目录

[max flow minimem cost 3](#_Toc448001003)

[Merge sort reverse 5](#_Toc448001004)

[prim 6](#_Toc448001005)

[Kruskal 7](#_Toc448001006)

[Segment tree 8](#_Toc448001007)

[Spare Table 10](#_Toc448001008)

[tarjan 11](#_Toc448001009)

[Trie 12](#_Toc448001010)

[4 points on a plane 15](#_Toc448001011)

[BIT 15](#_Toc448001012)

[Cantor 16](#_Toc448001013)

[Dijstra 17](#_Toc448001014)

[Dinic 18](#_Toc448001015)

[floyed 19](#_Toc448001016)

[Wythoff 20](#_Toc448001017)

[hangary 20](#_Toc448001018)

[josephus 21](#_Toc448001019)

[KMP 22](#_Toc448001020)

[Manacher 22](#_Toc448001021)

[Matrix pow 23](#_Toc448001022)

[Math 25](#_Toc448001023)

max flow minimem cost

const int maxn=120;

const int oo = 0x3f3f3f3f;

struct Edge

{

int u, v, cap, flow, cost;

Edge(int u, int v, int cap, int f, int cost):u(u), v(v), cap(cap), flow(f), cost(cost) {}

};

struct MCMF

{

int n, m, s, t;

vector<Edge> edge;

vector<int> G[maxn];

int inq[maxn], d[maxn], p[maxn], a[maxn];

void init(int n)

{

this->n=n;

for(int i=0; i<=n; i++)G[i].clear();

edge.clear();

}

void AddEdge(int u, int v, int cap, int cost)

{

edge.push\_back(Edge(u, v, cap, 0, cost));

edge.push\_back(Edge(v, u, 0, 0, -cost));

m=edge.size();

G[u].push\_back(m-2);

G[v].push\_back(m-1);

}

bool SPFA(int s, int t, int& flow, int& cost)

{

memset(d, 0x3f, sizeof d);

memset(inq, 0, sizeof inq);

d[s]=0, inq[s]=1, p[s]=0, a[s]=oo;

queue<int> q;

q.push(s);

while(!q.empty())

{

int u=q.front();

q.pop();

inq[u]=0;

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

{

Edge& e=edge[G[u][i]];

if(e.cap>e.flow && d[e.v]>d[u]+e.cost)

{

d[e.v]=d[u]+e.cost;

p[e.v]=G[u][i];

a[e.v]=min(a[u], e.cap-e.flow);

if(!inq[e.v])

{

q.push(e.v);

inq[e.v]=true;

}

}

}

}

if(d[t]==oo)return false;

flow+=a[t];

cost+=d[t]\*a[t];

int u=t;

while(u!=s)

{

edge[p[u]].flow+=a[t];

edge[p[u]^1].flow-=a[t];

u=edge[p[u]].u;

}

return true;

}

int Mincost(int s, int t, int& cost)

{

int flow=0;

while(SPFA(s, t, flow, cost))

;

return flow;

}

} net;

int ord[55][55], sto[55][55];

int main()

{

int n, m, k;

while(~scanf("%d%d%d", &n, &m, &k) && n+m+k)

{

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

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

scanf("%d", &ord[i][j]);

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

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

scanf("%d", &sto[i][j]);

int S=0, T=n+m+2;

int cost=0;

for(int p=1; p<=k; p++)

{

int sum=0;

net.init(n+m+10);

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

{

net.AddEdge(i, T, ord[i][p], 0);

sum+=ord[i][p];

}

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

net.AddEdge(S, i+n, sto[i][p], 0);

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

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

{

int x;

scanf("%d", &x);

net.AddEdge(n+j, i, oo, x);

}

if(~cost && net.Mincost(S, T, cost)<sum)

cost=-1;

}

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

}

return 0;

}

Merge sort reverse

int arr[1000200], tarr[1000200];

int cnt;

void merge(int low, int mid, int high)

{

int i, j, k;

for (i = low, j = mid + 1, k = 0; i <= mid && j <= high;)

{

if(arr[i] < arr[j])

tarr[k++] = arr[i++];

else

{

tarr[k++] = arr[j++];

cnt += mid - i + 1;

}

}

while(i <= mid) tarr[k++] = arr[i++];

while(j <= high) tarr[k++] = arr[j++];

for (k = 0; low <= high; low++, k++)

arr[low] = tarr[k];

}

void mergesort(int low, int high)

{

if(low == high) return;

int mid = (low + high) / 2;

mergesort(low, mid);

mergesort(mid + 1, high);

merge(low, mid, high);

}

int main()

{

int n;

scanf("%d", &n);

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

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

cnt = 0;

mergesort(0, n-1);

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

return 0;

}

prim

struct Edge

{

int u, v, c;

Edge(){}

Edge(int u, int v, int c):u(u),v(v),c(c){}

};

vector<Edge> G[10020];

void addedge(int u, int v, int c)

{

G[u].push\_back(Edge(u,v,c));

G[v].push\_back(Edge(v,u,c));

}

int n, m;

int vis[10020];

int dist[10020];

int prim()

{

int ans = 0;

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

memset(dist, 0x3f, sizeof(dist));

vis[1] = 1;

int minid, minc;

int now = 1;

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

{

for (int i = 0, len = G[now].size(); i < len; i++)

{

int to = G[now][i].v, c = G[now][i].c;

if(vis[to] == 1) continue;

if(dist[to] > c)

dist[to] = c;

}

minid = -1;

minc = 0x3f3f3f3f;

for (int i = 1; i <= n; i++) if((!vis[i]) && dist[i] < minc)

{

minid = i;

minc = dist[i];

}

ans += minc;

vis[minid] = 1;

now = minid;

}

return ans;

}

int main()

{

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

int u,v,c;

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

{

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

addedge(u,v,c);

}

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

return 0;

}

Kruskal

struct Edge

{

int u, v, c;

Edge(){}

Edge(int u, int v, int c):u(u),v(v),c(c){}

bool operator < (const Edge &e) const {

return c < e.c;

}

};

vector<Edge> ve;

int n, m;

int R[10020];

// int root(int x)

// {

// while(R[x] != x)

// x = R[x] = R[R[x]];

// return R[x];

// }

int root(int x)

{

if(R[x] == -1) return x;

if(R[x] != -1) R[x] = root(R[x]);

return R[x];

}

int main()

{

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

int u, v, c;

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

{

scanf("%d%d%d", &u,&n,&c);

ve.push\_back(Edge(u,n,c));

}

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

// R[i] = i;

memset(R, -1, sizeof(R));

sort(ve.begin(), ve.end());

int ans = 0;

int Ru, Rv;

for (int i = 0, len = ve.size(); i < len; i++)

{

Edge &now = ve[i];

Ru = root(now.u);

Rv = root(now.v);

if(Ru != Rv)

{

ans += now.c;

R[Ru] = Rv;

}

}

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

return 0;

}

Segment tree

#define maxn 100200

#define ll long long

#define lson l, m, rt<<1

#define rson m+1, r, rt<<1|1

using namespace std;

struct SegTree{

ll segsum[maxn<<2];

ll lazy[maxn<<2];

void clear()

{

memset(segsum, 0, sizeof(segsum));

memset(lazy, 0, sizeof(lazy));

}

void pushup(int rt)

{

segsum[rt] = segsum[rt<<1] + segsum[rt<<1|1];

}

void build(int l, int r, int rt)

{

if(l == r)

{

scanf("%lld",&segsum[rt]);

return;

}

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

build(lson);

build(rson);

pushup(rt);

}

void pushdown(int rt, int m)

{

lazy[rt << 1] += lazy[rt];

lazy[rt << 1 | 1] += lazy[rt];

segsum[rt << 1] += lazy[rt] \* (m - (m >> 1));

segsum[rt << 1 | 1] += lazy[rt] \* (m >> 1);

lazy[rt] = 0;

}

void update(int L, int R, int c, int l, int r, int rt)

{

if(L <= l && r <= R)

{

lazy[rt] += c;

segsum[rt] += (r - l + 1) \* c;

return;

}

if(lazy[rt] != 0)

pushdown(rt, r - l + 1);

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

if(L <= m) update(L, R, c, lson);

if(R > m) update(L, R, c, rson);

pushup(rt);

}

void check(int l, int r, int rt)

{

printf("l=%d r=%d rt=%d sum=%lld\n", l,r,rt,segsum[rt]);

if(l == r)

{

return;

}

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

check(lson);

check(rson);

}

ll querysum(int L, int R, int l, int r, int rt){

if(L <= l && R >= r)

{

return segsum[rt];

}

if(lazy[rt] != 0)

pushdown(rt, r - l + 1);

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

ll tmp = 0;

if(L <= m) tmp += querysum(L, R, lson);

if(R > m) tmp += querysum(L, R, rson);

return tmp;

}

}segtree;

int main()

{

int n, q;

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

segtree.clear();

segtree.build(1, n, 1);

char cmd[2];

int x, y ,z;

while(q--){

scanf("%s", cmd);

if(cmd[0] == 'Q')

{

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

printf("%lld\n", segtree.querysum(x, y, 1, n, 1));

}else{

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

segtree.update(x, y, z, 1, n, 1);

}

}

return 0;

}

Spare Table

using namespace std;

struct ST

{

int high[maxn][33],low[maxn][33], a[maxn];

int n;

int depth;

void clear()

{

n = 0;

depth = 1;

memset(high, 0, sizeof(high));

memset(low, 0, sizeof(low));

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

}

void rmq()

{

for(int j = 1; j <= 20; j++)

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

high[i][j] = max(high[i][j - 1], high[i + (1 << (j - 1))][j - 1]);

low[i][j] = min(low[i][j - 1], low[i + (1 << (j - 1))][j - 1]);

}

}

void init()

{

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

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

high[i][0] = low[i][0] = a[i];

}

}

int query(int s, int e)

{

int k = log2(e - s + 1);

// printf("max : %d , min : %d\n",max(high[s][k], high[e - (1 << k) + 1][k]), min(low[s][k], low[e - (1 << k) + 1][k]));

return max(high[s][k], high[e - (1 << k) + 1][k]) - min(low[s][k], low[e - (1 << k) + 1][k]);

}

void check()

{

for(int j = 0; j <= 3; j++)

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

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

printf("\n\n");

}

}st;

int main()

{

st.clear();

int m;

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

st.init();

st.rmq();

// st.check();

int s, e;

while(m--)

{

scanf("%d%d",&s,&e);

printf("%d\n",st.query(s, e));

}

return 0;

}

tarjan

using namespace std;

vector<int>v[maxn];

bool instack[maxn];

int dfn[maxn], low[maxn];

int n, m;

int depth, strongcnt;

int belong[maxn], strongsize[maxn];

int stk[maxn], top;

int inde[maxn], outde[maxn];

void clear(){

memset(dfn, 0, sizeof(dfn));

memset(low, 0, sizeof(low));

memset(instack, 0, sizeof(instack));

memset(belong, 0, sizeof(belong));

memset(strongsize, 0, sizeof(strongsize));

memset(inde, 0, sizeof(inde));

memset(outde, 0, sizeof(outde));

depth = 0;

strongcnt = 0;

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

{

v[i].clear();

}

}

void tarjan(int u){

dfn[u] = low[u] = ++depth;

instack[u] = true;

stk[++top] = u;

int to;

for (int i = 0; i < v[u].size(); ++i)

{

to = v[u][i];

if(!dfn[to]){

tarjan(to);

low[u] = min(low[to], low[u]);

}else if (instack[to])

{

//dfn[i] not low[i];

low[u] = min(low[u], dfn[to]);

}

}

if (dfn[u] == low[u])

{

strongcnt++;

do

{

to = stk[top--];

instack[to] = false;

belong[to] = strongcnt;

strongsize[strongcnt]++;

} while (to != u);

}

}

int main()

{

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

{

if (!dfn[i])

{

tarjan(i);

}

}

int to;

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

{

for (int j = 0; j < v[i].size(); ++j)

{

to = v[i][j];

if(belong[i] != belong[to]){

outde[belong[i]]++;

inde[belong[to]]++;

}

}

}

return 0;

}

Trie

struct Trie

{

struct Node

{

bool end;

int id;

Node \*next[26];

};

Node \*head;

void clear()

{

head = new Node();

}

void insert(char \*s, int id)

{

int len = strlen(s);

Node \*now = head;

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

{

int x = s[i] - 'a';

if (now->next[x] == NULL)

{

now->next[x] = new Node();

now->next[x]->end = false;

memset(now->next[x]->next, 0, sizeof(now->next[x]->next));

}

now = now->next[x];

if (i == len - 1)

{

now->end = true;

now->id = id;

}

}

}

int query(char \*s)

{

int len = strlen(s);

Node \*now = head;

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

{

int x = s[i] - 'a';

if (now->next[x] == NULL)

return false;

now = now->next[x];

if (i == len - 1){

if(now->end) return now->id;

else return 0;

}

}

return 0;

}

};

const int maxm = 10000000;

struct Trie

{

struct Node

{

bool end;

int id;

int next[26];

}node[maxm];

int head, tot;

void clear()

{

head = 0;

memset(node[head].next, -1, sizeof(node[head].next));

tot = 0;

}

void insert(char \*s, int id)

{

int len = strlen(s);

int nowid = head;

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

{

Node& now = node[nowid];

int x = s[i] - 'a';

if (now.next[x] == -1)

{

now.next[x] = ++tot;

node[tot].end = false;

memset(node[tot].next, -1, sizeof(node[tot].next));

}

nowid = now.next[x];

if (i == len - 1)

{

node[nowid].end = true;

node[nowid].id = id;

}

}

}

int query(char \*s)

{

int len = strlen(s);

int nowid = head;

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

{

Node& now = node[nowid];

int x = s[i] - 'a';

if (now.next[x] == -1)

return false;

nowid = now.next[x];

if (i == len - 1){

if(node[nowid].end) return node[nowid].id;

else return 0;

}

}

return 0;

}

};

4 points on a plane

using namespace std;

struct Point3 {

double x, y, z;

Point3 operator - ( Point3 & p ) {

Point3 ans;

ans.x = this->x - p.x;

ans.y = this->y - p.y;

ans.z = this->z - p.z;

return ans;

}

};

Point3 operator \* ( const Point3 & a, const Point3 & b ) {

Point3 ans;

ans.x = a.y \* b.z - a.z \* b.y;

ans.y = a.z \* b.x - a.x \* b.z;

ans.z = a.x \* b.y - a.y \* b.x;

return ans;

}

double dot( const Point3 & a, const Point3 & b ) {

return a.x \* b.x + a.y \* b.y + a.z \* b.z;

}

int main() {

Point3 p[4];

int T;

cin >> T;

while(T--)

{

for( int i = 0; i < 4; ++i ) scanf( "%lf%lf%lf", &p[i].x, &p[i].y, &p[i].z );

puts( dot( p[3] - p[0], (p[2] - p[0])\*(p[1] - p[0])) == 0.0 ? "Yes" : "No" );

}

return 0;

}

BIT

struct BIT{

int c[maxn];

int n;

void clear(int n){

memset(c, 0, sizeof(c));

this->n = n;

}

inline int lowbit(int x){

return x & (-x);

}

void add(int pos, int delta){

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

while(pos < maxn){

c[pos] += delta;

pos += lowbit(pos);

}

}

int getsum(int pos){

int ans = 0;

while(pos > 0){

ans += c[pos];

pos -= lowbit(pos);

}

return ans;

}

}bit;

Cantor

/\*

\* 康拓展开

\* 元素个数 len

\* 0-based count

\* last edit : 2015/9/25

\*/

int fact[10] = {1,1,2,6,24,120,720,5040,40320,362880};

int cantor(int\* a,int len)

{

int ret = 0;

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

{

int tmp = 0;

for(int j = i+1; j < len; j++)if(a[i] > a[j]) tmp++;

ret += tmp \* fact[len-i-1];

}

return ret;

}

void cantorrev(int\* a,int d, int len)

{

int vis[10] = {0}, tmp, tt;

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

{

tmp = d / fact[len-i-1];

d %= fact[len-i-1];

//the min

tt = 1;

while(tmp || vis[tt])

{

if(vis[tt] == 0)

tmp--;

tt++;

}

vis[tt] = 1;

a[i] = tt;

}

}

Dijstra

//v: node id

//l: length from start

//c: mincost

{

int v, l, c;

Node(){}

Node(int v, int l, int c):v(v),l(l),c(c){}

bool operator < (const Node &a) const

//priority\_queue 的优先级和 < 相反

{

if(l == a.l) return c > a.c;

return l > a.l;

}

};

vector<Edge>G[maxn];

priority\_queue<Node>pq;

int dist[maxn],cost[maxn],vis[maxn],tot;

void add\_edge(int u, int v, int l, int c)

{

G[u].push\_back(Edge(u, v, l, c));

}

PII dijstra(int s, int d)

//start s, dest d

{

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

memset(cost, INF, sizeof(cost));

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

while(!pq.empty()) pq.pop();

pq.push(Node(s, 0, 0));

while(!pq.empty())

{

const Node nd = pq.top();

pq.pop();

if(vis[nd.v]) continue;

vis[nd.v] = true;

dist[nd.v] = nd.l;

cost[nd.v] = nd.c;

if(nd.v == d) return make\_pair(dist[d], cost[d]);

for(int i = 0, len = G[nd.v].size(); i < len; i++)

{

Edge& e = G[nd.v][i];

if(!vis[e.v])

{

pq.push(Node(e.v, nd.l + e.l, nd.c+e.c));

}

}

}

//dist[d]: shortest distance

//cost[d]: mincost

return make\_pair(dist[d], cost[d]);

}

Dinic

#define maxn 320

using namespace std;

int G[maxn][maxn], layer[maxn];

int m, n;

bool vis[maxn];

bool countLayer()

{

queue<int>q;

memset(layer, 0xff, sizeof(layer));

layer[1] = 0;q.push(1);

while(!q.empty())

{

int v = q.front();q.pop();

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

if(G[v][j] > 0 && layer[j] == -1)

{

layer[j] = layer[v] + 1;

if(j == n) return true;

else q.push(j);

}

}

return false;

}

int Dinic()

{

int i;

int maxflow = 0;

deque<int> q;

while(countLayer())

{

q.push\_back(1);

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

vis[1] = true;

while(!q.empty())

{

int nd = q.back();

if(nd == n)

{

int minc = 1000000000;

int minstart;

for(i = 1; i < q.size();i++)

{

int vs = q[i-1];

int ve = q[i];

if(G[vs][ve] > 0 && minc > G[vs][ve])

{

minc = G[vs][ve];

minstart = vs;

}

}

maxflow += minc;

for(i = 1; i < q.size(); i++)

{

int vs = q[i-1];

int ve = q[i];

G[vs][ve] -= minc;

G[ve][vs] += minc;

}

while(!q.empty() && q.back() != minstart)

{

vis[q.back()] = false;

q.pop\_back();

}

}else{

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

if(G[nd][i] > 0 && layer[i] == layer[nd] + 1 && !vis[i])

{

vis[i] = true;

q.push\_back(i);

break;

}

if(i > n) q.pop\_back();

}

}

}

return maxflow;

}

int main()

{

while(scanf("%d%d", &m, &n) != EOF)

{

int s, e, c;

memset(G, 0, sizeof(G));

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

{

scanf("%d%d%d",&s,&e,&c);

G[s][e] += c;

}

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

}

return 0;

}

floyed

const int INF=10000000;

int dist[maxn][maxn], G[maxn][maxn];

int n, m, num, minc;

void floyd()

{

minc=INF;

// 求最小环

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

{

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

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

{

int ans=dist[i][j]+G[i][k]+G[k][j];

if(ans<minc) //找到最优解

{

minc=ans;

}

}

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

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

{

if(dist[i][j]>dist[i][k]+dist[k][j])

{

dist[i][j]=dist[i][k]+dist[k][j];

}

}

}

}

Wythoff

//Wythoff Game

//A first

//B second

//当 n 过大时需要用高精度处理，和精确的黄金比例数

int main()

{

int T;

scanf("%d", &T);

while(T--)

{

int a, b;

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

if(a > b) swap(a, b);

int k = b - a;

if(a == (int)((k)\*(1+sqrt(5.0))/2.0)) cout << "B" << endl;

else cout << "A" << endl;

}

return 0;

}

hangary

struct Edge

{

int from,to,weight;

Edge(int f, int t, int w):from(f), to(t), weight(w) {}

};

vector<Edge> G[\_\_maxNodes]; /\* G[i] 存储顶点 i 出发的边的编号 \*/

int matching[\_\_maxNodes]; /\* 存储求解结果 \*/

int check[\_\_maxNodes];

int n, m, sum;

/\*DFS\*/

bool dfs(int u)

{

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

int v = G[u][i].to;

if (!check[v]) { // 要求不在交替路中

check[v] = true; // 放入交替路

if (matching[v] == -1 || dfs(matching[v])) {

// 如果是未盖点，说明交替路为增广路，则交换路径，并返回成功

matching[v] = u;

matching[u] = v;

return true;

}

}

}

return false; // 不存在增广路，返回失败

}

int hungarian()

{

int ans = 0;

memset(matching, -1, sizeof(matching));

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

if (matching[u] == -1) {

memset(check, 0, sizeof(check));

if (dfs(u))

++ans;

}

}

return ans;

}

josephus

//編號從0開始，也就是說如果編號從1開始結果要加1

int josephus(int n, int k) { //非遞回版本

int s = 0;

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

s = (s + k) % i;

return s;

}

int josephus\_recursion(int n, int k) { //遞回版本

return n > 1 ? (josephus\_recursion(n - 1, k) + k) % n : 0;

}

int main() {

for (int i = 1; i <= 100; i++)

cout << i << ' ' << josephus(i, 5) << endl;

return 0;

}

KMP

char src[maxn],substring[maxn];

int nxt[maxn];

void get\_nxt(char\* substring)

{

int substring\_len = strlen(substring);

memset(nxt, 0, sizeof(nxt));

nxt[0] = -1;

int j = -1;

for(int i = 1; i < substring\_len; i++)

{

while(j > -1 && substring[i] != substring[j + 1])

j = nxt[j];

if(substring[j+1] == substring[i])

j = j + 1;

nxt[i] = j;

}

}

//process src & substring to get the position

int kmp(char\* src, char\* substring)

{

int j = -1;

int ans = 0;

int substring\_len = strlen(substring);

int src\_len = strlen(src);

for(int i = 0; i < src\_len; i++)

{

while(j > -1 && src[i] != substring[j + 1])

j = nxt[j];

if(src[i] == substring[j + 1])

j++;

if(j == substring\_len -1)

{

ans ++;

printf("From position %d to position %d\n", i + 2 - substring\_len, i+1);

j = nxt[j];

}

}

return ans;

}

Manacher

const int maxn = 2100000;

/\*

\* 求最长回文字串

\* O(n);

\*/

char Ma[maxn\*2];

int Mp[maxn\*2];

char s[maxn];

void manacher(int len)

{

int l = 0;

Ma[l++] = '$';

Ma[l++] = '#';

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

{

Ma[l++] = s[i];

Ma[l++] = '#';

}

Ma[l] = 0;

int mx = 0, id = 0;

for(int i = 0; i < l; i++)

{

Mp[i] = mx>i? min(Mp[2\*id-i],mx-i):1;

while(Ma[i+Mp[i]] == Ma[i-Mp[i]]) Mp[i]++;

if(i+Mp[i]>mx)

{

mx = i + Mp[i];

id = i;

}

}

}

int main()

{

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

{

scanf("%s", s);

int len = strlen(s);

manacher(len);

int ans = 0;

for(int i = 0; i < len\*2+2; i++)

{

ans = max(ans, Mp[i]-1);

// printf("%d ", Mp[i]);

}

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

}

return 0;

}

Matrix pow

#define maxn 30

using namespace std;

typedef long long LL;

struct Matrix{

LL m[maxn][maxn];

Matrix(){memset(m, 0, sizeof(m));}

};

typedef Matrix matrix;

LL Mod;

int n;

matrix operator\* (matrix A, matrix B)

{

matrix C;

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

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

{

C.m[i][j] = 0LL;

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

C.m[i][j] += A.m[i][k]\*B.m[k][j];

C.m[i][j] %= Mod;

}

return C;

}

matrix operator+ (matrix A, matrix B)

{

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

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

A.m[i][j] = (A.m[i][j] + B.m[i][j]) % Mod;

return A;

}

matrix operator% (matrix A, LL m)

{

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

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

A.m[i][j] %= m;

return A;

}

matrix matrix\_pow(int k, matrix M)

{

if(k == 1) return M;

matrix ans;

memset(ans.m, 0, sizeof(ans.m));

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

ans.m[i][i] = 1LL;

while(k)

{

if(k&1)

{

ans = ans \* M;

k--;

}

else

{

k /= 2;

M = M \* M;

}

}

return ans;

}

matrix sum(matrix ma, int k)

{

matrix ret;

if(k == 1) return ma;

if(k&1)

{

matrix tmp = sum(ma, k/2) % Mod, tmp1 = matrix\_pow(k/2+1, ma) % Mod;

ret = (tmp + tmp1 + tmp \* tmp1) % Mod;

}

else

{

matrix tmp = sum(ma, k/2) % Mod, tmp1 = matrix\_pow(k/2, ma) % Mod;

ret = (tmp + tmp \* tmp1) % Mod;

}

return ret;

}

int main()

{

int k;

matrix A;

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

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

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

scanf("%lld", &A.m[i][j]);

A = sum(A, k);

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

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

{

printf("%lld%c", A.m[i][j],(j == n-1)? '\n':' ');

}

return 0;

}

Math

/\*

\* math templates

\* created by poore : 2015/09/14

\* last edit : 2015/10/19

\*

\* Contents:

\*

\* GCD

\* ext\_GCD

\* 筛法求素数

\* slow\_mul

\* linear\_mod\_equation 一元线性方程组求解

\* pow\_mod

\* Lucas Lehmer 判定梅森素数

\* miller robbin 素数判定

\* pollard rho 返回一个随机的约数

\* calc 寻找最小的约数

\* mega\_mod(n)解 n 个一元线性同于方程组

\* CRT() 中国剩余定理

\* 欧拉函数

\* 整数拆分

\* Stirling's approximation

\*/

#include <cstdio>

#include <iostream>

#include <cmath>

#include <cstring>

#include <cstdlib>

#define INF 0x3f3f3f3f

typedef long long LL;

using namespace std;

const int MOD = 1e9+7;

//GCD

LL GCD(LL a, LL b)

//递归

{

if(a > b) swap(a, b);

LL r = a % b;

if(r == 0) return b;

return GCD(b, r);

}

LL gcd(LL M,LL N)

//非递归

{

LL Rem;

while(N > 0)

{

Rem = M % N;

M = N;

N = Rem;

}

return M;

}

void EXT\_GCD(LL a, LL b, LL &d, LL &x, LL &y)

//a , b 任意

{

if(!b) {d = a, x = 1, y = 0;}

else {EXT\_GCD(b, a % b, d, y, x), y -= x \* (a / b);}

}

//递归求逆元

//p, x 互质

LL inv(LL x, LL m)

{

if (x == 1) return x;

return inv(m % x, m)\*(m - m / x) % m;

}

ll inv(LL a, LL c)

// 用扩展欧几里得求逆元

// 要求 a, c 互质

// 如果没有逆元返回 -1

{

LL d, x, y;

EXT\_GCD(a, c, d, x, y);

return d == 1 ? (x + c) % c : -1;

}

LL ext\_gcd(LL a, LL b, LL& x, LL& y)

// a >= 0, b > 0

{

LL x1=0LL, y1=1LL, x0=1LL, y0=0LL;

LL r = (a%b + b) % b;

LL q = (a-r) / b;

x = 0LL,y = 1LL;

while(r)

{

x=x0-q\*x1;y=y0-q\*y1;

x0=x1;y0=y1;

x1=x;y1=y;

a=b;b=r;

r=a%b;

q=(a-r)/b;

}

return b;

}

const int maxn = 100020;

bool isprime[maxn];

LL prime[maxn];

int doprime(LL N)

//prime[] 储存质数。1-based index;

{

int nprime = 0;

memset(isprime, true, sizeof(isprime));

isprime[1] = false;

for(LL i = 2; i <= N; i++)

{

if(isprime[i])

{

prime[++nprime] = i;

for(LL j = i\*i; j <= N; j+=i)

isprime[j] = false;

}

}

return nprime;

}

LL slow\_mul(LL a, LL b, LL p)

{

// cout << a << " " << b << endl;

LL ret = 0;

while(b) {

if(b & 1) ret = (ret + a) % p;

a = (a + a) % p;

b >>= 1;

}

return ret % p;

}

LL pow\_mod(LL a, LL b, LL p)

//快速幂

{

LL ret = 1;

while(b) {

if(b & 1) ret = (ret\*a)%p;

a = (a\*a)%p;

b >>= 1;

}

return ret%p;

}

//判断Mp = 2^p-1 是否为梅森素数

bool lucas\_lehmer(int p)

{

if(p == 2) return true;

LL m = (1LL<<p)-1LL, tmp = 4LL;

for(int i = 0; i < p-2; i++)

{

tmp = (slow\_mul(tmp, tmp, m) - 2 + m) % m;

}

if(tmp == 0LL) return true;

return false;

}

LL witness(LL a,LL b,LL c)

{

if(b==0)return 1;

LL x,y,t=0;

while((b&1)==0)

b>>=1,t++;

y=x=pow\_mod(a,b,c);

//二次探测

while(t--)

{

y=slow\_mul(x,x,c);

if(y==1 && x!=1 && x!=c-1)

return false;

x=y;

}

return y==1;

}

bool miller\_rabin(LL n)

//..质数为true, 非质数为false..

{

if(n==2)return true;

if(n<2 || (n&1)==0)return false;

for(int i=0;i<3;i++)

if(witness(rand()%(n-2)+2,n-1,n)!=1)

return false;

return true;

}

LL ans = INF;

LL pollard\_rho(LL n,LL c)

//..随机返回一个 n 的约数..

{

if(n%2==0)return 2;

LL i=1,k=2,x=rand()%n,y=x,d;

while(1){

i++;

x=(slow\_mul(x,x,n)+c)%n;

d=gcd(y-x,n);

if(d==n)return n;

if(d!=n && d>1)return d;

if(i==k) y=x,k<<=1;

}

}

void calc(LL n,LL c=240)

//寻找最小的约数..

{

if(n==1)return;

if(miller\_rabin(n)){

ans=min(ans,n);

return;

}

LL k=n;

while(k==n)k=pollard\_rho(n,c--);

calc(k,c),calc(n/k,c);

}

vector<LL> linear\_mod\_equation(LL a, LL b, LL n)

//线性方程求解

//ax = b (mod n)

{

LL x, y, d;

vector<LL> sol;

sol.clear();

EXT\_GCD(a, n, d, x, y);

if( b%d ) d = 0;

else

{

sol.push\_back(x \* (b/d) % n);

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

sol.push\_back((sol[i-1] + n/d + n) % n);

}

return sol;

}

LL mega\_mod(int n)

//解 n 个一元线性同于方程组

//x ≡ r (mod a)

//求x

{

LL a1, a2, r1, r2, d, c, x, y, x0,s;

bool flag = true;

scanf("%lld%lld", &a1, &r1);

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

{

scanf("%lld%lld", &a2, &r2);

if(!flag) continue;

c = r2 - r1;

EXT\_GCD(a1, a2, d, x, y);

if(c%d!=0)

{

flag = false;

continue;

}

x0 = x\*c/d;

s = a2/d;

x0 = (x0%s+s)%s;

r1=r1+x0\*a1;

a1=a1\*a2/d;

}

if(flag) return r1;

else return -1LL;

}

LL CRT(LL \*a, LL \*m, int n)

//中国剩余定理

//x ≡ a[i] (mod m[i])

//m[i] is coprime

{

LL M = 1, Mi, x0, y0, d, ret = 0;

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

M \*= m[i];

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

{

Mi = M/m[i];

EXT\_GCD(Mi, m[i], d, x0, y0);

ret = (ret+Mi\*x0\*a[i]) % M;

}

if(ret < 0)

ret += M;

return ret;

}

//欧拉函数

LL calphi(LL n)

{

LL res = n;

for(LL i = 2; i\*i <= n; i++)if(n%i==0)

{

res -= res/i;

while(n%i==0) n/=i;

}

if(n > 1)

res -= res/n;

return res;

}

//欧拉函数预处理

int phi[maxn];

void getpthi(int n)

{

memset(phi, 0, sizeof(phi));

phi[1] = 1;

for(int i = 2; i <= n; i++)if(!phi[i])

{

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

{

if(!phi[j])

phi[j] = j;

phi[j] = phi[j]/i\*(i-1);

}

}

}

//把整数 n 拆分成几个数相加的形式， 问有多少种拆分方法

int dp[maxn];

void splitint()

{

memset(dp, 0, sizeof(dp));

dp[0]=1;

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

{

for(int j = 1, r = 1; i - (3\*j\*j-j)/2 >= 0; j++, r\*=-1)

{

dp[i] += dp[i-(3\*j\*j-j)/2]\*r;

dp[i] %= MOD;

dp[i] = (dp[i]+MOD)%MOD;

if(i-(3\*j\*j+j)/2 >= 0)

{

dp[i] += dp[i-(3\*j\*j+j)/2] \*r;

dp[i] %= MOD;

dp[i] = (dp[i] + MOD)%MOD;

}

}

}

}

//Stirling N的阶乘的长度

const double PI=3.1415926;

int main()

{

int t,n,a;

while(scanf("%d",&n)!=EOF)

{

a=(int)((0.5\*log(2\*PI\*n)+n\*log(n)-n)/log(10));

printf("%d\n",a+1);

}

return 0;

}

/\*

Something Tasteless

1.素数个数估算

设PI(x) 为小于 x 的素数的个数

当 x 足够大时，PI(x) = x/lnx;

2.n! 的素因子分解中的素数 p 的次数 为

[n/p] + [n/(p^2)] + [n/(p^3)] + ... +

3.

\*/