目录

1. MATH 2
   a) euler function 2
   b) exgcd 2
   c) fermat's little 2
   d) Chinese remainder 3
   e) fft 4
   f) else 6
2. GRATH 6
   a) Prim 6
   b) Kruskal 7
   c) Dijstra 7
   d) Spfa 8
   e) Two-sat 9
   f) MCMF 10
   g) Max-flow 12
   h) LCA 13
   i) Hungarian-dfs 15
3. DATA STRUCTURE 16
   a) Kd-tree 16
   b) Tree array 17
   c) Sa array 18
4. GEOMETRY 19
5. ALGORITHM 21
   a) RMQ 21
   b) Manacher 21
   c) Ac automatic 22
   d) Kmp 24
6. STL 25

1. MATH

a) euler function

```
int euler(int x){
    int ans = x, m = n;
    for(int i = 2; i * i <= x; i++){
        if(x % i == 0){
            ans = ans / i * (i - 1);
            while(x % i == 0){
                x /= i;
            }
        }
    }
    if(x > 1){
        ans = ans / x * (x - 1);
    }
```

b) exgcd

```
int gcd(int a, int b, int& x, int& y)
{
    if(b == 0){
        x = 1;
        y = 0;
        return a;
    }
    else{
        int temp = gcd(b, a % b, x, y);
        int t = y;
        y = x - y * (a / b);
        x = t;
        return temp;
    }
}
```

c) fermat's little

if gcd(a,p)=1， then $a^{(p-1)} \equiv 1 \pmod p$

```
#include <iostream>
#include <cstdio>
#define MAXN 2000001
#define MOD 1000000007
using namespace std;

long long f[MAXN + 5];
long long gcd(long long a, long long b, long long& x, long long& y)
{
    if(b == 0){
        x = 1;
        y = 0;
        return a;
    }
    else{
        long long temp = gcd(b, a % b, x, y);
```

```
        long long t = y;
        y = x - y * (a / b);
        x = t;
        return temp;
    }
}

int main()
{
    f[1] = 1;
    for(long long i = 2; i <= MAXN; i++){
        f[i] = (f[i - 1] * (4 * i - 2)) % MOD;
        gcd(i + 1, MOD, x, y);
        if(x < 0){
            x = MOD - (-x) % MOD;
        }
        x = x % MOD;
        f[i] = (f[i] * x) % MOD;
    }
```

d) Chinese remainder

```
#include <iostream>
#include <cstdio>
#define M 11
using namespace std;

long long a[M], b[M];
long long x, y;
bool flag;
long long gcd(long long  a, long long  b)
{
    if (b == 0){
        return a;
    }
    return gcd(b, a % b);
}

long long exgcd(long long  a, long long  b, long long &
x, long long & y)
{
    if (b == 0){
        x = 1;
        y = 0;
        return a;
    }
    else{
        long long  temp = exgcd(b, a % b, x, y);
        long long  t = y;
        y = x - y * (a / b);
        x = t;
        return temp;
    }
}
```

```cpp
int main()
{
    long long  T, d, n, m, t, A, B;
    scanf("%I64d", &T);
    for (int cas = 1; cas <= T; cas++){
        flag = false;
        scanf("%I64d%I64d", &n, &m);
        for (int i = 1; i <= m; i++){
            scanf("%d", &a[i]);
            //t = (t * a[i]) / gcd(t, a[i]);
        }
        for (int i = 1; i <= m; i++){
            scanf("%I64d", &b[i]);
        }
        A = a[1], B = b[1];
        for (int i = 2; i <= m; i++){
            d = exgcd(A, a[i], x, y);
            if ((b[i] - B) % d != 0){
                flag = true;
                break;
            }
            x = (b[i] - B) / d * x;
            y = a[i] / d;
            x = (x % y + y) % y;
            B = x * A + B;
            A = (A * a[i]) / d;
            B = (B % A + A) % A;
        }
        if (B > n || flag){
            printf("0\n");
        }
        else{
            t = 1 + (n - B) / A;
            if (B == 0){
                --t;
            }
            printf("%I64d\n", t);
        }
    }
}
```

e) fft

```cpp
#include <iostream>
#include <stdio.h>
#include <cmath>
#include <algorithm>
#include <cstring>
#include <vector>
using namespace std;
#define N 50500*2
const double PI = acos(-1.0);
struct Vir
{
```

```
    double re, im;
    Vir(double _re = 0., double _im = 0.) :re(_re), im(_im){}
    Vir operator*(Vir r) { return Vir(re*r.re - im*r.im, re*r.im + im*r.re); }
    Vir operator+(Vir r) { return Vir(re + r.re, im + r.im); }
    Vir operator-(Vir r) { return Vir(re - r.re, im - r.im); }
};
void bit_rev(Vir *a, int loglen, int len)
{
    for (int i = 0; i < len; ++i)
    {
        int t = i, p = 0;
        for (int j = 0; j < loglen; ++j)
        {
            p <<= 1;
            p = p | (t & 1);
            t >>= 1;
        }
        if (p < i)
        {
            Vir temp = a[p];
            a[p] = a[i];
            a[i] = temp;
        }
    }
}
void FFT(Vir *a, int loglen, int len, int on)
{
    bit_rev(a, loglen, len);

    for (int s = 1, m = 2; s <= loglen; ++s, m <<= 1)
    {
        Vir wn = Vir(cos(2 * PI*on / m), sin(2 * PI*on / m));
        for (int i = 0; i < len; i += m)
        {
            Vir w = Vir(1.0, 0);
            for (int j = 0; j < m / 2; ++j)
            {
                Vir u = a[i + j];
                Vir v = w*a[i + j + m / 2];
                a[i + j] = u + v;
                a[i + j + m / 2] = u - v;
                w = w*wn;
            }
        }
    }
    if (on == -1)
    {
```

```
        for (int i = 0; i < len; ++i) a[i].re /= len,
a[i].im /= len;
    }
}
char a[N * 2], b[N * 2];
Vir pa[N * 2], pb[N * 2];
int ans[N * 2];
int main()
{
    while (scanf("%s%s", a, b) != EOF)
    {
        int lena = strlen(a);
        int lenb = strlen(b);
        int n = 1, loglen = 0;
        while (n < lena + lenb) n <<= 1, loglen++;
        for (int i = 0, j = lena - 1; i < n; ++i, --j)
            pa[i] = Vir(j >= 0 ? a[j] - '0' : 0., 0.);
        for (int i = 0, j = lenb - 1; i < n; ++i, --j)
            pb[i] = Vir(j >= 0 ? b[j] - '0' : 0., 0.);
        for (int i = 0; i <= n; ++i) ans[i] = 0;

        FFT(pa, loglen, n, 1);
        FFT(pb, loglen, n, 1);
        for (int i = 0; i < n; ++i)
            pa[i] = pa[i] * pb[i];
        FFT(pa, loglen, n, -1);

        for (int i = 0; i < n; ++i) ans[i] = pa[i].re +
0.5;
        for (int i = 0; i<n; ++i) ans[i + 1] += ans[i] /
10, ans[i] %= 10;

        int pos = lena + lenb - 1;
        for (; pos>0 && ans[pos] <= 0; --pos);
        for (; pos >= 0; --pos) printf("%d", ans[pos]);
        puts("");
    }
    return 0;
}
```

f) else

2. GRATH

a) Prim

```
int  prim() {
    memset(vis,0,sizeof(vis));
    int i;
    int  maxedge=0;
    for (i = 1; i <= n; i++) {
        dis[i]= value[1][i];
    }
    dis[1] = 0;
```

```
    vis[1] = true;
    for (i = 2; i <= n; i++) {
        int temp = inf;
        int mark;
        for (int j = 1; j <= n; j++) {
            if (!vis[j] && dis[j] < temp) {
                temp = dis[j];
                mark = j;
            }
        }
        if(dis[mark]>maxedge)
            maxedge=dis[mark];
        vis[mark]=true;
        for (int j = 1; j <= n; j++) {
            if (!vis[j]&&dis[j]>value[mark][j])
                dis[j] = value[mark][j];
        }
    }
    return maxedge;
}
```

b) Kruskal

c) Dijstra

```
struct Edge
{
    int from, to, dist;
    Edge(int from, int to, int dist):from(from), to(to),
dist(dist){};
};
struct HeapNode
{
    int d, u;
    HeapNode(int d, int u):d(d), u(u){};
    bool operator <(const HeapNode& rhs) const{
        return d > rhs.d;
    }
};
struct Dijstra
{
    int n, m;
    vector<Edge> edges;
    vector<int> G[MAXN];
    bool done[MAXN];
    int d[MAXN];
    int p[MAXN];

    void init(int n){
        this->n = n;
```

```
25          for(int i = 0; i <= n; i++){
26              G[i].clear();
27              road[i].clear();
28          }
29          edges.clear();
30      }
31
32      void AddEdge(int from, int to, int dist){
33          edges.push_back(Edge(from, to, dist));
34          m = edges.size();
35          G[from].push_back(m - 1);
36      }
37
38      void dijstra(int s){
39          priority_queue<HeapNode> Q;
40          for(int i = 0; i <= n; i++){
41              d[i] = INF;
42          }
43          d[s] = 0;
44          memset(done, 0, sizeof(done));
45          Q.push(HeapNode(0, s));
46          while(!Q.empty()){
47              HeapNode x = Q.top();
48              Q.pop();
49              int u = x.u;
50              if(done[u]) continue;
51              done[u]  = true;
52              for(int i = 0; i < G[u].size(); i++){
53                  Edge& e = edges[G[u][i]];
54                  if(d[e.to] > d[u] + e.dist){
55                      d[e.to] = d[u] + e.dist;
56                      p[e.to] = G[u][i];
57                      Q.push(HeapNode(d[e.to], e.to));
58                  }
59              }
60          }
61      }
62  };
```

d) Spfa

```
1  int spfa(int s)
2  {
3      queue <int> q;
4      memset(d, INF, sizeof(d));
5      d[s] = 0;
6      memset(cnt, 0, sizeof(cnt));
7      memset(vis, 0, sizeof(vis));
8      q.push(s);
9      vis[s] = 1;
10     while (!q.empty())
11     {
12         int x;
13         x = q.front();
```

```
        q.pop();
        while (no[x]){
            x = q.front();
            q.pop();
        }
        vis[x] = 0;
        for (int i = 0; i < G[x].size(); i++)
        {
            int y = G[x][i].v;
            if (d[x] + G[x][i].w < d[y])
            {
                d[y] = d[x] + G[x][i].w;
                if (!vis[y])
                {
                    vis[y] = 1;
                        q.push(y);
                }
            }
        }
    }
}
```

e) Two-sat

```
struct TwoSat{
    int n;
    vector<int> G[MAXN*2];
    bool mark[MAXN*2];
    int S[MAXN*2], c;

    bool dfs(int x){
        if(mark[x^1]) return false;
        if(mark[x]) return true;
        mark[x] = true;
        S[c++] = x;
        for(int i = 0; i < G[x].size(); i++){
            if(!dfs(G[x][i])) return false;
        }
        return true;
    }

    void init(int n){
        this->n = n;
        for(int i = 0; i < n * 2; i++){
            G[i].clear();
        }
        memset(mark, 0, sizeof(mark));
    }

    void add_clause(int x, int xval, int y, int yval){
        x = x * 2 + xval;
        y = y * 2 + yval;
        G[x^1].push_back(y);
        G[y^1].push_back(x);
```

```
    }

    bool solve(){
        for(int i = 0; i < n * 2; i += 2){
            if(!mark[i] && !mark[i + 1]){
                c = 0;
                if(!dfs(i)){
                    while(c > 0){
                        mark[S[--c]] = false;
                    }
                    if(!dfs(i + 1)){
                        return false;
                    }
                }
            }
        }
        return true;
    }
};
```

f) MCMF

```
#include <iostream>
#include <string.h>
#include <stdio.h>
#include <algorithm>
#include <queue>
#define V 10100
#define E 1000100
#define inf 99999999
using namespace std;
int vis[V];
int dist[V];
int pre[V];

struct Edge{
    int u,v,c,cost,next;
}edge[E];
int head[V],cnt;

void init(){
    cnt=0;
    memset(head,-1,sizeof(head));
}
void addedge(int u,int v,int c,int cost)
{
    edge[cnt].u=u;edge[cnt].v=v;edge[cnt].cost=cost;
    edge[cnt].c=c;edge[cnt].next=head[u];head[u]=cnt++;

    edge[cnt].u=v;edge[cnt].v=u;edge[cnt].cost=-cost;
    edge[cnt].c=0;edge[cnt].next=head[v];head[v]=cnt++;
}

bool spfa(int begin,int end){
```

```
    int u,v;
    queue<int> q;
    for(int i=0;i<=end+2;i++){
        pre[i]=-1;
        vis[i]=0;
        dist[i]=inf;
    }
    vis[begin]=1;
    dist[begin]=0;
    q.push(begin);
    while(!q.empty()){
        u=q.front();
        q.pop();
        vis[u]=0;
        for(int i=head[u];i!=-1;i=edge[i].next){
            if(edge[i].c>0){
                v=edge[i].v;
                if(dist[v]>dist[u]+edge[i].cost){
                    dist[v]=dist[u]+edge[i].cost;
                    pre[v]=i;
                    if(!vis[v]){
                        vis[v]=true;
                        q.push(v);
                    }
                }
            }
        }
    }
    return dist[end]!=inf;
}

int MCMF(int begin,int end){
    int ans=0,flow;
    int flow_sum=0;
    while(spfa(begin,end)){
        flow=inf;
        for(int i=pre[end];i!=-1;i=pre[edge[i].u])
            if(edge[i].c<flow)
                flow=edge[i].c;
        for(int i=pre[end];i!=-1;i=pre[edge[i].u]){
            edge[i].c-=flow;
            edge[i^1].c+=flow;
        }
        ans+=dist[end];
        flow_sum += flow;
    }
    //cout << flow_sum << endl;
    return ans;
}

int main()
{
```

```
    //freopen("in.txt","r",stdin);
    int n,m,a,b,c;
    while(scanf("%d%d",&n,&m)!=EOF){
        init();
        addedge(0,1,2,0);
        addedge(n,n+1,2,0);
        for(int i=1;i<=m;i++){
            scanf("%d%d%d",&a,&b,&c);
            addedge(a,b,1,c);
            addedge(b,a,1,c);
        }
        printf("%d\n",MCMF(0,n+1));
    }
    return 0;
}
```

g) Max-flow

```
struct Edge{
    int from, to, cap, flow;
    //Edge(int u, int v, int c, int f) :from(u), to(v), cap(c), flow(f){};
};
bool comp(const Edge& a, const Edge& b){
    return (a.from < b.from || (a.from == b.from && a.to < b.to));
}
struct Dinic{
    int n, m, i, s, t;
    Edge e;
    vector<Edge> edges;
    vector<int> G[MAXN];
    int d[MAXN], cur[MAXN];
    bool vis[MAXN];
    void init(int n){
        this->n = n;
        for (i = 0; i <= n; i++){
            G[i].clear();
        }
        edges.clear();
    }
    void AddEdge(int from, int to, int cap){
        edges.push_back(Edge{ from, to, cap, 0 });
        edges.push_back(Edge{ to, from, 0, 0 });
        m = edges.size();
        G[from].push_back(m - 2);
        G[to].push_back(m - 1);
    }
    bool BFS(){
        memset(vis, 0, sizeof(vis));
        queue<int> Q;
        Q.push(s);
        d[s] = 0;
        vis[s] = 1;
```

```
        while (!Q.empty()){
            int x = Q.front();
            Q.pop();
            for (i = 0; i < G[x].size(); i++){
                Edge& e = edges[G[x][i]];
                if (!vis[e.to] && e.cap > e.flow){
                    vis[e.to] = true;
                    d[e.to] = d[x] + 1;
                    Q.push(e.to);
                }
            }
        }
        return vis[t];
    }
    int DFS(int x, int a){
        if (x == t || a == 0) return a;
        int flow = 0, f;
        for (int& i = cur[x]; i < G[x].size(); i++){
            Edge& e = edges[G[x][i]];
            if (d[x] + 1 == d[e.to] && (f = DFS(e.to,
min(a, e.cap - e.flow))) > 0){
                e.flow += f;
                edges[G[x][i] ^ 1].flow -= f;
                flow += f;
                a -= f;
                if (a == 0) break;
            }
        }
        return flow;
    }
    int MaxFlow(int s, int t, int need){
        int flow = 0;
        this->s = s;
        this->t = t;
        while (BFS()){
            memset(cur, 0, sizeof(cur));
            flow += DFS(s, INF);
            if (flow > need) return flow;
        }
        return flow;
    }
    bool checkFull(int s){
        for (int i = 0; i < G[s].size(); i++){
            if (edges[G[s][i]].flow !=
edges[G[s][i]].cap){
                return false;
            }
        }
        return  true;
    }
};
```

h) LCA

```cpp
#include <iostream>
#include <cstdio>
#include <algorithm>
#define LL long long
#define MAXN 10005
#define MAXM 30005
using namespace std;

int euler[MAXM], deep[MAXM], pos[MAXN];
int f[20][MAXN];
vector<int>G[MAXN];
bool vis[MAXN];
int top;
int cnt[MAXN];
void dfs(int t, int x)
{
    if (pos[x] == -1)
        pos[x] = top;
    deep[top] = t;
    euler[top++] = x;

    for (int i = 0; i < G[x].size(); i++)
    {
        dfs(t + 1, G[x][i]);
        deep[top] = t;
        euler[top++] = x;
    }
}

void rmq(int n)
{
    for (int i = 1; i <= n; i++)
        f[0][i] = deep[i];
    for (int j = 1; j <= (int)(log((double)n) / log(2.0)); j++){
        for (int i = 1; i <= n - (1 << j) + 1; i++){
            f[j][i] = min(f[j - 1][i], f[j - 1][i + (1 << (j - 1))]);
        }
    }
}

int get(int x, int y)
{
    if (x > y){
        swap(x, y);
    }
    int k = (int)(log((double)(y - x + 1.0)) / log(2.0));
    int temp = min(f[k][x], f[k][y - (1 << k) + 1]);
    for (int i = x; i <= y; i++)
    if (deep[i] == temp)
```

```
        return euler[i];
}

int main()
{
    int n;
    int a, num, b;
    int root;
    int m, x, y;
    int T;
    scanf("%d", &T);
    for (int cas = 1; cas <= T; cas++){
        scanf("%d", &n);
        top = 1;
        memset(pos, -1, sizeof(pos));
        memset(cnt, 0, sizeof(cnt));
        memset(vis, 0, sizeof(vis));
        for (int i = 1; i <= n; i++)
            G[i].clear();
        for (int i = 1; i < n; i++)
        {
            scanf("%d %d", &x, &y);
            vis[y] = true;
            G[x].push_back(y);
        }

        for (int i = 1; i <= n; i++){
            if (!vis[i]){
                root = i;
                break;
            }
        }
        dfs(0, root);
        rmq(2 * n - 1);
        scanf("%d %d", &x, &y);
        printf("%d\n", get(pos[x], pos[y]));
    }
    return 0;
}
```

i) Hungarian-dfs

```
bool dfs(int u){
    for(int i = 1; i <= n; i++){
        if(a[u][i] && !visit[i]){
            visit[i] = true;
            if(match[i] == -1 || dfs(match[i])){
                match[i] = u;
            }
            return true;
        }r
    }
    return false;
}
```

3. DATA STRUCTURE

a) Kd-tree

```cpp
#include <iostream>
#include <cstdio>
#define LL long long
#define eps 1e-8
#define INF 0x3f3f3f3f
#define MAXN 100005
using namespace std;
int sum[MAXN * 3], add[MAXN * 3];

void pushup(int t){
    sum[t] = sum[t << 1] + sum[t << 1 | 1];
}
void pushdown(int t, int x){
    if (add[t]){
        add[t << 1] += add[t];
        add[t << 1 | 1] += add[t];
        sum[t << 1] += ((x + 1) >> 1)* add[t];
        sum[t << 1 | 1] += (x >> 1) * add[t];
        add[t] = 0;
    }
}
void update(int L, int R, int t, int p, int q, int x){
    if (p <= L && q >= R){
        sum[t] += (R - L + 1) * x;
        add[t] += x;
        return;
    }

    pushdown(t, R - L + 1);
    int mid = (L + R) >> 1;
    if (p <= mid){
        update(L, mid, t << 1, p, q, x);
    }
    if (q > mid){
        update(mid + 1, R, t << 1 | 1, p, q, x);
    }
    pushup(t);
}
int query(int L, int R, int t, int p, int q){
    if (p <= L && q >= R){
        return sum[t];
    }
    pushdown(t, R - L + 1);
    int mid = (L + R) >> 1;
    int res = 0;
    if (p <= mid){
        res += query(L, mid, t << 1, p, q);
    }
    if (q > mid){
        res += query(mid + 1, R, t << 1 | 1, p, q);
```

```
    }
    return  res;
}
int main()
{
    int n;
    while (~scanf("%d", &n) && n){
        memset(sum, 0, sizeof(sum));
        memset(add, 0, sizeof(add));
        int x, y;
        for (int i = 1; i <= n; i++){
            scanf("%d%d", &x, &y);
            update(1, n, 1, x, y, 1);
        }
        for (int i = 1; i < n; i++){
            printf("%d ", query(1, n, 1, i, i));
        }
        printf("%d\n", query(1, n, 1, n, n));
    }
}
```

b) Tree array

```
int lowbit(int x)
{
    return x & (-x);
}
void modify(int x,int add)//一维
{
    while(x<=MAXN)
    {
        a[x]+=add;
        x+=lowbit(x);
    }
}
int get_sum(int x)
{
    int ret=0;
    while(x!=0)
    {
        ret+=a[x];
        x-=lowbit(x);
    }
    return ret;
}
void modify(int x,int y,int data)//二维
{
    for(int i=x;i<MAXN;i+=lowbit(i))
        for(int j=y;j<MAXN;j+=lowbit(j))
            a[i][j]+=data;
}
int get_sum(int x,int y)
{
    int res=0;
```

```
    for(int i=x;i>0;i-=lowbit(i))
        for(int j=y;j>0;j-=lowbit(j))
            res+=a[i][j];
    return res;
}
```

c) Sa array

```
#include<iostream>
#include<stdio.h>
#include<string.h>
using namespace std;
#define min(x,y) x>y? y:x
#define N 200010
int dp[N][33];
int wa[N], wb[N], wsf[N], wv[N], sa[N];
int ra[N], height[N], s[N];
char str[N], str1[N];
int cmp(int *r, int a, int b, int k)
{
    return r[a] == r[b] && r[a + k] == r[b + k];
}
void getsa(int *r, int *sa, int n, int m)
{
    int i, j, p, *x = wa, *y = wb, *t;
    for (i = 0; i < m; i++)  wsf[i] = 0;
    for (i = 0; i < n; i++)  wsf[x[i] = r[i]]++;
    for (i = 1; i < m; i++)  wsf[i] += wsf[i - 1];
    for (i = n - 1; i >= 0; i--)  sa[--wsf[x[i]]] = i;
    p = 1;
    j = 1;
    for (; p < n; j *= 2, m = p)
    {
        for (p = 0, i = n - j; i < n; i++)  y[p++] = i;
        for (i = 0; i < n; i++)  if (sa[i] >= j)  y[p++]
= sa[i] - j;
        for (i = 0; i < n; i++)  wv[i] = x[y[i]];
        for (i = 0; i < m; i++)  wsf[i] = 0;
        for (i = 0; i < n; i++)  wsf[wv[i]]++;
        for (i = 1; i < m; i++)  wsf[i] += wsf[i - 1];
        for (i = n - 1; i >= 0; i--)  sa[--wsf[wv[i]]] =
y[i];
        t = x;
        x = y;
        y = t;
        x[sa[0]] = 0;
        for (p = 1, i = 1; i < n; i++)
            x[sa[i]] = cmp(y, sa[i - 1], sa[i], j) ? p -
1 : p++;
    }
}
void getheight(int *r, int n)
{
    int i, j, k = 0;
```

```
    for (i = 1; i <= n; i++)  ra[sa[i]] = i;
    for (i = 0; i < n; i++)
    {
        if (k)
            k--;
        else
            k = 0;
        j = sa[ra[i] - 1];
        while (r[i + k] == r[j + k])
            k++;
        height[ra[i]] = k;
    }
}
int main()
{
    while (cin >> str)
    {
        cin >> str1;
        int n = 0, len = strlen(str);
        for (int i = 0; i < len; i++)
            s[n++] = str[i] - 'a' + 1;
        s[n++] = 28;
        len = strlen(str1);
        for (int i = 0; i < len; i++)
            s[n++] = str1[i] - 'a' + 1;
        s[n] = 0;
        getsa(s, sa, n + 1, 30);
        getheight(s, n);
        int max = 0, pos = 0;
        len = strlen(str);
        for (int i = 2; i<n; i++)
        if (height[i]>max)
        {
            if (0 <= sa[i - 1] && sa[i - 1] < len&&len <
sa[i])
                max = height[i];
            if (0 <= sa[i] && sa[i] < len&&len < sa[i -
1])
                max = height[i];
        }
        cout << max << endl;
    }
    return 0;
}
```

4. GEOMETRY

```
#define eps 1e-8
int dcmp(double x){
    if (fabs(x) < eps) return 0;
    return x < 0 ? -1 : 1;
}
struct Point{
```

```
    double x, y;
    Point(double p = 0, double q = 0){
        x = p;
        y = q;
    }
};
struct Node{
    int p;
    Point a, b;
    Node(Point a1, Point a2, int t){
        a = a1;
        b = a2;
        p = t;
    }
};

typedef Point Vector;

Vector operator + (Vector A, Vector B){
    return Vector(A.x + B.x, A.y + B.y);
}
Vector operator - (Vector A, Vector B){
    return Vector(A.x - B.x, A.y - B.y);
}
Vector operator * (Vector A, double p){
    return Vector(A.x * p, A.y * p);
}
Vector operator / (Vector A, double p){
    return Vector(A.x / p, A.y / p);
}
bool operator == (Vector A, Vector B){
    return dcmp(A.x - B.x) == 0 && dcmp(A.y - B.y) == 0;
}
bool operator > (Vector A, Vector B){
    return A.x > B.x && A.y > B.y;
}
bool operator <(Vector A, Vector B){
    return A.x < B.x && A.y < B.y;
}
//点积
double  Dot(Vector A, Vector B){
    return A.x * B.x + A.y * B.y;
}
//模
double Length(Vector A){
    return sqrt(Dot(A, A));
}
//夹角
double Angle(Vector A, Vector B){
    return acos(Dot(A, B) / Length(A) / Length(B));
}
//叉积
```

```
double Cross(Vector A, Vector B){
    return A.x * B.y - A.y*B.x;
}
//三角形面积
double Area2(Point A, Point B, Point C){
    return Cross(B - A, C - A);
}
//点在直线上投影
Point GetLineProjection(Point P, Point A, Point B){
    Vector v = B - A;
    return A + v * (Dot(v, P - A) / Dot(v, v));
}
//线段相交(不含端点)
bool SegmentProperIntersection(Point a1, Point a2,
Point b1, Point b2){
    double c1 = Cross(a2 - a1, b1 - a1);
    double c2 = Cross(a2 - a1, b2 - a1);
    double c3 = Cross(b2 - b1, a1 - b1);
    double c4 = Cross(b2 - b1, a2 - b1);
    return dcmp(c1) * dcmp(c2) < 0 && dcmp(c3) *
dcmp(c4) < 0;
}
//点在直线上(不含端点)
bool OnSegment(Point p, Point a1, Point a2){
    return dcmp(Cross(a1 - p, a2 - p)) == 0 &&
dcmp(Dot(a1 - p, a2 - p)) < 0;
}
```

5. ALGORITHM

a) RMQ

```
void rmq(int n)
{
    for (int i = 1; i <= n; i++)
        f[0][i] = deep[i];
    for (int j = 1; j <= (int)(log((double)n) /
log(2.0)); j++){
        for (int i = 1; i <= n - (1 << j) + 1; i++){
            f[j][i] = min(f[j - 1][i], f[j - 1][i + (1
<< (j - 1))]);
        }
    }
}
```

b) Manacher

```
void manacher(){
    int res = 0, id = 0;
    for(int i = 1; i <= m; i++) {
        if(res > i){
            p[i] = min(p[2 * id - i], res - i);
        }
        else{
            p[i] = 1;
        }
        //p[i] = mx > i? min(mp[2*id-i], mx-i): 1;
```

```
        while(s[i + p[i]] == s[i - p[i]]){
            p[i]++;
        }
        //while(s[i+mp[i]] == s[i-mp[i]]) mp[i]++;
        if(i + p[i] > res) {
            res = i + p[i];
            id = i;
        }
    }
}
```

c) Ac automatic

```
#include <iostream>
#include <cstdio>
#include <algorithm>
#include <cstring>
#define LL long long
#define INF 0x3f3f3f3f

#define  MAXM 1000005
using namespace std;

int cnt[200];
char s[MAXM];
char words[160][100];
int n, ans;

struct Node
{
    int count, id;
    struct Node *next[26];
    struct Node *fail;
    void init(){
        int i;
        for (int i = 0; i < 26; i++){
            next[i] = NULL;
        }
        count = -1;
        fail = NULL;
        id = -1;
    }
};
Node *root, *d[MAXM];


void insert(char *s, int id){
    int len, k;
    Node *p = root;
    len = strlen(s);
    for (k = 0; k < len; k++){
        int pos = s[k] - 'a';
        if (p->next[pos] == NULL){
            p->next[pos] = new Node;
```

```
            p->next[pos]->init();
            p = p->next[pos];
        }
        else
            p = p->next[pos];
    }
    p->count = id;
}

void build(Node *root){
    int head, tail, i;
    Node *p, *temp;
    head = 0;
    tail = 0;
    root->fail = NULL;
    d[head] = root;
    while (head <= tail){
        temp = d[head++];
        for (int i = 0; i < 26; i++){
            if (temp->next[i] == NULL) continue;
            if (temp == root){
                temp->next[i]->fail = root;
            }
            else{
                p = temp->fail;
                while (p != NULL){
                    if (p->next[i] != NULL){
                        temp->next[i]->fail = p->next[i];
                        break;
                    }
                    p = p->fail;
                }
                if (p == NULL){
                    temp->next[i]->fail = root;
                }
            }
            d[++tail] = temp->next[i];
        }
    }
}

void query(){
    int len = strlen(s);
    Node *p, *temp;
    p = root;
    for (int i = 0; i < len; i++){
        int pos = s[i] - 'a';
        while (!p->next[pos] && p != root)  p = p->fail;
        p = p->next[pos];
        if (!p) p = root;
```

```
        temp = p;
        while (temp != root){
            if (temp->count >= 0){
                cnt[temp->count]++;
            }
            temp = temp->fail;
        }
    }
}


int main()
{
    while (~scanf("%d", &n)){
        if (n == 0) break;
        memset(cnt, 0, sizeof(cnt));
        root = new Node;
        root->init();
        for (int i = 0; i < n; i++){
            scanf("%s", &words[i]);
            insert(words[i], i);
        }
        build(root);
        scanf("%s", s);
        query();
        ans = -1;
        for (int i = 0; i < n; i++){
            if (cnt[i] >ans){
                ans = cnt[i];
            }
        }
        printf("%d\n", ans);
        for (int i = 0; i < n; i++){
            if (cnt[i] == ans){
                printf("%s\n", words[i]);
            }
        }
    }
}
```

d) Kmp

```
#include <iostream>
#include <cstdio>
#define MAXN 1000005
using namespace std;
int n, last[MAXN], j, m = 0;
char s[MAXN];
int main()
{
    while (~scanf("%d", &n)){
        if (n == 0) break;
        memset(last, 0, sizeof(last));
        scanf("%s", s + 1);
```

```
13
14          int k = 0;
15          last[1] = 0;
16          for (int i = 2; i <= n; i++){
17              while (s[k + 1] != s[i] && k > 0){
18                  k = last[k];
19              }
20              if (s[k + 1] == s[i]){
21                  k++;
22              }
23              last[i] = k;
24          }
25          printf("Test case #%d\n", ++m);
26          for (int i = 2; i <= n; i++){
27              j = i - last[i];
28              if (i % j == 0 && i > j){
29                  printf("%d %d\n", i, i / j);
30              }
31          }
32          printf("\n");
33      }
34  }
```

6. STL