**

**return** o1**.**y **-** o2**.**y**;**

**}**

**return** o1**.**x**.**compareTo**(**o2**.**x**);**

**}**

**};**

Scanner cin **=** **new** Scanner**(**System**.**in**);**

int n **=** cin**.**nextInt**();**

structSort a**[]** **=** **new** structSort**[**10**];**

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

a**[**i**]** **=** **new** structSort**();**

a**[**i**].**x **=** cin**.**next**();**

a**[**i**].**y **=** cin**.**nextInt**();**

**}**

Arrays**.**sort**(**a**,**0**,**n**,**comparator**);**//这个直接使用Comparator

Arrays**.**sort**(**a**,**0**,**n**,new** cmp**());**//这个实现Comparator，就就跟c++中的sort 函数调用就差不多了

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

System**.**out**.**println**(**a**[**i**].**x**+**" "**+**a**[**i**].**y**);**

**}**

**}**

**}**

Comparable强行对实现它的每个类的对象进行整体排序，实现此接口的对象列表（和数组）可以通过Collections**.**sort 或Arrays**.**sort 进行自动排序。就是输入完了直接就默认排序了，

接口方法：

**import** java**.**util**.\*;**

class structSort **implements** Comparable**<**structSort**>{**

String x**;**

int y**;**

public int compareTo**(**structSort o1**)** **{**

**if(this.**x**.**compareTo**(**o1**.**x**)** **==** 0**){**

**return** **this.**y **-** o1**.**y**;**

**}**

**return** **this.**x**.**compareTo**(**o1**.**x**);**

**}**

**}**

public class Main **{**

public static void main**(**String**[]** args**)** **{**

Scanner cin **=** **new** Scanner**(**System**.**in**);**

int n **=** cin**.**nextInt**();**

structSort a**[]** **=** **new** structSort**[**10**];**

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

a**[**i**]** **=** **new** structSort**();**

a**[**i**].**x **=** cin**.**next**();**

a**[**i**].**y **=** cin**.**nextInt**();**

**}**

Arrays**.**sort**(**a**,**0**,**n**);**

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

System**.**out**.**println**(**a**[**i**].**x**+**" "**+**a**[**i**].**y**);**

**}**

**}**

**}**

## vimrc\_long

set nocompatible

syntax on

colo desert

set nu

set tabstop=4

set softtabstop=4

set shiftwidth=4

set cursorline

set cindent

set ruler

set expandtab

set clipboard=unnamed

set backspace=indent,eol,start

au BufNewFile \*.cpp 0r head.cpp

map <F10> :sil call Run()<CR>

map <F9> :sil call CompileRun()<CR>

inoremap { {}<ESC>i

function! Run()

winc l

winc k

w

winc h

w

only

!(time ./%< <%<.in >%<.out)2>>%<.out

bel vs %<.in

bel sv %<.out

winc h

redr

colo desert

endfunc

function! CompileRun()

set mp=g++\ %\ -O2\ -Wall\ --std=c++11\ -o\ %<\ -DLOCAL

winc l

winc k

w

winc h

w

only

let v:statusmsg = ''

make

if empty(v:statusmsg)

call Run()

else

cw

winc k

redr

colo desert

endif

endfunc

## vimrc\_short

set nu mouse=a sw=4 ts=4 sts=4

set mp=g++\ %<\ -O2\ -Wall\ --std=c++11\ -g\ -o\ %<

nmap <F2> :bel vs %<.in <CR>

nmap <F8> :!time ./%< < %<.in <CR>

nmap <F9> :w <CR> :make<CR>