**Standard Code Library**

**Barty**

Content

[AC自动机+多串匹配 3](#_Toc337645058)

[catalan数列求和 4](#_Toc337645059)

[N皇后构造 5](#_Toc337645060)

[布斯乘法 7](#_Toc337645061)

[带权并查集+路径压缩 7](#_Toc337645062)

[笛卡尔树建立 9](#_Toc337645063)

[点双联通缩点 10](#_Toc337645064)

[划分树 13](#_Toc337645065)

[离散化+映射二叉堆 15](#_Toc337645066)

[罗马数字 16](#_Toc337645067)

[螺旋矩阵求坐标或值 17](#_Toc337645068)

[枚举子集 18](#_Toc337645069)

[全排列求后继 18](#_Toc337645070)

[三维凸包求重心 19](#_Toc337645071)

[筛法求欧拉函数 23](#_Toc337645072)

[实数高斯消元 24](#_Toc337645073)

[求逆元 25](#_Toc337645074)

[循环字符串的最小表示 26](#_Toc337645075)

[自动机+状压dp 26](#_Toc337645076)

[LCA 28](#_Toc337645077)

### AC自动机+多串匹配

//已通过hdu2222

struct Aho\_Corasick {

int ch[maxn][26], d[maxn], sz, fail[maxn], q[maxn], l, r;

void init() {

sz = 1;

memset(ch[0], 0, sizeof(ch[0]));

}

void insert(char \*str) {

int p = 0;

for (; \*str; ++str) {

int x = (\*str) - 'a';

if (!ch[p][x]) {

ch[p][x] = sz++;

memset(ch[sz-1], 0, sizeof(ch[sz-1]));

fail[sz-1] = 0;

d[sz-1] = 0;

}

p = ch[p][x];

}

++d[p];//同一个模式串可能出现多次

}

void build\_automation() {

clr(q); l = 1; r = 0;

for (int i = 0; i < 26; ++i)

if (ch[0][i]) q[++r] = ch[0][i];

while (l <= r) {

int now = q[l++];

for (int i = 0; i < 26; ++i)

if (ch[now][i]) {

q[++r]= ch[now][i];

fail[ch[now][i]] = ch[fail[now]][i];

} else ch[now][i] = ch[fail[now]][i];

}

}

int count(char \*str) {

int p = 0, ret = 0, q;

for (; \*str; ++str) {

while (!ch[p][(\*str)-'a'] && p) p = fail[p];

if (ch[p][(\*str)-'a']) p = ch[p][(\*str)-'a'];

q = p;

while (q && ~ d[q]) {

ret += d[q];

d[q] = -1;//同一个模式串只计算一次

q = fail[q];

}

}

return ret;

}

}ac;

### catalan数列求和

#include <iostream>

#include <cstring>

#include <cstdio>

using namespace std;

#define N 110

typedef long long lld;

void extgcd(int a, int b, int & x, int & y)

{

if (b == 0) { x = 1; y = 0; return; }

extgcd(b, a % b, x, y);

int t = x; x = y; y = t - a / b \* y;

return;

}

lld catalan\_sum(int n, int m)

{

int sm[N],sa[N];

lld sum, res;

int k = m, p = 0;

n -= 2;

sum = res = 1LL;

p = 0;

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

if(k % i == 0)

{

sm[p++] = i;

do

{

k /= i;

}while (k % i == 0);

}

if (k > 1) sm[p++] = k;

memset(sa, 0, sizeof(sa));

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

{

int t = 4 \* i - 2;

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

{

while (t % sm[j] == 0)

{

sa[j]++;

t /= sm[j];

}

}

res = res \* t % m;

t = i + 1;

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

{

while (t % sm[j] == 0)

{

sa[j]--;

t /= sm[j];

}

}

if(t > 1)

{

int x, y;

extgcd(t, m, x, y);

x = (x % m + m) % m;

res = (res \* x) % m;

}

lld fin = res;

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

for(int k = 0; k < sa[j]; k++)

fin = (fin \* sm[j]) % m;

sum = (sum + fin) % m;

}

return sum;

}

### N皇后构造

#include <iostream>

#include <cstdio>

#include <cstring>

using namespace std;

int main()

{

int n, k;

while (scanf("%d", &n), n)

{

if ((n % 6 != 2) && (n % 6 != 3))

{

if (n & 1)

{

for (int i = 2; i <= n - 1; i += 2) printf("%d ", i);

for (int i = 1; i < n; i += 2) printf("%d ", i);

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

}

else

{

for (int i = 2; i <= n; i += 2) printf("%d ", i);

for (int i = 1; i < n - 1; i += 2) printf("%d ", i);

printf("%d\n", n - 1);

}

}

else

{

if (n & 1) k = (n - 1) / 2;

else k = n / 2;

if (!(k & 1) && !(n & 1))

{

for (int i = k; i <= n; i += 2) printf("%d ", i);

for (int i = 2; i <= k - 2; i += 2) printf("%d ", i);

for (int i = k + 3; i <= n - 1; i += 2) printf("%d ", i);

for (int i = 1; i < k + 1; i += 2) printf("%d ", i);

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

}

else if (!(k & 1) && (n & 1))

{

for (int i = k; i <= n - 1; i += 2) printf("%d ", i);

for (int i = 2; i <= k - 2; i += 2) printf("%d ", i);

for (int i = k + 3; i <= n - 2; i += 2) printf("%d ", i);

for (int i = 1; i <= k + 1; i += 2) printf("%d ", i);

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

}

else if ((k & 1) && !(n & 1))

{

for (int i = k; i <= n - 1; i += 2) printf("%d ", i);

for (int i = 1; i <= k - 2; i += 2) printf("%d ", i);

for (int i = k + 3; i <= n; i += 2) printf("%d ", i);

for (int i = 2; i < k + 1; i += 2) printf("%d ", i);

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

}

else

{

for (int i = k; i <= n - 2; i += 2) printf("%d ", i);

for (int i = 1; i <= k - 2; i += 2) printf("%d ", i);

for (int i = k + 3; i <= n - 1; i += 2) printf("%d ", i);

for (int i = 2; i <= k + 1; i += 2) printf("%d ", i);

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

}

}

}

return 0;

}

### 布斯乘法

LL mul(LL lhs, LL rhs){//布斯乘法

LL sq = 100000;

LL lhs1 = lhs % sq;

LL rhs1 = rhs % sq;

return ((lhs / sq \* rhs1 + rhs / sq \* lhs1) \* sq + lhs1 \* rhs1) % module;

}

### 带权并查集+路径压缩

/\*

读入n,m

m行s,t,d,表示sum[s..t]=d,询问是否合法，如果合法则加入

输出每个位置的答案

\*/

#include <iostream>

#include <cstdio>

#include <cstring>

#define LL long long

using namespace std;

struct node

{

LL fa, l;

}d[100010];

LL n, m;

node father(LL v)

{

node now = d[v];

LL sum = 0, root = v;

while (d[root].fa != root)

{

sum += d[root].l;

root = d[root].fa;

}

LL i = v;

while (i != root)

{

LL ni = i;

LL ts = d[ni].l;

d[ni].l = sum;

sum -= ts;

i = d[i].fa;

d[ni].fa = root;

}

return d[v];

}

int main()

{

while (~scanf("%lld%lld", &n, &m))

{

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

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

d[i].fa = i;

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

{

LL x, y, z;

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

y++;

node fx = father(x);

node fy = father(y);

if (fx.fa != fy.fa)

{

d[fx.fa].fa = fy.fa;

d[fx.fa].l = z + fy.l - fx.l;

printf("Accepted!\n");

}

else

{

LL res = fx.l - fy.l;

if (z != res) printf("Error!\n");

else printf("Accepted!\n");

}

}

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

{

node fx = father(i);

node fy = father(i + 1);

if (fx.fa != fy.fa)

{

printf("Unknown!\n");

}

else

{

LL res = fx.l - fy.l;

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

}

}

}

return 0;

}

### 笛卡尔树建立

int n;

struct Node{

int mkey, akey;//mkey满足排序性，akey满足堆性

bool operator < (Node b) const {

return mkey < b.mkey;

}

} a[maxn];

int stack[maxn];

int fa[maxn], ch[maxn][2];

void Build\_Cartesian\_Tree() {

int i, k, top = 0;

stack[0] = 0; // As the father of root;

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

k = top;

while( k > 0 && a[stack[k]].akey > a[i].akey ) --k;

if( top == k ){

fa[i]= stack[k];

stack[++k]= i;

}

else if( top > k ){

fa[ stack[k+1] ] = i;

fa[i] = stack[k];

stack[++k] = i;

}

top = k;

}

for(i = 1; i <= n; ++i) ch[i][0] = ch[i][1] = 0;

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

if( a[ fa[i] ].mkey > a[i].mkey ) ch[ fa[i] ][0] = i;

else ch[ fa[i] ][1] = i;

}

}

### 点双联通缩点

#include <iostream>

#include <cstdio>

#include <cstring>

#include <vector>

#define maxn 300

#define clr(x) memset(x, 0, sizeof(x))

using namespace std;

struct edge {

int v, d, next;

}e[maxn\*maxn];

int p[maxn], eid;

void mapinit() {

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

eid = 0;

}

void insert(int x, int y) {

e[eid].v = y;

e[eid].next = p[x];

p[x] = eid++;

}

void checkmin(int &x, int y) {

if (y < x) x = y;

}

int dfn[maxn], low[maxn], stack[maxn], top, dindex, circnt, n, belong[maxn];

vector<int> cir[maxn];

bool iscut[maxn];

int rootcnt;

void dfs(int v, int last) {

dfn[v] = low[v] = ++dindex;

stack[++top] = v;

for (int i = p[v]; i != -1; i = e[i].next) {

if (e[i].v == last) continue;

if (dfn[e[i].v]) checkmin(low[v], dfn[e[i].v]);

else {

/\* if (last == -1) {

if (++rootcnt > 1) iscut[v] = 1;

} else iscut[v] = 1;

GG: 这段你在干什么，这样的话，所有点都是割点？

\*/

dfs(e[i].v, v);

checkmin(low[v], low[e[i].v]);

if (low[e[i].v] >= dfn[v]) {

//GG add below

iscut[v] = 1;

//GG add above

++circnt;

int t;

vector<int> now; now.clear();

do {

t = stack[top--];

now.push\_back(t);

belong[t] = circnt;

} while (t != e[i].v);

now.push\_back(v);/\*

cout << " " << now.size() << ": ";

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

cout << now[k] << " ";

} cout << endl;\*/

cir[circnt] = now;

belong[v] = circnt;

}

}

}

}

void solve() {

clr(belong); clr(dfn); clr(low); dindex = 0; top = 0; circnt = 0;

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

if (!dfn[i]) {

rootcnt = 0;

dfs(i, -1);

}

}

void print\_map() {

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

printf("%d: ", i);

for (int j = p[i]; j != -1; j = e[j].next)

printf("%d,", e[j].v);

puts("");

}

}

int m, s, d;

int g[maxn][maxn], hasht[maxn], w[maxn], f[maxn];

bool vst[maxn];

void dfs2(int v, int l) {

vst[v] = 1;

f[v] = l;

for (int i = p[v]; i != -1; i = e[i].next)

if (!vst[e[i].v]) {

dfs2(e[i].v, l + w[e[i].v]);

}

}

int main() {

while (scanf("%d%d%d%d", &n, &m, &s, &d) == 4) {

int orin = n;

clr(g);

clr(iscut); // GG ADDED

++s; ++d;

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

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

g[x+1][y+1] = g[y+1][x+1] = 1;

}

if (s == d) {

printf("%d\n", n - 1);

continue;

}

mapinit();

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

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

if (g[i][j]) insert(i, j);

solve();

int now = 0;

clr(g); clr(w);

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

if (iscut[i]) hasht[i] = circnt + (++now);

w[hasht[i]] = -1;

}

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

w[i] = cir[i].size();

int canReach = 0; /\* GG added \*/

if (iscut[s]) s = hasht[s], /\*GG added\*/canReach++;

else s = belong[s];

if (iscut[d]) d = hasht[d], /\*GG added\*/canReach++;

else d = belong[d];

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

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

if (iscut[cir[i][j]]) g[i][hasht[cir[i][j]]] = g[hasht[cir[i][j]]][i] = 1;

mapinit();

n = circnt + now;

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

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

if (g[i][j]) insert(i, j);

clr(vst); clr(f); dfs2(s, w[s]);

canReach += f[d];//GG added

cout << orin - canReach << endl;

//printf("%d\n", orin - f[d]); GG deleted

}

return 0;

}

### 划分树

#include<cstdio>

#include<string>

#include<vector>

#include<algorithm>

#define N 100009

using namespace std;

int n, arr[N], od[N], lfnum[20][N], val[20][N];

bool cmp(const int &x, const int &y){ return arr[x] < arr[y]; }

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

if(l == r) return;

int mid = (l + r) >> 1, p = 0;

for(int i = l; i <= r; i++) {

if(val[d][i] <= mid) {

val[d + 1][l + p] = val[d][i];

lfnum[d][i] = ++p;

}

else {

lfnum[d][i] = p;

val[d + 1][mid + i + 1 - l - p] = val[d][i];

}

}

build(l, mid, d + 1);

build(mid + 1, r, d + 1);

}

int query(int s, int e, int k, int l = 1, int r = n, int d = 0) {

if(l == r) return l;

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

ss = (s == l ? 0 : lfnum[d][s - 1]);

ee = lfnum[d][e];

if(ee - ss >= k) return query(l + ss, l + ee - 1, k, l, mid, d + 1);

return query(mid + 1 + (s - l - ss), mid + 1 + (e - l - ee), k - (ee - ss), mid + 1, r, d + 1);

}

int main() {

int ca = 0, m, l, r;

while(scanf("%d", &n) == 1) {

printf("Case %d:\n", ++ca);

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

sort(od + 1, od + n + 1, cmp);

for(int i = 1; i <= n; i++) val[0][od[i]] = i;

build(1, n, 0);

scanf("%d", &m);

while(m--) {

int num, k;

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

k = (r - l) / 2 + 1;

num = query(l, r, k);

int ans = arr[od[num]];

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

}

}

return 0;

}

### 离散化+映射二叉堆

#include <iostream>

#include <cstdio>

#include <cstring>

#include <set>

#include <algorithm>

#define maxn 200010

#define PII pair<int, int>

#define mpp make\_pair

#define inf 1000000000

using namespace std;

int tsize, table[maxn\*2];

void init\_table() {

sort(table, table+tsize);

tsize = unique(table, table+tsize)-table;

}

int get\_index(int key) {

return lower\_bound(table, table+tsize, key)-table;

}

int n, m, a[maxn\*2];

int pos[maxn\*2], next[maxn\*2];

bool inset[maxn\*2];

int main() {

while (scanf("%d%d", &n, &m) == 2) {

tsize = 0;

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

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

table[tsize++] = a[i];

}

for (int i = 1; i <= m; ++i) table[tsize++] = i;

init\_table();

for (int i = 1; i <= n; ++i) a[i] = get\_index(a[i]);

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

pos[i] = inf;

inset[i] = 0;

}

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

next[i] = pos[a[i]];

pos[a[i]] = i;

}

m = min(m, tsize);

set< PII, greater< PII > > s;

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

s.insert(mpp(pos[i], i));

inset[i] = true;

}

int ans = 0;

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

if (inset[a[i]]) {

s.erase(mpp(i, a[i]));

s.insert(mpp(next[i], a[i]));

} else {

ans++;

inset[a[i]] = 1;

s.insert(mpp(next[i], a[i]));

inset[(\*s.begin()).second] = 0;

s.erase(\*s.begin());

}

}

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

}

return 0;

}

### 罗马数字

#include <iostream>

#include <cstdio>

#include <cstring>

#include <algorithm>

using namespace std;

string name[10010];

string thousands[4] = {"", "M", "MM", "MMM"};

string hundreds[10] = {"", "C", "CC", "CCC", "CD", "D", "DC", "DCC","DCCC","CM"};

string tens[10] = {"", "X", "XX","XXX","XL","L","LX","LXX","LXXX","XC"};

string ones[10] = {"", "I","II","III","IV","V","VI","VII","VIII","IX"};

struct data {

string name;

int x;

bool operator < (const data &a) const {

return name < a.name;

}

}d[10010];

string getname(int x) {

return thousands[x/1000] + hundreds[x/100%10] + tens[x/10%10] + ones[x%10];

}

int main () {

int re;

scanf("%d", &re);

while (re--) {

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

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

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

d[i].name = getname(x);

d[i].x = x;

}

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

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

printf("%d%c", d[i].x, " \n"[i==n]);

}

return 0;

}

### 螺旋矩阵求坐标或值

po getpos(LL n) {

if (n == 1) return po(0, 0);

LL k = 1;

while (n > sqr(2\*k-1)) ++k;

k--;

LL now = sqr(2\*k-1);

if (n <= now + 2 \* k) {

return po(k, -k+n-now);

} else if (n <= now + 4 \* k) {

return po(k-(n-now-2\*k), k);

} else if (n <= now + 6 \* k) {

return po(-k, k-(n-now-4\*k));

} else {

return po(-k+n-now-6\*k, -k);

}

}

LL getdat(po p) {

if (p.x==0&&p.y==0) return 1;

LL k = max(iabs(p.x), iabs(p.y));

LL now = sqr(2\*k-1);

if (p.x==k&&p.y!=-k) {

return now + p.y+k;

} else if (p.y==k) {

return now + 2\*k + k-p.x;

} else if (p.x==-k) {

return now + 4\*k + k-p.y;

} else {

return now + 6\*k + p.x+k;

}

}

### 枚举子集

for (int i = 1; i < p2[n]; ++i) {

int x = s[(p2[n]-1)^i], y, z;

if (x > s[i]) continue;

for (int j = i&(i-1); j; j = (j - 1) & i) {

y = s[j], z = s[j^i];

if (z >= y && y >= x && x + y > z)

mm.insert(make\_pair(x, y));

}

}

### 全排列求后继

bool next(vector<int> &now)

{

int cnt = now.size(), a = -1, b, t = INT\_MAX;

for (int i = 0; i + 1 < cnt; ++i)

if (now[i] < now[i + 1]) a = i;

if (a == -1)

{

return false;

}

for (int i = a + 1; i < cnt; ++i)

if ((now[i] > now[a]) && (now[i] < t))

{

t = now[i];

b = i;

}

swap(now[a], now[b]);

for (int i = a + 1; a + cnt - i > i; ++i)

swap(now[i], now[a + cnt - i]);

return true;

}

### 三维凸包求重心

#include <cstdio>

#include <cstring>

#include <cstdlib>

#include <iostream>

#include <algorithm>

#include <vector>

#include <cmath>

using namespace std;

#define SIZE(a) ((int)(a).size())

const double eps = 1e-8;

const double pi = acos(-1.0);

inline int Sign(double x)

{

if (x<-eps) return -1;

return x>eps;

}

inline double sqr(double x)

{

return x\*x;

}

struct Point

{

double x,y,z;

Point(){x=y=z=0;}

Point(double x,double y,double z):x(x),y(y),z(z){}

inline double norm()

{

return x\*x+y\*y+z\*z;

}

inline double length()

{

return sqrt(norm());

}

inline void read()

{

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

}

};

inline Point operator +(const Point &a,const Point &b){return Point(a.x+b.x,a.y+b.y,a.z+b.z);}

inline Point operator -(const Point &a,const Point &b){return Point(a.x-b.x,a.y-b.y,a.z-b.z);}

inline bool operator <(const Point &a,const Point &b){return Sign(a.x-b.x)<0 || Sign(a.x-b.x)==0 && Sign(a.y-b.y)<0 || Sign(a.x-b.x)==0 && Sign(a.y-b.y)==0 && Sign(a.z-b.z)<0;}

inline bool operator ==(const Point &a,const Point &b){return Sign(a.x-b.x)==0 && Sign(a.y-b.y)==0 && Sign(a.z-b.z)==0;}

inline Point operator \*(const Point &a,const double &b){return Point(a.x\*b,a.y\*b,a.z\*b);}

inline Point operator /(const Point &a,const double &b){return Point(a.x/b,a.y/b,a.z/b);}

inline Point det(const Point &a,const Point &b)

{

return Point(a.y\*b.z-a.z\*b.y, -( a.x\*b.z-a.z\*b.x ),a.x\*b.y-a.y\*b.x);

}

inline double dot(const Point &a,const Point &b)

{

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

}

//====================================================

int mark[1005][1005];

Point info[1005];

int n,cnt;

double mix(const Point &a,const Point &b,const Point &c)

{

return dot(a,det(b,c));

}

double area(int a,int b,int c)

{

return (info[b]-info[a],info[c]-info[a]).length();

}

double volume(int a,int b,int c,int d)

{

return mix(info[b]-info[a],info[c]-info[a],info[d]-info[a]);

}

struct Face

{

int a,b,c;

Face(){}

Face(int a,int b,int c):a(a),b(b),c(c){}

int& operator [](int k){

if (k==0) return a;

if (k==1) return b;

return c;

}

};

vector<Face> face;

inline void insert(int a,int b,int c)

{

face.push\_back(Face(a,b,c));

}

inline void add(int v)

{

vector<Face> tmp;

int a,b,c;

++cnt;

for (int i=0;i<SIZE(face);++i){

a=face[i][0];

b=face[i][1];

c=face[i][2];

if (Sign(volume(v,a,b,c))<0)

mark[a][b]=mark[b][a]=mark[b][c]=mark[c][b]=mark[c][a]=mark[a][c]=cnt;

else

tmp.push\_back(face[i]);

}

face=tmp;

for (int i=0;i<SIZE(tmp);++i){

a=face[i][0];

b=face[i][1];

c=face[i][2];

if (mark[a][b]==cnt) insert(b,a,v);

if (mark[b][c]==cnt) insert(c,b,v);

if (mark[c][a]==cnt) insert(a,c,v);

}

}

inline int Find()

{

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

Point ndir=det(info[0]-info[i],info[1]-info[i]);

if (ndir==Point()) continue;

swap(info[i],info[2]);

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

if (Sign(volume(0,1,2,j))!=0){

swap(info[j],info[3]);

insert(0,1,2);

insert(0,2,1);

return 1;

}

}

}

return 0;

}

int main()

{

for (;scanf("%d",&n)==1;){

for (int i=0;i<n;++i) info[i].read();

sort(info,info+n);

n=unique(info,info+n)-info;

face.clear();

random\_shuffle(info,info+n);

if (Find()){

memset(mark,0,sizeof(mark));

cnt=0;

for (int i=3;i<n;++i) add(i);

Point ans(0,0,0),o=info[0];

double total=0;

for (int i=0;i<SIZE(face);++i){

double volume=fabs(mix(info[face[i][0]]-o,info[face[i][1]]-o,info[face[i][2]]-o));

total+=volume;

ans=ans+(o+info[face[i][0]]+info[face[i][1]]+info[face[i][2]])/4.0\*volume;

}

ans=ans/total;

double len=(ans-info[0]).length();

for (int i=0;i<SIZE(face);++i){

Point ndir=det(info[face[i][1]]-info[face[i][0]],info[face[i][2]]-info[face[i][0]]);

len=min(len,fabs(dot(ans-info[face[i][0]],ndir))/ndir.length());

}

printf("%.3f\n",len);

}

}

return 0;

}

### 筛法求欧拉函数

int phi[maxn];

bool isp[maxn];

int prime[maxn/10];

void eulerfunc() {

memset(isp, 1, sizeof(isp));

phi[1] = 1;

for (LL i = 2, k = 0; i <= maxn; ++i) {

if (!phi[i]) {

phi[i] = i-1;

prime[k++] = i;

}

for (LL j = 0; j < k && i\*prime[j]<=maxn; ++j)

if (i%prime[j]) phi[i\*prime[j]] = phi[i]\*(prime[j]-1);

else {

phi[i\*prime[j]] = phi[i] \* prime[j];

break;

}

}

}

### 实数高斯消元

namespace gauss {//高斯消元double模板

const double eps = 1e-12;

inline int sgn(double x) { return (x > eps) - (x < -eps); }

int gauss(double a[12][12], int n, double ans[12]) {//a矩阵为系数矩阵(0下标开始)，ans为计算结果

int i, j, k, max\_r, col, free\_x\_num, free\_index, equ = n, var = n;//equ为等式个数，var为变量个数

double temp;

bool free\_x[12];

col = 0;

memset(free\_x, 1, sizeof(free\_x));

for (k = 0; k < equ && col < var; k++, col++) {

max\_r = k;

for (i = k + 1; i < equ; i++) {

if (sgn(fabs(a[i][col]) - fabs(a[max\_r][col]))>0) max\_r = i;

}

if (max\_r != k) {

for (j = k; j < var + 1; j++) swap(a[k][j], a[max\_r][j]);

}

if (sgn(a[k][col]) == 0 ) {

k--; continue;

}

for (i = k + 1; i < equ; i++) {

if (sgn(a[i][col]) != 0) {

double t = a[i][col] / a[k][col];

for (j = col; j < var + 1; j++) {

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

}

}

}

}

if (k < var) {

for (i = k - 1; i >= 0; i--) {

free\_x\_num = 0;

for (j = 0; j < var; j++) {

if ( sgn(a[i][j]) != 0 && free\_x[j]) {

free\_x\_num++, free\_index = j;

}

}

if(free\_x\_num > 1) continue;

temp = a[i][var];

for (j = 0; j < var; j++) {

if (sgn(a[i][j]) != 0 && j != free\_index) temp -= a[i][j] \* ans[j];

}

ans[free\_index] = temp / a[i][free\_index];

free\_x[free\_index] = 0;

}

return var - k;

}

for (i = var - 1; i >= 0; i--) {

temp = a[i][var];

for (j = i + 1; j < var; j++) {

if (sgn(a[i][j]) != 0) temp -= a[i][j] \* ans[j];

}

ans[i] = temp / a[i][i];

}

return 0;

}

}

### 求逆元

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

if (a == 0 && b == 0) return -1;

if (b == 0) {

x = 1; y = 0; return a;

}

LL d = ext\_gcd(b, a%b, y, x);

y -= a / b \* x;

return d;

}

//ax = 1(mod n)

LL mod\_reverse(LL a, LL n) {

LL x, y, d = ext\_gcd(a, n, x, y);

if (d == 1) return (x % n + n) % n;

else return -1;

}

### 循环字符串的最小表示

int MinCircularDenote(const char\* s, int len) {

int p1 = 0, p2 = 1, k, t1, t2;

while (1) {

k = 0;

while (1) {

t1 = (p1+k)%len; t2 = (p2+k)%len;

if(s[t1] > s[t2]) {

if (p1+k+1 <= p2) p1 = p2+1; // optimize

else p1 = p1+k+1;

if (p1 >= len) return p2; // p1 has checked len, return p2

break;

}

else if (s[t1] < s[t2]) {

if (p2+k+1 <= p1) p2 = p1+1;

else p2 = p2+k+1;

if (p2 >= len) return p1;

break;

}

else k++;

if (k == len) // has macthed len，return Min pos

return (p1<p2 ? p1 : p2);

}

}

}

### 自动机+状压dp

/\*

2011大连现场赛原题

\*/

#include <iostream>

#include <cstdio>

#include <cstring>

#define maxn 1050

using namespace std;

int ch[maxn][4], d[maxn], sz, fail[maxn];

char ord[256];

void insert(char \*str, int val) {

int p = 0;

for (; \*str; ++str) {

if (!ch[p][ord[\*str]]) {

ch[p][ord[\*str]] = sz++;

}

p = ch[p][ord[\*str]];

}

d[p] |= (1<<val);

}

int q[maxn];

void ac() {

memset(q, 0, sizeof(q));

int l = 1, r = 0;

for (int i = 0; i < 4; ++i)

if (ch[0][i]) q[++r] = ch[0][i];

while (l <= r) {

int now = q[l++];

for (int i = 0; i < 4; ++i)

if (ch[now][i]) {

q[++r] = ch[now][i];

fail[ch[now][i]] = ch[fail[now]][i];

d[ch[now][i]] |= d[ch[fail[now]][i]];

} else ch[now][i] = ch[fail[now]][i];

}

}

char str[100010];

int n, l, f[110], p2[11], s[maxn];

bool dp[2][maxn][maxn];

int main() {

p2[0] = 1;

for (int i = 1; i <= 10; ++i) p2[i] = p2[i-1]<<1;

ord['A'] = 0; ord['T'] = 1; ord['C'] = 2; ord['G'] = 3;

while (scanf("%d%d", &n, &l) == 2) {

memset(ch, 0, sizeof(ch));

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

memset(fail, 0, sizeof(fail));

sz = 1;

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

scanf("%s%d", str, &f[i]);

insert(str, i);

}

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

for (int i = 0; i < p2[n]; ++i)

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

if (i&p2[j]) s[i] += f[j];

ac();

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

int now = 0, next = 1;

dp[now][0][0] = 1;

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

memset(dp[next], 0, sizeof(dp[next]));

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

for (int k = 0; k < p2[n]; ++k)

if (dp[now][j][k]) {

for (int t = 0; t < 4; ++t)

dp[next][ch[j][t]][k|d[ch[j][t]]] = 1;

}

swap(now, next);

}

int ans = -1;

for (int i = 0; i < p2[n]; ++i)

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

if (dp[now][j][i]) ans = max(ans, s[i]);

if (ans >= 0) printf("%d\n", ans);

else printf("No Rabbit after 2012!\n");

}

return 0;

}

### LCA

struct LCA {

struct edge { int v, next; }e[M];

int p[N], eid, n, fa[N][LOGN], deep[N];

bool vst[N], isrt[N];

void init() { clx(p); eid = 0; clr(vst); clr1(isrt); }

void insert(int x, int y) {

e[eid].v = y; e[eid].next = p[x]; p[x] = eid++;

}

void insert2(int x, int y) {

insert(x, y); insert(y, x);

}

void build(int now, int dp) {

deep[now] = dp; vst[now] = 1;

for (int i = p[now]; i != -1; i = e[i].next)

if (!vst[e[i].v]) {

fa[e[i].v][0] = now;

build(e[i].v, dp+1);

}

}

void read() {

int x, y; init();

scanf("%d", &n);

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

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

insert(x, y); isrt[y] = 0;

}

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

if (isrt[i]) {

build(i, 0); fa[i][0] = i;

}

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

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

fa[j][i] = fa[fa[j][i-1]][i-1];

}

void gao(int &x, int dp) {

int j = LOGN - 1;

while (deep[x] != dp) {

while (deep[fa[x][j]] < dp) --j;

x = fa[x][j];

}

}

int query(int x, int y) {

if (deep[x] < deep[y]) swap(x, y);

if (deep[x] != deep[y]) gao(x, deep[y]);

if (x == y) return x;

int j = LOGN - 1;

while (fa[x][0] != fa[y][0]) {

while (fa[x][j] == fa[y][j]) --j;

x = fa[x][j]; y = fa[y][j];

}

return fa[x][0];

}

};