# Southampton team notebook

November 25, 2015

## Contents



# 1 Data Structures

## 1.1 aib

```
int AIB[Nmax], N;
inline int zeros(int x) { return x & (-x); }

inline void Add(int x, int q) {
    for (int i = x; i <= N; i += zeros(i)) AIB[i]+=q;
}
```

```
inline int comp(int x) {
    int ret = 0;
    for (int i = x; i > 0; i -= zeros(i)) ret += AIB[i];
    return ret;
}
```

## 1.2 aint2d

```
int N;
vector<long long> AIB[401010];
vector<long long> v[404040];
inline int zeros(int x) { return x & (-x); }
int val[401010];
inline void Add(int nod, int x, int q){
    for(int i=x ;i < AIB[nod].size(); i+= zeros(i)) AIB[nod][i] += q;
}
long long comp(int nod, int x){
    long long ret = 0;
    if(AIB[nod].size() == 0)
        return 0;
    for(int i=x;i>0;i-=zeros(i)) ret += AIB[nod][i];
    return ret;
}
void init(int nod, int st, int dr){
    if(dr - st == 0){
        v[nod].pb(val[st]);
        AIB[nod].resize(v[nod].size()+1);
        Add(nod,1,1);
    } else {
        int mij = (st+dr) / 2;
        init(nod*2, st, mij);
        init(nod*2+1, mij+1, dr);
        v[nod].resize(v[nod*2].size() + v[nod*2+1].size());
        merge(v[nod*2].begin(), v[nod*2].end(), v[nod*2 +1].begin(),
            v[nod*2+1].end(),v[nod].begin());
        AIB[nod].resize(v[nod].size()+1);
        for(int i=1;i<AIB[nod].size();++i){
            Add(nod, i, 1);
        }
    }
}
int getInd(int nod, int x){
    int ret = -1;
    int st = 0, dr = v[nod].size() -1;
    while(st <= dr ){
        int mij = (st+dr)/2;
        if(v[nod][mij] <= x){
            st = mij + 1;
            ret = mij;
        } else {
            dr = mij - 1;
        }
    }
    return ret+1;
}

int K;
long long retV = 0;
int calc(int nod,int ist, int idr, int st, int dr, int k, int l){ //
    beteen ist,idr and k,l
    if (K == 0 && k > l){
        return 0;
    }
    if(ist <= st && idr >= dr){
        retV += comp(nod,getInd(nod,l)) - comp(nod,getInd(nod,k-1));
    } else {
        int mij = (st+dr)/2;
        if (ist <= mij){
            calc(nod*2,ist,idr,st,mij,k,l);
        }
        if (idr > mij){
            calc(nod*2+1,ist,idr,mij+1,dr,k,l);
        }
    }
}
```

## 1.3 lazyaint

```
#include<stdio.h>
#include<algorithm>
#define Nmax 100100
using namespace std;
long long aint[4*Nmax+100],v[Nmax],maxim,M,x,z,y,indic;
long long up[4*Nmax+100];
long long ind[4*Nmax+100];
long long SUM,SUMI;
```

```
long long N;
long long inf = (long long)101010000*100;

inline void relax(int nod,int st,int dr)
{
    long long mij=(st+dr)/2;
    long long val=up[nod];
    if(st!=dr)
    {
        up[2*nod]+=val;
        up[2*nod+1]+=val;
    }
    if(st==dr)
        ind[nod]=st;
    aint[nod]+=up[nod];
    up[nod]=0;
}
// TOP NODE IS MAX, IND TOP IS INDICE OF TOP. VAL IS INTERVAL TO UPDATE
    += VAL;
void update(int nod,int ist,int idr,int st,int dr,long long val)
{
    if(ist<=st&&idr>=dr)
    {
        aint[nod]+=val;
        if(st!=dr) {
            up[2*nod]+=val;
            up[2*nod+1]+=val;
        }
        else ind[nod]=ist;
    }
    else
    {
        if(aint[nod]>0)
            relax(nod,st,dr);
        long long mij=(st+dr)/2;
        if(ist<=mij) update(2*nod,ist,idr,st,mij,val);
        if(idr>mij) update(2*nod+1,ist,idr,mij+1,dr,val);

        if(up[nod*2])
            relax(nod*2,st,mij);
        if(up[nod*2+1])
            relax(nod*2+1,mij+1,dr);
        if(aint[nod*2]>aint[nod*2+1])
        {
            aint[nod]=aint[2*nod+1];
            ind[nod]=ind[2*nod+1];
        }
        else
        {
            aint[nod]=aint[2*nod];
            ind[nod]=ind[2*nod];
        }
    }
    return;
}
```

## 1.4 paduri

```
int findx(int x) {
    int R = x, y;
    while(tata[R] != R)
        R = tata[R];
    while(tata[x] != x) {
    y = tata[x];
    tata[x] = R;
    x = y;
    }
    return R;
}
void unite(int x, int y) {
    x = findx(x);
    y = findx(y);
    if (x == y) return;
    if(h[x] > h[y]) {
        tata[y] = x;
        h[x] += h[y];
    }
    else {
        tata[x] = y;
        h[y] += h[x];
    }
}
```

## 1.5 rmq

```
int rmq[log][Nmax],x,y,maxi,sh;
```

```
int v[Nmax];
int lg[Nmax],N,M;

void genLog(){
    for(int i=2;i<=N;++i)
        lg[i]=lg[i/2]+1;

    for(int i=1;i<=N;++i)
        rmq[0][i]=v[i];
}

void genRmq() {
    for(int i=1;(1<<i)<=N;++i) {
        for(int j=1;j+(1<<i)-1<=N;++j) {
            maxi=1<<(i-1);
            rmq[i][j]=min(rmq[i-1][j],rmq[i-1][j+maxi]);
        }
    }
}
void query(int x,int y) {
    maxi=lg[y-x+1];
    sh=y-x+1-(1<<maxi);
    return min(rmq[maxi][x], rmq[maxi][x+sh]);
}
```

## 1.6 treap

```
struct T {
    int key, priority,nr;
    T *left, *right;
    T() {}
    T(int key, int priority, T* left, T* right) {
        this->key = key;
        this->priority = priority;
        this->left = left, this->right = right;
        this->nr = 0;
    }
} *R, *nil; // nil indica un nod 'gol'
void init(T* &R) {
    srand(unsigned(time(0)));
    R = nil = new T(0, 0, NULL, NULL);
}
void parc(T* n){
    if(n== nil)
        return;
    parc(n->left);
    parc(n->right);
}
inline void update(T* &n){
    if(n==nil)
        return;
    n->nr = n->left->nr + n->right->nr + 1;

}
int search(T* n, int key) {
    if (n == nil) return 0;
    if (key == n->key) return 1;
    if (key < n->key)
        return search(n->left, key);
    else
        return search(n->right, key);
    update(n->right);update(n->left);update(n);
}
void rotleft(T* &n) {
    T *t = n->left;
    n->left = t->right, t->right = n;
    n = t;
    update(t->right);update(t->left);update(t);
    update(n->right);update(n);
}
void rotright(T* &n) {
    T *t = n->right;
    n->right = t->left, t->left = n;
    n = t;
    update(t->right);
    update(t->left);
    update(t);
    update(n->left);
    update(n);
}
int nth(T* &n,int nr){

    if(n==nil)
    return -1;
    if(nr==0)
        return n->key;
    int leftval = n->left->nr;
    if(nr-leftval == 1)
```

```
        return n->key;
    if(leftval >= nr)
        return nth(n->left,nr);
    return nth(n->right,nr-leftval-1);
}
void balance(T* &n) {
    if (n->left->priority > n->priority)
        rotleft(n);
    else if (n->right->priority > n->priority)
        rotright(n);
    update(n->right);update(n->left);update(n);
}
void insert(T* &n, int key, int priority) {
    if (n == nil) {
        n = new T(key, priority, nil, nil);
        n->nr=1; return;
    }
    (n->nr)++;
    if (key <= n->key)
        insert(n->left, key, priority);
    else if (key > n->key)
        insert(n->right, key, priority);
    balance(n);
}
```

# 2 Graf

## 2.1 2sat

```
int N,s[210000],curr,c[210000],sol[210000];
vector<int> g[210000],gt[210000],v[210000];
bool viz[210000],vz[210000];

void dfs(int x) {
    viz[x] = 1;
    for(auto y: g[x]) if(!viz[y]) dfs(y);
    s[++curr] = x;
}
void dfs2(int x, int comp) {
    viz[x] = 0; c[x] = comp;
    v[comp].push_back(x);
    for(auto y: gt[x]) if(viz[y]) dfs2(y,comp);
}
inline int ng(int x) {
    if(x%2) return x-1; return x+1;
}
bool f(int x, int val) {
    vz[x] = 1;
    for(auto y: v[x]) {
        if(sol[y] && sol[y]!=val) return false;
        sol[y] = val;
    }
    for(auto y: v[x]) {
        y = ng(y);
        if(sol[y] && sol[y]!=3-val) return false;
        if(!sol[y]) return f(c[y],3-val);
    }
    return true;
}
inline bool sat() {
    int comp = 0;
    for(int i=2;i<=2*N+1;++i) if(!viz[i]) dfs(i);
    for(int i=curr;i>=1;--i) if(viz[s[i]]) dfs2(s[i],++comp);
    for(int i=1;i<=comp;++i) if(!vz[i]) if(!f(i,1)) return false;
    return true;
}
//s 0 normal, s 1 negation
inline void add_disj(int x, int sx, int y, int sy) {
    g[2*x+(1-sx)].push_back(2*y+sy);
    g[2*y+(1-sy)].push_back(2*x+sx);
    gt[2*y+sy].push_back(2*x+(1-sx));
    gt[2*x+sx].push_back(2*y+(1-sy));
}
```

## 2.2 bellman

```
#define inf 1000000000
vector<pair<int,int> > m[55000];
int N,cnt[55000],d[55000];
queue<int> q;

void bellman(int r) {
    for(int i=1;i<=N;++i) {
        d[i] = inf;
    }
```

```
    d[r] = 0; q.push(r);
    while(!q.empty()) {
        int x = q.front();
        ++cnt[x];
        if(cnt[x] > N) {
            return; // do something for ciclu negativ
        }
        q.pop();
        for(auto p: m[x]) {
            int y = p.first;
            int c = p.second;
            if(d[y] > d[x] + c) {
                d[y] = d[x] + c;
                q.push(y);
            }
        }
    }
}
```

## 2.3 biconex

```
int N,w[100100],low[100100],depth[100100],comp;
bool viz[100100];
vector<pair<int,int> > m; //edges
vector <vector <int> > c; //result
vector <int> a[100100], com; //lista adiacenta

void dfs(int x, int p, int dep) {
    viz[x] = 1; depth[x]=dep; low[x]=dep;
    for(auto y: a[x]) {
        if(!viz[y]) {
            m.pb(mp(x,y)); dfs(y,x,dep+1);
            low[x] = min(low[x],low[y]);
            if(low[y] >= depth[x]) {
                ++comp; com.clear();
                while(true) {
                    int t = m.back().fs, u = m.back().sc;
                    if(w[t] != comp) {
                        w[t] = comp; com.pb(t);
                    }
                    if(w[u] != comp) {
                        w[u] = comp; com.pb(u);
                    }
                    m.pop_back();
                    if(t==x && u==y) break;
                }
                c.pb(com);
            }
        } else if(y!=p) {
            low[x]=min(low[x],depth[y]);
        }
    }
}
void biconex() {
    for(int i=1;i<=N;++i) {
        if(!viz[i]) {
            dfs(i,0,0);
        }
    }
}
```

## 2.4 cicluEuler

```
void dfs(int x){
    for(auto n : g[x]){
        if(isD[n.sc]==0){
            isD[n.sc]=1; // isD = isDeleted
            dfs(n.fs);
        }
    }
    ret.pb(x);
} // fs = node, sc = edge num
```

## 2.5 cuplaj

```
#include<stdio.h>
#include<algorithm>
#include<vector>
using namespace std;
int N,M,K;
int v[25000],x,p=0;
char car;
vector <int> g[25000];
int l[25000],r[25000],u[25000],was[25000],S;
```

```
int cupj(int q)
{
    if(was[q])
        return 0;
    was[q]=1;
    for(auto x : g[q])
    {
        if(!r[x])
        {

            l[q]=x;
            r[x]=q;
            return 1;
        }
    }
    for(auto x: g[q])
    {
        if(cupj(r[x])) {
            l[q]=x;
            r[x]=q;
            return 1;
        }
    }
    return 0;
}
int L,R,y;

int main()
{
freopen("cuplaj.in","r",stdin);
freopen("cuplaj.out","w",stdout);
    scanf("%d%d%d\n",&L,&R,&M);
    for(int i=1;i<=M;++i)
    {
        scanf("%d%d",&x,&y);
            g[x].push_back(y);
    }

   int ok=1;
   while(ok)
   {
    ok=0;
    for(int i=0;i<=L;++i)
        was[i]=0;
    for(int i=1;i<=L;++i)
        if(!l[i])
        {
            ok|= cupj(i);
        }
   }
   for(int i=1;i<=L;++i)
    if(l[i]>0)
        ++S;
    printf("%d\n",S);
    for(int i=1;i<=M;++i)
        if(l[i]>0)
        {
            printf("%d %d\n",i,l[i]);
        }


}
```

## 2.6 dijkstra

```
#define inf 2000000000
int N, d[50010], viz[50010];
vector<pair<int,int> > m[50010];
priority_queue<pair<int,int> > pq;

void dijkstra(int r) {
    for(int i=1;i<=N;++i) {
        d[i] = inf;
    }
    d[r] = 0;
    pq.push(mp(0,r));
    while(!pq.empty()) {
        int x = pq.top().sc;
        pq.pop(); viz[x] = 1;
        for(auto a: m[x]) {
            int y = a.fs;
            int c = a.sc;
            if(!viz[y]) {
                if(d[y] > c + d[x]) {
                    d[y] = c + d[x];
                    pq.push(mp(-d[y],y));
                }
            }
```

```
        }
    }
}
```

## 2.7 flux

```
#define inf 1000000000
int flux,N,S,D,rez,c[1100][1100],f[1100][1100],p[1100];
int q[1100],first,last;
vector<int> m[1100];
bool viz[1100];

inline bool bfs() {
    first = 0, last = 0;
    for(int i=1;i<=N;++i) {
        viz[i] = 0;
    }
    q[last++] = S; viz[S] = 1;
    while(first < last) {
        int x = q[first++];
        if(x==D) continue;
        for(auto y: m[x]) {
            if(!viz[y] && f[x][y] < c[x][y]) {
                q[last++] = y; viz[y]=1; p[y] = x;
            }
        }
    }
    return viz[D];
}
void update() {
    for(auto y: m[D]) {
        if(f[y][D] < c[y][D] && viz[y]) {
            p[D] = y; flux = inf;
            int curr = D;
            while(curr!=S) {
                if(c[p[curr]][curr] - f[p[curr]][curr] < flux) {
                    flux = c[p[curr]][curr] - f[p[curr]][curr];
                }
                if(!flux) break;
                curr = p[curr];
            }
            curr = D;
            while(curr != S) {
                f[p[curr]][curr] += flux;
                f[curr][p[curr]] -= flux;
                curr = p[curr];
            }
            rez += flux;
        }
    }
}
void flow() {
    while(true) {
        if(!bfs()) break;
        update();
    }
}
```

## 2.8 fluxCostMin

```
#define inf 1000000000
int S,D,flux,N,rez,d[400],c[400][400],v[400][400],f[400][400],p[400];
int q[160000],first,last;
vector<int> m[400];
bool viz[400],inq[400];

inline void add(int x) {
    if(inq[x]) return;
    inq[x] = viz[x] = 1;
    q[last++] = x;
}
inline int pop() {
    int x = q[first++];
    inq[x] = 0;
    return x;
}
inline void reset_stuff() {
    first = last = 0;
    for(int i = 1; i <= N; ++i) {
        viz[i] = inq[i] = 0;
        d[i] = 1000000000;
    }
    d[S] = 0;
}
inline bool bfs() {
    reset_stuff();
```

```
    add(S);
    while(first < last) {
        int x = pop();
        if(x==D) continue;
        for(auto y: m[x]) {
            if(f[x][y] < c[x][y] && d[y] > d[x] + v[x][y]) {
                add(y); p[y] = x;
                d[y] = d[x] + v[x][y];
            }
        }
    }
    return viz[D];
}
void update() {
    flux = inf;
    int curr = D;
    while(curr!=S) {
        if(c[p[curr]][curr] - f[p[curr]][curr] < flux) {
            flux = c[p[curr]][curr] - f[p[curr]][curr];
        }
        if(!flux) break;
        curr = p[curr];
    }
    curr = D;
    while(curr!=S) {
        f[p[curr]][curr] += flux;
        f[curr][p[curr]] -= flux;
        curr = p[curr];
    }
    rez += d[D]*flux;
}
void flow() {
    while(true) {
        if(!bfs()) break;
        update();
    }
}
```

## 2.9 heavyPath

```
int N,M,q,x,y,K;
int
    poz[1000100],v[100100],nr[100100],l[100100],p[100100],cmp[100100],viz[100100];
vector<int> g[100100], c[100100];
//Aint stuff
vector<vector<int> > t;
void update(int comp, int nod, int st, int dr, int c, int d) {
    if(st==dr) { t[comp][nod]=d; return; }
    int mij = (st+dr)/2;
    if(c<=mij) update(comp,2*nod,st,mij,c,d);
    else update(comp,2*nod+1,mij+1,dr,c,d);
    t[comp][nod] = max(t[comp][2*nod],t[comp][2*nod+1]);
}
int getmax(int comp, int nod, int st, int dr, int c, int d) {
    int ret = 0;
    if(c<=st && dr<=d) return t[comp][nod];
    int mij = (st+dr)/2;
    if(d>mij) ret = max(ret,getmax(comp,nod*2+1,1+mij,dr,c,d));
    if(c<=mij) ret = max(ret,getmax(comp,nod*2,st,mij,c,d));
    return ret;
}
inline int query_val(int comp, int st, int dr) {
    return getmax(comp, 1, 1, c[comp].size(), st+1, dr+1);
}
//Available queries
inline void update_val(int comp, int poz, int val) {
    update(comp, 1, 1, c[comp].size(), poz+1, val);
    v[c[comp][poz]] = val;
}
int find_max(int x, int y) {
    if(cmp[x] == cmp[y]) return
        query_val(cmp[x],min(poz[x],poz[y]),max(poz[x],poz[y]));
    int px = p[c[cmp[x]][0]], py = p[c[cmp[y]][0]];
    if(l[px] < l[py]) { swap(x,y); swap(px,py); }
    int M = query_val(cmp[x],0,poz[x]);
    return max(M, find_max(px,y));
}
//Preprocessing
void dfs(int x) {
    viz[x] = 1; nr[x] = 1;
    int ind = -1, nrc = -1;
    for(auto y: g[x]) {
        if(viz[y]) continue;
        l[y] = l[x] + 1; p[y] = x; dfs(y);
        if(nr[y] > nrc) { ind = y; nrc = nr[y]; }
        nr[x] += nr[y];
    }
    if(nrc == -1) {
```

```
        vector<int> C; C.pb(x);
        ++K; c[K] = C; cmp[x] = K;
    } else {
        c[cmp[ind]].pb(x); cmp[x] = cmp[ind];
    }
}
void heavy_path(int r) {
    l[r] = 1; dfs(r);
    for(int i=1;i<=K;++i) {
        reverse(c[i].begin(),c[i].end());
        for(int j=0;j<c[i].size();++j) poz[c[i][j]] = j;
    }
    t.resize(K+10);
    for(int i=1;i<=K;++i) t[i].resize(4*c[i].size()+10,0);
    for(int i=1;i<=N;++i) update_val(cmp[i],poz[i],v[i]);
}
```

## 2.10 hungarian

```
#define INF 100000000

int c[60][60], n, M;
int lx[60], ly[60], solx[60], soly[60];
bool S[60], T[60];
int slack[60], slackx[60], prv[60];

void init_labels() {
    for(int i=0;i<n;++i) {
        for(int j=0;j<n;++j) {
            lx[i] = max(lx[i],c[i][j]);
        }
    }
}
void update_labels() {
    int delta = INF;
    for (int i=0;i<n;++i) {
        if (!T[i]) delta = min(delta, slack[i]);
    }
    for(int i=0;i<n;++i) {
        if(S[i]) lx[i] -= delta;
        if(T[i]) ly[i] += delta;
        else slack[i] -= delta;
    }
}
void add_to_tree(int x, int p) {
    S[x] = 1; prv[x] = p;
    for (int y=0;y<n;++y)
        if (lx[x] + ly[y] - c[x][y] < slack[y]) {
            slack[y] = lx[x] + ly[y] - c[x][y];
            slackx[y] = x;
    }
}
void augment() {
    if (M == n) return;
    int root;
    int q[60], dr = 0, st = 0;
    for(int i=0;i<n;++i) {
        S[i] = 0; T[i] = 0; prv[i] = -1;
    }
    for (int i=0;i<n;++i) {
        if (solx[i] == -1) {
            root = i; q[dr++] = i;
            prv[i] = -2; S[i] = 1; break;
        }
    }
    for (int i=0;i<n;++i) {
        slack[i] = lx[root] + ly[i] - c[root][i];
        slackx[i] = root;
    }
    bool ok = 0;
    int x = -1, y = -1;
    while(1) {
        while(st < dr) {
            x = q[st++];
            for (y = 0; y < n; ++y) {
                if (c[x][y] == lx[x] + ly[y] && !T[y]) {
                    if (soly[y] == -1) {
                        ok = 1; break;
                    }
                    T[y] = 1; q[dr++] = soly[y];
                    add_to_tree(soly[y], x);
                }
            }
            if(ok) break;
        }
        if(ok) break;
        update_labels();
        for (y = 0; y < n; ++y) {
```

```
            if (!T[y] && slack[y] == 0) {
                if (soly[y] == -1) {
                    x = slackx[y];
                    ok = 1;
                    break;
                } else {
                    T[y] = 1;
                    if (!S[soly[y]]) {
                        add_to_tree(soly[y], slackx[y]);
                    }
                }
            }
        }
        if(ok) break;
    }
    if(ok) {
        ++M; int cx = x, cy = y;
        while(cx != -2) {
            int temp = solx[cx];
            soly[cy] = cx; solx[cx] = cy;
            cy = temp; cx = prv[cx];
        }
        augment();
    }
}
int hungarian() { //Returns max cost matching
    int ret = 0;
    M = 0;
    for(int i=0;i<n;++i) {
        solx[i] = -1; soly[i] = -1;
    }
    init_labels();
    augment();
    for (int x = 0; x < n; x++) {
        ret += c[x][solx[x]];
        cout<<x<<" "<<solx[x]<<" "<<c[x][solx[x]]<<"\n";
    }
    return ret;
}
```

## 2.11 lca

```
void dfs(int x, int lev) {
    e[++K] = x;
    L[K] = lev;
    poz[x] = K;
    for(auto y: g[x]) {
        dfs(y,lev+1);
        e[++K] = x;
        L[K] = lev;
    }
}
void preprocess_lca() {
    dfs(1,0);
    for(int i=2;i<=K;++i) {
        Lg[i] = Lg[i/2]+1;
    }
    for(int i=1;i<=K;++i) {
        rmq[0][i]=i;
    }
    for(int i=1;(1<<i) < K; ++i) {
        for(int j=1;j<=K-(1<<i);++j) {
            rmq[i][j] = rmq[i-1][j];
            if(L[rmq[i-1][j + (1<<(i-1))]] < L[rmq[i][j]]) {
                rmq[i][j] = rmq[i-1][j + (1<<(i-1))];
            }
        }
    }
}
int lca(int x, int y) {
    int a = poz[x], b = poz[y];
    if(a>b) {
        swap(a,b);
    }
    int l = Lg[b-a+1];
    int sol = rmq[l][a];
    if(L[sol] > L[rmq[l][b - (1<<l) + 1]]) {
        sol = rmq[l][b - (1<<l) + 1];
    }
    return e[sol];
}
```

## 2.12 lcaLog

```
using namespace std;
int tata[202020][25],N,M, lg[202020], lvl[202020];
```

```
vector<int> g[101010];
void dfs(int nod, int lev){
    lvl[nod] = lev;
    for(auto x: g[nod]){
        dfs(x, lev+1);
    }
}
int lca(int x,int y){
    if(lvl[x] < lvl[y]) swap(x,y);
    int log1=1, log2=1;
    for(;(1<<log1) < lvl[x]; ++log1);
    for(;(1<<log2) < lvl[y]; ++log2);
    for(int k = log1; k >= 0; --k){
        if(lvl[x] - (1 << k) >= lvl[y]){
            x = tata[x][k];
        }
    }
    if (x == y) return x;
    for(int k=log2; k>=0 ;--k) {
        if(tata[x][k] && tata[x][k] != tata[y][k]){
            x = tata[x][k];
            y = tata[y][k];
        }
    }
    return tata[x][0];
}

int main(){
    scanf("%d%d",&N,&M);
    for(int i=2;i<=N;++i){
        scanf("%d",&tata[i][0]);
        g[tata[i][0]].pb(i);
    }
    dfs(1,0);
    for(int k=1; (1<<k) <= N; ++k){
        for(int i=1;i<=N;++i){
            tata[i][k] = tata[tata[i][k-1]][k-1];
        }
    }
    for(int i=1;i<=M;++i){
        int x,y;
        scanf("%d %d",&x,&y);
        printf("%d\n",lca(x,y));
    }
}
```

## 2.13 roy

```
for(int k=1;k<=N;++k)
    for(int i=1;i<=N;++i)
        for(int j=1;j<=N;++j) {
            if(best[i][k]!=0&&best[k][j]!=0&&(best[i][k]+best[k][j]<best[i][j]||be
                best[i][j]=best[i][k]+best[k][j];
        }
```

## 2.14 tareConex

```
include <vector>
#include <stdio.h>
#include <algorithm>
#include <vector>
#include <iostream>

#define pb push_back

using namespace std;

vector<int> g[102020],stack,viz,low,iss;
vector<int> aux;
vector<vector <int> > comp;
int k,index=1;
int N,M,x,y;

void printv(vector<int> v){

    for(auto x : v){
        printf("%d ",x);
    }
    printf("\n");
}

void df(int x){

    viz[x] = index;
    low[x] = index;
    stack[++k] = x;
    iss[x] = 1;
    ++index;
    for(auto n : g[x]){
```

```
        if(viz[n] == 0){
            df(n);
            low[x] = min(low[x],low[n]);
        }
        else{
            if(iss[n]){
                low[x] = min(low[x],low[n]);
            }
        }
    }
    if(low[x] == viz[x]){
        aux.clear();
        do{
            aux.pb(stack[k]);
            iss[stack[k]] = 0;
            --k;
        } while(stack[k+1] != x);

        comp.pb(aux);
    }
}

void init(){
    stack.resize(N+10);
    viz.resize(N+10);
    iss.resize(N+10);
    low.resize(N+10);
}

int main()
{
freopen("ctc.in","r",stdin);
freopen("ctc.out","w",stdout);

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

    init();


    for(int i=1;i<=M;++i){
        scanf("%d%d",&x,&y);
        g[x].pb(y);
    }
    for(int i=1;i<=N;++i){
        if(viz[i]==0){
            df(i);
        }
    }
    printf("%d\n",comp.size());

    for(int i=0;i<comp.size();++i){
        printv(comp[i]);
    }
    return 0;
}
```

# 3 Misc

## 3.1 Brainstoming

```
Brainstoming
- Binary search answer
- Dinamica group & sum to reduce complexity
- Sume/diferente partiale
- Graph traversing (e.g. Euler)
- Pruned backtracking
- Game Theory (win/lose states, min/max)
- Decompose in prime factors
- Sorting (e.g. atan2)
- Split in 2 & bash & combine
- Transform weird math inequalities to geometry (e.g. cross product, area)
- Deque
- Process queries offline (sort, sqrt buckets)
```

## 3.2 Math

```
//start from point [x y 1]
double t[4][4] =
{{1, 0, 0, X},
{0,1,0,Y},
{0,0,1,Z},
{0,0,0,1}};

double sc[4][4] =
{{X,0,0,0},
```

```
{0,Y,0,0},
{0,0,Z,0},
{0,0,0,1}};

double rot[3][3] =
{{cos(a),-sin(a),0},
{sin(a),cos(a)},0},
{0,0,1}};

double rotx[4][4] =
{{1,0,0,0},
{0,cos(a),-sin(a),0},
{0,sin(a),cos(a)},0},
{0,0,0,1}};

double roty[4][4] =
{{cos(a),0,sin(a),0},
{0,1,0,0},
{-sin(a),0,cos(a)},0},
{0,0,0,1}};

double rotz[4][4] =
{{cos(a),-sin(a),0,0},
{sin(a),cos(a),0,0},
{0,0,1,0},
{0,0,0,1}};

polig area: S += x[i]*y[i+1] - x[i+1]*y[i]; S*=0.5;
```

## 3.3 cbin

```
int N, A[N];

int binary_search(int val) {
    int i, step;
    for (step = 1; step < N; step <<= 1);
    for (i = 0; step; step >>= 1)
        if (i + step < N && A[i + step] <= val)
          i += step;
    return i;
}
```

## 3.4 closestpoints

```
long long INF = 4e18;
vector <pair<long long, long long> > v, x, y;
int N;

long long dist(pair <long long, long long> a, pair <long long, long long>
    b) {
    return (a.fs - b.fs) * (a.fs - b.fs) + (a.sc - b.sc) * (a.sc - b.sc);
}
long long solve(int st, int dr) {
    if (st >= dr - 1) return INF;
    if (dr - st == 2) {
        if (y[st] > y[st + 1]) {
            swap(y[st], y[st + 1]);
        }
        if(dist(x[st],x[st+1]) < 0) cout<<x[st].fs<<" "<<x[st].sc<<"
            "<<x[st+1].fs<<" "<<x[st+1].sc<<endl;
        return dist(x[st], x[st + 1]);
    }
    int mij = (st + dr) / 2;
    long long ret = min(solve(st, mij), solve(mij, dr));
    merge(y.begin() + st, y.begin() + mij, y.begin() + mij, y.begin() +
        dr, v.begin());
    copy(v.begin(), v.begin() + (dr - st), y.begin() + st);
    int nr = 0;
    for (int i=st; i<dr; ++i) if (abs(y[i].sc - x[mij].fs) <= ret) {
        v[nr++] = y[i];
    }
    for (int i=0; i<nr-1; ++i) {
        for (int j=i+1; j<nr && j<i+8; ++j) {
            ret = min(ret, dist(v[i], v[j]));
        }
    }
    return ret;
}
double closest_points() {
    sort(x.begin(), x.end());
    y.resize(N), v.resize(N);
    for (int i = 0; i < (int) x.size(); ++ i) {
        y[i] = mp(x[i].sc, x[i].fs);
    }
    return sqrt(solve(0, N));
}
```

### 3.5 euclidExtins

```
int gcd(int a, int b, int &x, int &y) {
    if(b==0) {
        x=1; y=0; return a;
    }
    else {
        int x0, y0, d = gcd(b,a%b,x0,y0);
        x = y0; y = x0 - a/b *y0;
        return d;
    }
}
pair<int,int> euclid(int a, int b, int c) {
    int x, y, sol1, sol2;
    int d = gcd(a,b,x,y);
    if(c%d) {
        return mp(0,0);
    } else {
        sol1 = (c/d)*x; sol2 = -(c/d)*y;
    }
    //only if minimal
    while(sol1 < 0 || sol2 < 0) {
        sol1 += b/d; sol2 += a/d;
    }
    while(sol1 >= b/d || sol2 >= a/d) {
        sol1 -= b/d; sol2 -= a/d;
    }
    return mp(sol1,sol2);
}
int inversmod(int a, int b) {
    int x,y;
    gcd(a,b,x,y);
    if(x<0) {
        int k = (-x-1)/N + 1; x += k*b;
    }
    return x%b;
}
```

### 3.6 fft

```
#define MOD 5767169
vector<int> v1,v2,B;
int k=1,z[2][530000],b[530000];
int powy(int x, int y) {
    if(!y) return 1;
    int z = powy(x,y/2);
    z = (1LL*z*z) % MOD;
    if(y%2) {
        z = (1LL*z*x) % MOD;
    }
    return z;
}
void fft(vector<int> &v, int start, int inc, int rev) {
    if(inc==k) return;
    fft(v,start,inc*2,rev);
    fft(v,start+inc,inc*2,rev);
    int nr = k/inc, Z = 1, zN = z[rev][nr];
    for(int i=0;i<nr/2;++i) {
        int x = (1LL*Z*v[start + (2*i+1)*inc]) % MOD;
        b[start+i*inc] = (v[start + 2*i*inc] + x) % MOD;
        b[start+(i+nr/2)*inc] = (v[start + 2*i*inc] - x + MOD) % MOD;
        Z = (1LL*Z*zN)%MOD;
    }
    for(int i=0;i<nr;++i) {
        v[start+i*inc] = b[start+i*inc];
    }
}
void preprocess_fft(vector<int> &v1, vector<int> &v2) {
    int pw = 0, GEN1 = 177147, GEN2 = 5087924;
    int N = v1.size(), M = v2.size(), deg = M+N;
    while(k<deg) {
        k*=2;
        ++pw;
    }
    for(int i=N;i<k;++i) {
        v1.pb(0);
    }
    for(int i=M;i<k;++i) {
        v2.pb(0);
    }
    pw = 19 - pw;
    for(int i=0;i<pw;++i) {
        GEN1 = (1LL*GEN1*GEN1) % MOD;
        GEN2 = (1LL*GEN2*GEN2) % MOD;
    }
    for(int nr=k;nr>=1;nr/=2) {
        z[0][nr] = GEN1;
```

```
        z[1][nr] = GEN2;
        GEN1 = (1LL*GEN1*GEN1) % MOD;
        GEN2 = (1LL*GEN2*GEN2) % MOD;
    }
}
vector<int> multiply(vector<int> &v1, vector<int> &v2) {
    preprocess_fft(v1, v2);
    vector<int> ret;
    fft(v1,0,1,0);
    fft(v2,0,1,0);
    for(int i=0;i<k;++i) {
        ret.pb((1LL*v1[i]*v2[i])%MOD);
    }
    fft(ret,0,1,1);
    int inv = powy(k,MOD-2);
    for(int i=0;i<k;++i) {
        ret[i] = (1LL*ret[i]*inv) % MOD;
    }
    return ret;
}
```

## 3.7 gaus

```
#define EPS 0.0000001
int i,j,k, N,M;
double A[303][303], X[303];
//N EQUASIONS, M unknowns; A[i][M+1] = result
// A[i][1] * x1 + ... + A[i][M] = A[i][M+1]
int main()
{
    scanf("%d%d",&N,&M);

    for(int i=1;i<=N;++i) {
        for(int j=1;j<=M+1;++j) {
        scanf("%lf",&A[i][j]);
        }
    }
    i=1,j=1;
    while(i<=N && j<=M) {
        for(k=i;k<=N;++k)
            if( A[k][j]<-EPS || A[k][j]>EPS)
                break;
        if(k==N+1) {
            ++j;
            continue;
        }
        if(k!=i) {
            for(int q=1;q<=M+1;++q) {
                double aux = A[i][q];
                A[i][q]= A[k][q];
                A[k][q]= aux;
            }
        }
        for(int q=j+1;q<=M+1;++q) {
            A[i][q]=A[i][q]/A[i][j];
        }
        A[i][j]=1;
        for(int u=i+1;u<=N;++u) {
            for(int q=j+1;q<=M+1;++q) {
                A[u][q]-=A[u][j]*A[i][q];
            }
            A[u][j]=0;
        }
        ++i;++j;
    }
    for(int i=N;i>0;--i)
        for(int j=1;j<=M+1;++j) {
        if(A[i][j]>EPS||A[i][j]<-EPS) {
            if(j==M+1) {
                printf("Imposibil\n");
                return 0;
            }
            X[j]=A[i][M+1];
            for(int k=j+1;k<=M;++k) {
                X[j]-=X[k]*A[i][k];
            }
         break;
        }
    }

    for(int i=1;i<=M;++i) {
        printf("%.8lf ",X[i]);
    }
    printf("\n");
    return 0;
}
```

## 3.8 infasuratoare

```
#define x first
#define y second

using namespace std;

typedef pair<double, double> pct;

int N;

pct v[121010];
pct stiv[121010];

double signc(pct a,pct b,pct c){

    return (b.x - a.x) * (c.y - a.y) - (b.y - a.y) * (c.x - a.x);

}

int cmp(pct a, pct b){

    return signc(v[1],a,b)<0;

}

int k;

int main(){
    scanf("%d",&N);

    for(int i=1;i<=N;++i){
        scanf("%lf%lf",&v[i].x,&v[i].y);
    }
    int pos=1;
    for(int i=2;i<=N;++i){
        if(v[i]<v[pos]){
            pos = i;
        }
    }
    swap(v[pos],v[1]);
    sort(v+2,v+N+1,cmp);

    for(int i=1;i<=N;++i){
        while(k>=2 && signc(stiv[k-1],stiv[k],v[i]) > 0) --k;
        stiv[++k] = v[i];
    }
    printf("%d\n",k);
    for(int i=k;i>=1;--i){
        printf("%lf %lf\n",stiv[i].x,stiv[i].y);
    }

    return 0;
}
```

## 3.9 infasuratoare2

```
struct pt {
    double x, y;
};
bool cmp (pt a, pt b) {
    return a.x < b.x || (a.x == b.x && a.y < b.y);
}
bool cw (pt a, pt b, pt c) {
    return a.x*(b.y-c.y)+b.x*(c.y-a.y)+c.x*(a.y-b.y) < 0;
}
bool ccw (pt a, pt b, pt c) {
    return a.x*(b.y-c.y)+b.x*(c.y-a.y)+c.x*(a.y-b.y) > 0;
}
vector<pt> convex_hull (vector<pt> & a) { //Returns clockwise!
    vector<pt> b;
    if (a.size() == 1) return;
    sort (a.begin(), a.end(), &cmp);
    pt p1 = a[0], p2 = a.back();
    vector<pt> up, down;
    up.push_back (p1);
    down.push_back (p1);
    for (int i=1; i<a.size(); ++i) {
        if (i==a.size()-1 || cw (p1, a[i], p2)) {
            while (up.size()>=2 && !cw (up[up.size()-2], up[up.size()-1],
                a[i])) up.pop_back();
            up.pb(a[i]);
        }
        if(i==a.size()-1 || ccw (p1, a[i], p2)) {
            while (down.size()>=2 && !ccw (down[down.size()-2],
                down[down.size()-1], a[i])) down.pop_back();
            down.pb (a[i]);
```

```
        }
    }
    for (size_t i=0; i<up.size(); ++i) b.push_back (up[i]);
    for (size_t i=down.size()-2; i>0; --i) b.push_back (down[i]);
    return b;
}
```

## 3.10 inmultire

```
inline void mult (int a[DIM][DIM],int b[DIM][DIM],int n,int m,int p)
{
    for (int i=1; i<=n; ++i)
        for (int j=1; j<=p; ++j)
        {
            c[i][j]=0;
            for (int k=1; k<=m; ++k)
                c[i][j]=(c[i][j]+(1LL*a[i][k]*b[k][j])%MOD)%MOD;
        }
} // A[N][M] * B[M][P] = C[N][P]; ANS IN C;
```

## 3.11 misc

```
namespace std{
    template <>
    struct hash<pair<int, int> > {
    public:
           size_t operator()(pair<int, int> x) const throw() {
                size_t h = x.fs + x.sc * 1145;
                return h;
           }
    };
}

ios_base::sync_with_stdio(false);
```

## 3.12 parse

```
#include<stdio.h>
#include<algorithm>
#include<iostream>
using namespace std;
#define BUFFER_SIZE 1234
char buff[BUFFER_SIZE];
int buffIt;

inline int getNumber() {
  int ret = 0;

  while (buff[buffIt] < '0' || buff[buffIt] > '9')
    if (++buffIt == BUFFER_SIZE)
      fread(buff, BUFFER_SIZE, 1, stdin),
      buffIt = 0;

  while (buff[buffIt] >= '0' && buff[buffIt] <= '9') {
    ret = ret * 10 + buff[buffIt] - '0';

    if (++buffIt == BUFFER_SIZE) {
      buffIt = 0;
      fread(buff, BUFFER_SIZE, 1, inputFile);
    }
  }
  return ret;
}


cin >> N >> M;
  for(int i=1;i<=M;++i){
    if(!getline(cin, s))
      break;
    if (s == "") {
      --i;continue; }
    stringstream ss; ss << s;
    int x;
    while(ss >> x) {
      g[i].push_back(x+M+1);
    }
    ++num;
  }
```

## 3.13 read

```
import java.util.StringTokenizer;
```

```
import java.io.BufferedReader;
import java.io.BufferedOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.io.OutputStream;

class Kattio extends PrintWriter {
    public Kattio(InputStream i) {
    super(new BufferedOutputStream(System.out));
    r = new BufferedReader(new InputStreamReader(i));
    }
    public Kattio(InputStream i, OutputStream o) {
    super(new BufferedOutputStream(o));
    r = new BufferedReader(new InputStreamReader(i));
    }

    public boolean hasMoreTokens() {
    return peekToken() != null;
    }

    public int getInt() {
    return Integer.parseInt(nextToken());
    }

    public double getDouble() {
    return Double.parseDouble(nextToken());
    }

    public long getLong() {
    return Long.parseLong(nextToken());
    }

    public String getWord() {
    return nextToken();
    }
    private BufferedReader r;
    private String line;
    private StringTokenizer st;
    private String token;

    private String peekToken() {
    if (token == null)
        try {
        while (st == null || !st.hasMoreTokens()) {
            line = r.readLine();
            if (line == null) return null;
            st = new StringTokenizer(line);
        }
        token = st.nextToken();
        } catch (IOException e) { }
    return token;
    }

    private String nextToken() {
    String ans = peekToken();
    token = null;
    return ans;
    }
}
```

### 3.14 ternary

```
while(true) {
        if (abs(right - left) < e) {
            printf("%.10f", F((left + right) / 2.0 ));
            return 0;
        }
        long double lt = left + (right - left) / 3.0;
        long double rt = right - (right - left) / 3.0;
        if (F(lt) > F(rt)) {
            left = lt;
        } else {
            right = rt;
        }
    }
```

# 4 String

## 4.1 ahoCorasick

```
char s[1000100]; //big string
char t[10100]; //dictionary
int N, r1[10100],r[10100];
```

```
int Q[10100],st,dr;
vector<int> sol,g[1000100];

struct Trie {
    int ind;
    Trie *q[26], *f;
    Trie(int x) {
        ind = x;
        for(int i=0;i<26;++i) q[i] = 0;
    }
};
Trie *T = new Trie(0);
vector<Trie*> v;

void ins(Trie *nod, char *s) {
    if(*s=='\n') {
        sol.pb(nod->ind); return;
    }
    int c = *s - 'a';
    if(nod->q[c]==0) {
        Trie *nt = new Trie(v.size());
        v.pb(nt); nod->q[c] = nt;
    }
    ins(nod->q[c],s+1);
}
void make_fail() {
    for(int i=0;i<26;++i) {
        if(v[0]->q[i] == 0) continue;
        Trie *y = v[0]->q[i];
        Q[dr++] = y->ind;
        y->f = v[0];
        g[0].pb(y->ind);
    }
    while(st < dr) {
        int x = Q[st++];
        for(int i=0;i<26;++i) {
            if(v[x]->q[i] == 0) continue;
            Trie *y = v[x]->q[i];
            Trie *curr = v[x]->f;
            while(curr != 0 && curr->q[i] == 0) curr = curr->f;
            if(curr == 0) y->f = v[0];
            else y->f = curr->q[i];
            g[y->f->ind].pb(y->ind);
            Q[dr++] = y->ind;
        }
    }
}
void dfs(int x) {
    r[x] = r1[x];
    for(auto y: g[x]) {
        dfs(y); r[x] += r[y];
    }
}
void count_occurences() {
    int L = strlen(s);
    Trie *curr = v[0];
    for(int i=0;i<L;++i) {
        int c = s[i]-'a';
        while(curr != 0 && curr->q[c] == 0) {
            curr = curr->f;
        }
        if(curr == 0) curr = v[0];
        else {
            ++r1[curr->q[c]->ind];
            curr = curr->q[c];
        }
    }
    dfs(0);
}
int main() {
    scanf("%s%d",s,&N);
    v.pb(T);
    for(int i=1;i<=N;++i) {
        scanf("%s",t);
        t[strlen(t)] = '\n';
        ins(T,t);
    }
    make_fail();
    count_occurences();
    for(auto x: sol) printf("%d\n",r[x]);
    return 0;
}
```

## 4.2 kmp

```
void make() {
    int k=0;
    nextx[1]=0;
```

```
    for(int i=2;i<=S1;++i) {
        while(k >= 1 && v[k+1] != v[i]) k = nextx[k];
            if(v[k+1] == v[i]) ++k;
            nextx[i] = k;
    }
}

void match() {
    int k=0;
    for(int i=1;i<=S2;++i) {
        while(k >= 1 && v[k+1] != l[i])
            k = nextx[k];
        if(v[k+1] == l[i]) ++k;
        if(k==S1) { // match at i-S1
            k = nextx[k];
        }
    }
}
```

## 4.3 pscpld

```
char s[2010000];
char s1[2010000];
int val[2020201];
int maxind,maxVal,N;
long long S=0;
void make_sir(){
    s1[0]='*';
    for(int i=1;i<=N;++i){
        s1[i*2-1]=s[i]; s1[i*2]='*';
    }
}

int main() {
    scanf("%s",s+1);
    N = strlen(s+1);
    make_sir();
    for(int i=1;i<2*N;++i)
    {
        if(maxVal >= i){
            int loc = maxind - (i-maxind);
            val[i] = min(val[loc],maxVal-i);
        }
        while((i - val[i] >= 0) && (i + val[i] <= 2*N) && (s1[i-val[i]] ==
            s1[i+val[i]])){
            ++val[i];
            if(i + val[i] > maxVal){
                maxVal = i + val[i];
                maxind = i;
            }
        }
    }
    for(int i=1;i<2*N;++i){
        S+=(val[i]+1)/2;
        if(s1[i]=='*') --S;
    }
    printf("%lld",S);
    return 0;
}
```

## 4.4 rabinquery

```
#define Nmax 101010
#define p1 47
#define p2 149
#define MOD1 666013
#define MOD2 991777

int nr1[Nmax],pow1[Nmax],pow2[Nmax],nr2[Nmax],nrfin;
char car[Nmax]; // string we want to hash, STARTING FROM 1

int N,M;

void make()
{
    pow1[0]=1,pow2[0]=1;
    for(int i=1;i<=N;++i)
    {
        pow1[i]=(1LL*pow1[i-1]*p1)%MOD1;
        pow2[i]=(1LL*pow2[i-1]*p2)%MOD2;

        nr1[i]=((1LL*nr1[i-1]*p1)%MOD1 + car[i])%MOD1;
        nr2[i]=((1LL*nr2[i-1]*p2)%MOD2 + car[i])%MOD2;
    }
}
int querry(int x,int y,int x1,int y1,int debug)
```

```
{
    int sol1,sol2,sol12,sol22;

    sol1=1LL*(nr1[y]-(1LL*pow1[y-x+1]*nr1[x-1])%MOD1+MOD1)%MOD1;
    sol2=1LL*(nr1[y1]-(1LL*pow1[y1-x1+1]*nr1[x1-1])%MOD1+MOD1)%MOD1;

    sol12=(nr2[y]-(1LL*pow2[y-x+1]*nr2[x-1])%MOD2+MOD2)%MOD2;
    sol22=(nr2[y1]-(1LL*pow2[y1-x1+1]*nr2[x1-1])%MOD2+MOD2)%MOD2;

    if(sol1==sol2 && sol12 == sol22) return 1;
    return 0;
}
```

## 4.5 suffixAuto

```
unordered_map<char, int> h[200100];
int len[200100],lnk[200100],last,curr,nr;

long long add_char(char c) {
    long long ret = 0;
    curr = ++nr;
    len[curr] = len[last] + 1;
    int p = last;
    while(p !=-1 && !h[p][c]) {
        h[p][c] = curr;
        p = lnk[p];
    }
    if(p==-1) {
        lnk[curr] = 0;
        ret += len[curr];
    } else {
        int q = h[p][c];
        if(len[q] == len[p]+1) {
            lnk[curr] = q;
            ret += (len[curr] - len[q]);
        } else {
            int clone = ++nr;
            len[clone] = len[p] + 1;
            lnk[clone] = lnk[q];
            ret += (len[clone] - len[lnk[q]]);
            h[clone] = h[q];
            while(p!=-1 && h[p][c] == q) {
                h[p][c] = clone;
                p = lnk[p];
            }
            ret -= (len[q] - len[lnk[q]]);
            lnk[q] = clone;
            ret += (len[q] - len[lnk[q]]);
            lnk[curr] = clone;
            ret += (len[curr] - len[clone]);
        }
    }
    last = curr;
    return ret;
}
void suffix_automaton(string &s) {
    last = 0, curr = 0, nr = 0;
    len[0] = 0;
    lnk[0] = -1;
    for(auto c: s) {
        add_char(c); //also counts new suffixes
    }
}
```

## 4.6 suffixarray

```
#include <cstdio>
#include <cstring>
#include <algorithm>
using namespace std;

const int MAXN = 65536;
const int MAXLG = 17;

char A[MAXN];
struct entry {
    int nr[2], p;
} L[MAXN];
int P[MAXLG][MAXN], N, i, stp, cnt;

bool cmp(const entry &a, const entry &b) {
    return a.nr[0] == b.nr[0] ? (a.nr[1] < b.nr[1]) : (a.nr[0] < b.nr[0]);
}

int main() {
    gets(A);
```

```
    for (N = strlen(A), i = 0; i < N; ++i)
        P[0][i] = A[i] - 'a';
    for (stp = 1, cnt = 1; cnt >> 1 < N; ++stp, cnt <<= 1) {
        for (i = 0; i < N; ++i) {
            L[i].nr[0] = P[stp - 1][i];
            L[i].nr[1] = i + cnt < N ? P[stp - 1][i + cnt] : -1;
            L[i].p = i;
        }
        sort(L, L + N, cmp);
        for (i = 0; i < N; ++i)
            P[stp][L[i].p] = i > 0 && L[i].nr[0] == L[i - 1].nr[0] &&
                L[i].nr[1] == L[i - 1].nr[1] ? P[stp][L[i - 1].p] : i;
    }
    return 0;
}
```

## 4.7 trie

```
struct Trie {
    int cnt, nrfii;
    Trie *fiu[26];
    Trie() {
        cnt = nrfii = 0;
        memset(fiu,0,sizeof(fiu));
    }
};
Trie *T = new Trie;

void ins(Trie *nod, char *s) {
    if(*s=='\n') {
        ++nod->cnt;
        return;
    }
    if(nod->fiu[*s-'a']==0) {
        nod->fiu[*s-'a'] = new Trie;
        ++nod->nrfii;
    }
    ins(nod->fiu[*s-'a'],s+1);
}
int del(Trie *nod, char *s) { //only call if exists!!
    if(*s == '\n')
        --nod->cnt;
    else if(del(nod->fiu[*s-'a'],s+1)) {
        nod->fiu[*s-'a'] = 0;
        --nod->nrfii;
    }
    if(nod->cnt == 0 && nod->nrfii == 0 && nod !=T) {
        delete nod;
        return 1;
    }
    return 0;
}
int number(Trie *nod, char *s) {
    if(*s == '\n') {
        return nod->cnt;
    }
    if(nod->fiu[*s-'a']) {
        return number(nod->fiu[*s-'a'],s+1);
    }
    return 0;
}
int longest_prefix(Trie *nod, char *s, int k) {
    if(*s == '\n' || nod->fiu[*s-'a'] == 0) {
        return k;
    }
    return longest_prefix(nod->fiu[*s-'a'],s+1,k+1);
}
void read_lines() {
    char line[32];
    while(!feof(stdin)) {
        fgets(line,32,stdin);
    }
}
```

## 4.8 zalgo

```
// z -> int array, s2 -> char array, N, it's length. At the end have z[i]
int left=0,right=0;
for(int i=1;i<N;++i)
{
    if( i > right){
        left = i;
        right = i;
        while (right < N && s2[right-left] == s2[right]) ++right;
        z[i] = right - left; --right;
    } else {
```

```
        int k = i - left;
        if(z[k] < right-i+1) z[i] = z[k];
        else {
            left=i;
            while (right < N && s2[right - left] == s2[right]) ++right;
            z[i] = right-left; --right;
        }
    }
}
```