Contents

1 Basic

1.1 default code

```
#include<bits/stdc++.h>
#define IO cin.tie(0);cout.tie(0);ios_base::
    sync_with_stdio(false)
#define ll long long
using namespace std;
int main()
{
    IO;
    return 0;
}
```

1.2 vim 指令

```
up - k
down - j
left - h
right - l
復原: u
回復上個動作: ctrl + r
選取整行: 大V
一字一字選取: 小v
區塊選取: ctrl + v
🄵🄵: y
剪下(🄵除): d
貼上: p
移動到文件開頭: gg
移動到文件尾巴: shift + g
游標以下全部選取: vG
游標以下全部🄵除: dG
全選: ggvG
若要整行🄵除, 不必先整行🄵🄵再🄵除, 可以直接dd🄵除整
    行;
要🄵🄵貼上的話, 也不用三個指令, 直接按ppy即可。
```

1.3 .vimrc

```
syn on
se ai nu ru cul mouse=a
se cin et ts=2 sw=2 sts=2
so $VIMRUNTIME/mswin.vim
colo desert
se gfn=Monospace\ 14
```

1.4 check

```
for ((i=0;;i++))
do
    echo "$i"
    python3 gen.py > input
    ./ac < input > ac.out
    ./wa < input > wa.out
    diff ac.out wa.out || break
done
```

1.5 python-related

```
parser:
int(eval(num.replace("/","//")))

from fractions import Fraction
from decimal import Decimal, getcontext
getcontext().prec = 250 # set precision

itwo = Decimal(0.5)
two = Decimal(2)

format(x, '0.10f') # set precision

N = 200
def angle(cosT):
  """given cos(theta) in decimal return theta"""
  for i in range(N):
    cosT = ((cosT + 1) / two) ** itwo
  sinT = (1 - cosT * cosT) ** itwo
  return sinT * (2 ** N)
pi = angle(Decimal(-1))
```

1.6 Binary Search

```
while (l < r)
{
    int mid = l + r >> 1;
    if (q[mid] >= x) r = mid;
    else l = mid + 1;
}
while (l < r)
{
    int mid = l + r + 1 >> 1;
    if (q[mid] <= x) l = mid;
    else r = mid - 1;
}
```

1.7 merge sort

```
void merge_sort(int q[], int l, int r)
{
    if (l >= r) return;

    int mid = l + r >> 1;

    merge_sort(q, l, mid), merge_sort(q, mid + 1, r);

    int k = 0, i = l, j = mid + 1;
    while (i <= mid && j <= r)
        if (q[i] <= q[j]) tmp[k ++ ] = q[i ++ ];
        else tmp[k ++ ] = q[j ++ ];
    while (i <= mid) tmp[k ++ ] = q[i ++ ];
    while (j <= r) tmp[k ++ ] = q[j ++ ];
```

```
    for (i = l, j = 0; i <= r; i ++, j ++ ) q[i] = tmp[
        j];
}
```

1.8 逆序數對

```
//逆序數對
LL merge_sort(int q[], int l, int r)
{
    if (l >= r) return 0;

    int mid = l + r >> 1;

    LL res = merge_sort(q, l, mid) + merge_sort(q, mid
        + 1, r);

    int k = 0, i = l, j = mid + 1;
    while (i <= mid && j <= r)
        if (q[i] <= q[j]) tmp[k ++ ] = q[i ++ ];
        else
        {
            res += mid - i + 1;
            tmp[k ++ ] = q[j ++ ];
        }
    while (i <= mid) tmp[k ++ ] = q[i ++ ];
    while (j <= r) tmp[k ++ ] = q[j ++ ];

    for (i = l, j = 0; i <= r; i ++, j ++ ) q[i] = tmp[
        j];

    return res;
}
```

2 flow

3 Math

3.1 Binary Exponentiation

```
long long binpow(long long a, long long b) {
    long long res = 1;
    while (b > 0) {
        if (b & 1)
            res = res * a;
        a = a * a;
        b >>= 1;
    }
    return res;
}
```

3.2 Euclidean gcd

```
int gcd (int a, int b) {
    while (b) {
        a %= b;
        swap(a, b);
    }
    return a;
}
```

3.3 lcm 最小公倍數

```
int lcm (int a, int b) {
    return a / gcd(a, b) * b;
}
```

3.4 Miller Rabin

```
// n < 4,759,123,141        3 :  2, 7, 61
// n < 1,122,004,669,633    4 :  2, 13, 23, 1662803
// n < 3,474,749,660,383       6 :  pirmes <= 13
// n < 2^64                    7 :
// 2, 325, 9375, 28178, 450775, 9780504, 1795265022
// Make sure testing integer is in range [2, n−2] if
// you want to use magic.
LL magic[]={}
bool witness(LL a,LL n,LL u,int t){
  if(!a) return 0;
  LL x=mypow(a,u,n);
  for(int i=0;i<t;i++) {
    LL nx=mul(x,x,n);
    if(nx==1&&x!=1&&x!=n-1) return 1;
    x=nx;
  }
  return x!=1;
}
bool miller_rabin(LL n) {
  int s=(magic number size)
  // iterate s times of witness on n
  if(n<2) return 0;
  if(!(n&1)) return n == 2;
  ll u=n-1; int t=0;
  // n-1 = u*2^t
  while(!(u&1)) u>>=1, t++;
  while(s--){
    LL a=magic[s]%n;
    if(witness(a,n,u,t)) return 0;
  }
  return 1;
}
```

3.5 ax+by=gcd

```
PII gcd(int a, int b){
  if(b == 0) return {1, 0};
  PII q = gcd(b, a % b);
  return {q.second, q.first - q.second * (a / b)};
}
```

3.6 Discrete sqrt

```
void calcH(LL &t, LL &h, const LL p) {
  LL tmp=p-1; for(t=0;(tmp&1)==0;tmp/=2) t++; h=tmp;
}
// solve equation x^2 mod p = a
bool solve(LL a, LL p, LL &x, LL &y) {
  if(p == 2) { x = y = 1; return true; }
  int p2 = p / 2, tmp = mypow(a, p2, p);
  if (tmp == p - 1) return false;
  if ((p + 1) % 4 == 0) {
    x=mypow(a,(p+1)/4,p); y=p-x; return true;
  } else {
    LL t, h, b, pb; calcH(t, h, p);
    if (t >= 2) {
      do {b = rand() % (p - 2) + 2;
      } while (mypow(b, p / 2, p) != p - 1);
      pb = mypow(b, h, p);
    } int s = mypow(a, h / 2, p);
    for (int step = 2; step <= t; step++) {
      int ss = (((LL)(s * s) % p) * a) % p;
      for(int i=0;i<t-step;i++) ss=mul(ss,ss,p);
      if (ss + 1 == p) s = (s * pb) % p;
      pb = ((LL)pb * pb) % p;
    } x = ((LL)s * a) % p; y = p - x;
  } return true;
}
```

3.7 Roots of Polynomial 找多項式的根

```
const double eps = 1e-12;
const double inf = 1e+12;
double a[ 10 ], x[ 10 ]; // a[0..n](coef) must be
     filled
int n; // degree of polynomial must be filled
int sign( double x ){return (x < -eps)?(-1):(x>eps);}
double f(double a[], int n, double x){
  double tmp=1,sum=0;
  for(int i=0;i<=n;i++)
  { sum=sum+a[i]*tmp; tmp=tmp*x; }
  return sum;
}
double binary(double l,double r,double a[],int n){
  int sl=sign(f(a,n,l)),sr=sign(f(a,n,r));
  if(sl==0) return l; if(sr==0) return r;
  if(sl*sr>0) return inf;
  while(r-l>eps){
    double mid=(l+r)/2;
    int ss=sign(f(a,n,mid));
    if(ss==0) return mid;
    if(ss*sl>0) l=mid; else r=mid;
  }
  return l;
}
void solve(int n,double a[],double x[],int &nx){
```

```
  if(n==1){ x[1]=-a[0]/a[1]; nx=1; return; }
  double da[10], dx[10]; int ndx;
  for(int i=n;i>=1;i--) da[i-1]=a[i]*i;
  solve(n-1,da,dx,ndx);
  nx=0;
  if(ndx==0){
    double tmp=binary(-inf,inf,a,n);
    if (tmp<inf) x[++nx]=tmp;
    return;
  }
  double tmp;
  tmp=binary(-inf,dx[1],a,n);
  if(tmp<inf) x[++nx]=tmp;
  for(int i=1;i<=ndx-1;i++){
    tmp=binary(dx[i],dx[i+1],a,n);
    if(tmp<inf) x[++nx]=tmp;
  }
  tmp=binary(dx[ndx],inf,a,n);
  if(tmp<inf) x[++nx]=tmp;
} // roots are stored in x[1..nx]
```

3.8 Primes

```
/* 12721, 13331, 14341, 75577, 123457, 222557, 556679
* 999983, 1097774749, 1076767633, 100102021, 999997771
* 1001010013, 1000512343, 987654361, 999991231
* 999888733, 98789101, 987777733, 999991921, 1010101333
* 1010102101, 1000000000039, 1000000000000037
* 2305843009213693951, 4611686018427387847
* 9223372036854775783, 18446744073709551557 */
int mu[ N ] , p_tbl[ N ];
vector<int> primes;
void sieve() {
  mu[ 1 ] = p_tbl[ 1 ] = 1;
  for( int i = 2 ; i < N ; i ++ ){
    if( !p_tbl[ i ] ){
      p_tbl[ i ] = i;
      primes.push_back( i );
      mu[ i ] = -1;
    }
    for( int p : primes ){
      int x = i * p;
      if( x >= M ) break;
      p_tbl[ x ] = p;
      mu[ x ] = -mu[ i ];
      if( i % p == 0 ){
        mu[ x ] = 0;
        break;
} } } }
vector<int> factor( int x ){
  vector<int> fac{ 1 };
  while( x > 1 ){
    int fn = SZ(fac), p = p_tbl[ x ], pos = 0;
    while( x % p == 0 ){
      x /= p;
      for( int i = 0 ; i < fn ; i ++ )
        fac.PB( fac[ pos ++ ] * p );
  } }
  return fac;
}
```

3.9 Result

- Lucas' Theorem :
  For $n,m \in \mathbb{Z}^*$ and prime P, $C(m,n)$ mod $P = \Pi(C(m_i,n_i))$ where m_i is the i-th digit of m in base P.
- Stirling approximation :
  $n! \approx \sqrt{2\pi n}(\frac{n}{e})^n e^{\frac{1}{12n}}$
- Stirling Numbers(permutation $|P|=n$ with k cycles):
  $S(n,k) =$ coefficient of x^k in $\Pi_{i=0}^{n-1}(x+i)$
- Stirling Numbers(Partition n elements into k non-empty set):
  $S(n,k) = \frac{1}{k!}\sum\limits_{j=0}^{k}(-1)^{k-j}\binom{k}{j}j^n$
- Pick' s Theorem : $A = i + b/2 - 1$
  其面積 A 和□部格點數目 i、邊上格點數目 b 的關□
- Catalan number : $C_n = \binom{2n}{n}/(n+1)$
  $C_n^{n+m} - C_{n+1}^{n+m} = (m+n)!\frac{n-m+1}{n+1} \quad for \quad n \geq m$
  $C_n = \frac{1}{n+1}\binom{2n}{n} = \frac{(2n)!}{(n+1)!n!}$
  $C_0 = 1 \quad and \quad C_{n+1} = 2(\frac{2n+1}{n+2})C_n$
  $C_0 = 1 \quad and \quad C_{n+1} = \sum_{i=0}^{n} C_i C_{n-i} \quad for \quad n \geq 0$
- Euler Characteristic:
  planar graph: $V - E + F - C = 1$
  convex polyhedron: $V - E + F = 2$
  V, E, F, C: number of vertices, edges, faces(regions), and components
- Kirchhoff's theorem :
  $A_{ii} = deg(i), A_{ij} = (i,j) \in E\ ? -1 : 0$, Deleting any one row, one column, and cal the det(A)
- Polya' theorem (c □方法數, m □總數):
  $(\sum_{i=1}^{m} c^{gcd(i,m)})/m$
- Burnside lemma:
  $|X/G| = \frac{1}{|G|}\sum\limits_{g \in G}|X^g|$
- 錯排公式: (n 個人中, 每個人皆不再原來位置的組合數):
  $dp[0] = 1; dp[1] = 0;$
  $dp[i] = (i-1)*(dp[i-1] + dp[i-2]);$
- Bell 數 (有 n 個人, 把他們拆組的方法總數) :
  $B_0 = 1$
  $B_n = \sum_{k=0}^{n} s(n,k) \quad (second - stirling)$
  $B_{n+1} = \sum_{k=0}^{n}\binom{n}{k}B_k$
- Wilson's theorem :
  $(p-1)! \equiv -1(mod\ \ p)$
- Fermat's little theorem :
  $a^p \equiv a(mod\ \ p)$
- Euler's totient function:
  $A^{B^C} mod\ \ p = pow(A, pow(B, C, p-1)) mod\ \ p$
- 歐拉函數降□公式:
  $A^B \mod C = A^{B \mod \phi(c) + \phi(c)} \mod C$
- 6 的倍數:
  $(a-1)^3 + (a+1)^3 + (-a)^3 + (-a)^3 = 6a$

4 Geometry

5 Graph

5.1 graph

```
稠 密 圖(m>=n^2) 用 鄰 接 矩 陣
稀 疏 圖(m<n^2) 用 鄰 接 表(vt)

最 短 路:
1.單 源 最 短 路
    (1.) 所 有 權 邊 都 是 正 數 -> Dijkstra O(mlogn)
    (2.) 有 負 權 邊 -> Bellman-Ford O(nm)
    (3.) 有 負 權 邊, 但□有 負 環 -> SPFA O(m)~O(nm)
2. 多 源 匯 最 短 路
    (1.) 所 有 權 邊 都 是 正 數 -> Floyd O(n^3)
```

5.2 bfs

```
vector<vector<int>> adj;  // adjacency list
    representation
int n; // number of nodes
int s; // source vertex

queue<int> q;
vector<bool> used(n);
vector<int> d(n), p(n);

q.push(s);
used[s] = true;
p[s] = -1;
while (!q.empty()) {
    int v = q.front();
    q.pop();
    for (int u : adj[v]) {
        if (!used[u]) {
            used[u] = true;
            q.push(u);
            d[u] = d[v] + 1;
            p[u] = v;
        }
    }
}
```

5.3 dfs

```
vector<vector<int>> adj; // graph represented as an
    adjacency list
int n; // number of vertices

vector<bool> visited;

void dfs(int v) {
    visited[v] = true;
    for (int u : adj[v]) {
        if (!visited[u])
            dfs(u);
    }
}
```

5.4 dijkstra(單 源 最 短 路 徑) (堆 優 化 O(mlogn))

```
#define INF 0x3FFFFFFF
typedef pair<int,int>PII;
const int MAXN = 100010;
vector<PII> G[MAXN];
void add_edge(int u,int v,int d){
    G[u].push_back(make_pair(v,d));
}
void init(int n){
    for(int i=0;i<n;i++)
        G[i].clear();
}
int vis[MAXN];
int dis[MAXN];
void dijkstra(int s,int n){
    for(int i=0;i<n;i++)vis[i] = 0;
    for(int i=0;i<n;i++)dis[i] = (i == s ? 0 : INF);
    priority_queue<PII,vector<PII>,greater<PII> >q;
  q.push(make_pair(dis[s],s));
    while(!q.empty()){
        PII p = q.top();
        int x = p.second;
        q.pop();
        if(vis[x])continue;
        vis[x] = 1;
        for(int i=0;i<G[x].size();i++){
            int y = G[x][i].first;
            int d = G[x][i].second;
            if(!vis[y]&&dis[x] + d < dis[y]){
                dis[y] = dis[x] + d;
                q.push(make_pair(dis[y],y));
            }
        }
    }
}
int main()
{
    cin>>n>>m;
    for(ll i=0,a,b,w;i<m;i++)
        cin>>a>>b>>w;add(a,b,w);
    dijkstra(1,n);
    for(ll i=1;i<=n;i++)    cout<<dis[i]<<" ";
    return 0;
}
```

5.5 bellman ford

```
#include<bits/stdc++.h>
using namespace std;
const int N=100010;
int dist[N],backup[N];
int k,n,m;
struct edge{
    int a;int b;int w;
}edge[N];
int bellman_ford()
{
    memset(dist,0x3f,sizeof dist);
    dist[1]=0;
    for(int i=1;i<=k;i++)
    {
        memcpy(backup,dist,sizeof dist);
        for(int j=1;j<=m;j++)
        {
            int a=edge[j].a,b=edge[j].b,w=edge[j].w;
            dist[b]=min(dist[b],backup[a]+w);
        }
    }
    return dist[n];
}
int main()
{
    cin>>n>>m>>k;
    for(int i=1;i<=m;i++)
    {
        int a,b,c;
        cin>>a>>b>>c;
        edge[i].a=a,edge[i].b=b,edge[i].w=c;
    }
    int t=bellman_ford();
    if(t>=0x3f3f3f3f/2)puts("impossible");
    else cout<<t<<endl;
}
```

5.6 SPFA

```
bool spfa(){
    deque<int> dq;
    dis[0]=0;
    dq.push_back(0);
    inq[0]=1;
    while(!dq.empty()){
        int u=dq.front();
        dq.pop_front();
        inq[u]=0;
        for(auto i:edge[u]){
            if(dis[i.first]>i.second+dis[u]){
                dis[i.first]=i.second+dis[u];
                len[i.first]=len[u]+1;
                if(len[i.first]>n)  return 1;
                if(inq[i.first])  continue;
                if(!dq.empty()&&dis[dq.front()]>dis[i.
                    first])
                    dq.push_front(i.first);
                else
                    dq.push_back(i.first);
                inq[i.first]=1;
    } } }
    return 0;
}
```

5.7 floyd

```
#include <iostream>
using namespace std;

const int N = 210, M = 2e+10, INF = 1e9;

int n, m, k, x, y, z;
int d[N][N];

void floyd() {
    for(int k = 1; k <= n; k++)
        for(int i = 1; i <= n; i++)
            for(int j = 1; j <= n; j++)
                d[i][j] = min(d[i][j], d[i][k] + d[k][j
                    ]);
}

int main() {
    cin >> n >> m >> k;
    for(int i = 1; i <= n; i++)
        for(int j = 1; j <= n; j++)
            if(i == j) d[i][j] = 0;
            else d[i][j] = INF;
    while(m--) {
        cin >> x >> y >> z;
        d[x][y] = min(d[x][y], z);
        //注意保存最小的边
    }
    floyd();
    while(k--) {
        cin >> x >> y;
        if(d[x][y] > INF/2) puts("impossible");
        //由于有负权边存在所以约大过INF/2也很合理
        else cout << d[x][y] << endl;
```

```
    }
    return 0;
}
```

5.8 無向圖中字典序最小歐拉路徑

```
//給一個無向圖，點的編號最多□500，邊數最多□1024，首先
    輸入一個m代表邊的數量，然後讓輸出字典序最小的歐拉路
    徑(字典序最小—經過點的編號字典序最小)。題目保證至
    少有一個歐拉路徑。
int n = 500, m,ans[1100], cnt,d[N],g[N][N];
void dfs(int u)
{
    //因□最後的歐拉路徑的序列是ans數組逆序，
    //節點u只有在遍歷完所有邊之後最後才會加到ans數組□
        面，所以逆序過來就是最小的字典序
    for (int i = 1; i <= n; i ++ )
        if (g[u][i])//□邊優化
            g[u][i] --, g[i][u] -- ; dfs(i);
    ans[ ++ cnt] = u;
}
int main()
{
    cin >> m;
    while (m -- )
    {
        int a, b;
        cin >> a >> b;
        g[a][b] ++, g[b][a] ++ ;
        d[a] ++, d[b] ++ ;
    }
    int start = 1;
    while (!d[start]) ++start; // 較小編號作□起點
    //數據保證有解一定存在歐拉□路，那□讓第一條度數□奇
        數的點作□起點
    for (int i = 1; i <= 500; ++i) {
        if (d[i] % 2) { // 奇數點作□起點
            start = i;
            break;
        }
    }
    dfs(start);
    for (int i = cnt; i; i -- ) printf("%d\n", ans[i]);
    return 0;
}
```

5.9 topological sort

```
#include <iostream>
#include <vector>
#include <cstdio>
#include <queue>
#include <cstring>
#include <algorithm>
using namespace std;
int in[10010];
vector<int> v[10010];
int main()
{
    int n,m;
    while(~scanf("%d %d",&n,&m)&&n&&m)
    {
        memset(in,0,sizeof in);//清空入度
        for(int i=0;i<=n;i++) v[i].clear();
        for(int i=0;i<m;i++)
        {
            int x,y;
            cin >>x>>y;//比如x贏了y  或者y是x的兒子 ；
                那□就讓x指向y;
            v[x].push_back(y);
            in[y]++;//y的入度加1
        }
        priority_queue<int,vector<int>,greater<int> >q;
            //優先隊列，設置從小到大排序，小的在隊列下
            □
        for(int i=0;i<n;i++)
        {
            if(in[i]==0)
                q.push(i);//把入度□0的節點壓入隊列
        }
        while(!q.empty())
        {
            int xx=q.top();
            q.pop();
            n--;//每次去掉一個節點
            for(int i=0;i<v[xx].size();i++)
            {
                int yy=v[xx][i];
                in[yy]--;
                if(!in[yy])
                    q.push(yy);//如果去掉上一個節點之後
                        下一個節點的入度變□0，則壓入隊
                        列中
            }
        }
        if(n) cout <<"NO"<<endl;//如果有環的話節點數不
            會□0
        else cout <<"YES"<<endl;
    }
    return 0;
}
```

5.10 Strongly Connected Component

```
struct Scc{
  int n, nScc, vst[MXN], bln[MXN];
  vector<int> E[MXN], rE[MXN], vec;
  void init(int _n){
    n = _n;
    for (int i=0; i<MXN; i++)
      E[i].clear(), rE[i].clear();
  }
  void addEdge(int u, int v){
    E[u].PB(v); rE[v].PB(u);
  }
  void DFS(int u){
    vst[u]=1;
    for (auto v : E[u]) if (!vst[v]) DFS(v);
    vec.PB(u);
  }
  void rDFS(int u){
    vst[u] = 1; bln[u] = nScc;
    for (auto v : rE[u]) if (!vst[v]) rDFS(v);
  }
  void solve(){
    nScc = 0;
    vec.clear();
    FZ(vst);
    for (int i=0; i<n; i++)
      if (!vst[i]) DFS(i);
    reverse(vec.begin(),vec.end());
    FZ(vst);
    for (auto v : vec)
      if (!vst[v]){
        rDFS(v); nScc++;
      }
  }
};
```

5.11 BCC based on vertex

```
struct BccVertex {
  int n,nScc,step,dfn[MXN],low[MXN];
  vector<int> E[MXN],sccv[MXN];
  int top,stk[MXN];
  void init(int _n) {
    n = _n; nScc = step = 0;
    for (int i=0; i<n; i++) E[i].clear();
  }
  void addEdge(int u, int v)
  { E[u].PB(v); E[v].PB(u); }
  void DFS(int u, int f) {
    dfn[u] = low[u] = step++;
    stk[top++] = u;
    for (auto v:E[u]) {
      if (v == f) continue;
      if (dfn[v] == -1) {
        DFS(v,u);
        low[u] = min(low[u], low[v]);
        if (low[v] >= dfn[u]) {
          int z;
          sccv[nScc].clear();
          do {
            z = stk[--top];
```

```
          sccv[nScc].PB(z);
        } while (z != v);
        sccv[nScc++].PB(u);
      }
    }else
      low[u] = min(low[u],dfn[v]);
  } }
  vector<vector<int>> solve() {
    vector<vector<int>> res;
    for (int i=0; i<n; i++)
      dfn[i] = low[i] = -1;
    for (int i=0; i<n; i++)
      if (dfn[i] == -1) {
        top = 0;
        DFS(i,i);
      }
    REP(i,nScc) res.PB(sccv[i]);
    return res;
  }
}graph;
```

5.12 K-th Shortest Path

```
// time: O(|E| \lg |E| + |V| \lg |V| + K)
// memory: O(|E| \lg |E| + |V|)
struct KSP{ // 1-base
  struct nd{
    int u, v; ll d;
    nd(int ui = 0, int vi = 0, ll di = INF)
    { u = ui; v = vi; d = di; }
  };
  struct heap{
    nd* edge; int dep; heap* chd[4];
  };
  static int cmp(heap* a,heap* b)
  { return a->edge->d > b->edge->d; }
  struct node{
    int v; ll d; heap* H; nd* E;
    node(){}
    node(ll _d, int _v, nd* _E)
    { d =_d; v = _v; E = _E; }
    node(heap* _H, ll _d)
    { H = _H; d = _d; }
    friend bool operator<(node a, node b)
    { return a.d > b.d; }
  };
  int n, k, s, t;
  ll dst[ N ];
  nd *nxt[ N ];
  vector<nd*> g[ N ], rg[ N ];
  heap *nullNd, *head[ N ];
  void init( int _n , int _k , int _s , int _t ){
    n = _n; k = _k; s = _s; t = _t;
    for( int i = 1 ; i <= n ; i ++ ){
      g[ i ].clear(); rg[ i ].clear();
      nxt[ i ] = NULL; head[ i ] = NULL;
      dst[ i ] = -1;
  } }
  void addEdge( int ui , int vi , ll di ){
    nd* e = new nd(ui, vi, di);
    g[ ui ].push_back( e );
    rg[ vi ].push_back( e );
  }
  queue<int> dfsQ;
  void dijkstra(){
    while(dfsQ.size()) dfsQ.pop();
    priority_queue<node> Q;
    Q.push(node(0, t, NULL));
    while (!Q.empty()){
      node p = Q.top(); Q.pop();
      if(dst[p.v] != -1) continue;
      dst[ p.v ] = p.d;
      nxt[ p.v ] = p.E;
      dfsQ.push( p.v );
      for(auto e: rg[ p.v ])
        Q.push(node(p.d + e->d, e->u, e));
  } }
  heap* merge(heap* curNd, heap* newNd){
    if(curNd == nullNd) return newNd;
    heap* root = new heap;
    memcpy(root, curNd, sizeof(heap));
    if(newNd->edge->d < curNd->edge->d){
      root->edge = newNd->edge;
      root->chd[2] = newNd->chd[2];
      root->chd[3] = newNd->chd[3];
      newNd->edge = curNd->edge;
      newNd->chd[2] = curNd->chd[2];
      newNd->chd[3] = curNd->chd[3];
    }
    if(root->chd[0]->dep < root->chd[1]->dep)
      root->chd[0] = merge(root->chd[0],newNd);
    else
      root->chd[1] = merge(root->chd[1],newNd);
    root->dep = max(root->chd[0]->dep, root->chd[1]->
        dep) + 1;
    return root;
  }
  vector<heap*> V;
  void build(){
    nullNd = new heap;
    nullNd->dep = 0;
    nullNd->edge = new nd;
    fill(nullNd->chd, nullNd->chd+4, nullNd);
    while(not dfsQ.empty()){
      int u = dfsQ.front(); dfsQ.pop();
      if(!nxt[ u ]) head[ u ] = nullNd;
      else head[ u ] = head[nxt[ u ]->v];
      V.clear();
      for( auto&& e : g[ u ] ){
        int v = e->v;
        if( dst[ v ] == -1 ) continue;
        e->d += dst[ v ] - dst[ u ];
        if( nxt[ u ] != e ){
          heap* p = new heap;
          fill(p->chd, p->chd+4, nullNd);
          p->dep = 1;
          p->edge = e;
          V.push_back(p);
      } }
      if(V.empty()) continue;
      make_heap(V.begin(), V.end(), cmp);
#define L(X) ((X<<1)+1)
#define R(X) ((X<<1)+2)
      for( size_t i = 0 ; i < V.size() ; i ++ ){
        if(L(i) < V.size()) V[i]->chd[2] = V[L(i)];
        else V[i]->chd[2]=nullNd;
        if(R(i) < V.size()) V[i]->chd[3] = V[R(i)];
        else V[i]->chd[3]=nullNd;
      }
      head[u] = merge(head[u], V.front());
  } }
  vector<ll> ans;
  void first_K(){
    ans.clear();
    priority_queue<node> Q;
    if( dst[ s ] == -1 ) return;
    ans.push_back( dst[ s ] );
    if( head[s] != nullNd )
      Q.push(node(head[s], dst[s]+head[s]->edge->d));
    for( int _ = 1 ; _ < k and not Q.empty() ; _ ++ ){
      node p = Q.top(), q; Q.pop();
      ans.push_back( p.d );
      if(head[ p.H->edge->v ] != nullNd){
        q.H = head[ p.H->edge->v ];
        q.d = p.d + q.H->edge->d;
        Q.push(q);
      }
      for( int i = 0 ; i < 4 ; i ++ )
        if( p.H->chd[ i ] != nullNd ){
          q.H = p.H->chd[ i ];
          q.d = p.d - p.H->edge->d + p.H->chd[ i ]->
              edge->d;
          Q.push( q );
  } }   }
  void solve(){ // ans[i] stores the i-th shortest path
    dijkstra();
    build();
    first_K(); // ans.size() might less than k
} }solver;
```

6 String

6.1 PalTree

```cpp
// len[s]是對應的回文長度
// num[s]是有幾個回文後綴
// cnt[s]是這個回文子字串在整個字串中的出現次數
// fail[s]是他長度次長的回文後綴, aba的fail是a
const int MXN = 1000010;
struct PalT{
  int nxt[MXN][26],fail[MXN],len[MXN];
  int tot,lst,n,state[MXN],cnt[MXN],num[MXN];
  int diff[MXN],sfail[MXN],fac[MXN],dp[MXN];
  char s[MXN]={-1};
  int newNode(int l,int f){
    len[tot]=l,fail[tot]=f,cnt[tot]=num[tot]=0;
    memset(nxt[tot],0,sizeof(nxt[tot]));
    diff[tot]=(l>0?l-len[f]:0);
    sfail[tot]=(l>0&&diff[tot]==diff[f]?sfail[f]:f);
    return tot++;
  }
  int getfail(int x){
    while(s[n-len[x]-1]!=s[n]) x=fail[x];
    return x;
  }
  int getmin(int v){
    dp[v]=fac[n-len[sfail[v]]-diff[v]];
    if(diff[v]==diff[fail[v]])
        dp[v]=min(dp[v],dp[fail[v]]);
    return dp[v]+1;
  }
  int push(){
    int c=s[n]-'a',np=getfail(lst);
    if(!(lst=nxt[np][c])){
      lst=newNode(len[np]+2,nxt[getfail(fail[np])][c]);
      nxt[np][c]=lst; num[lst]=num[fail[lst]]+1;
    }
    fac[n]=n;
    for(int v=lst;len[v]>0;v=sfail[v])
        fac[n]=min(fac[n],getmin(v));
    return ++cnt[lst],lst;
  }
  void init(const char *_s){
    tot=lst=n=0;
    newNode(0,1),newNode(-1,1);
    for(;_s[n];) s[n+1]=_s[n],++n,state[n-1]=push();
    for(int i=tot-1;i>1;i--) cnt[fail[i]]+=cnt[i];
  }
}palt;
```

6.2 KMP

```cpp
/* len-failure[k]:
在k結尾的情□下, 這個子字串可以由開頭
長度□(len-failure[k])的部分重□出現來表達

failure[k]□次長相同前綴後綴
如果我們不只想求最多, 而且以0-base做□考量
, 那可能的長度由大到小會是
failuer[k]、failure[failuer[k]-1]
、failure[failure[failuer[k]-1]-1]..
直到有值□0□止 */
int failure[MXN];
void KMP(string& t, string& p)
{
    if (p.size() > t.size()) return;
    for (int i=1, j=failure[0]=-1; i<p.size(); ++i)
    {
        while (j >= 0 && p[j+1] != p[i])
            j = failure[j];
        if (p[j+1] == p[i]) j++;
        failure[i] = j;
    }
    for (int i=0, j=-1; i<t.size(); ++i)
    {
        while (j >= 0 && p[j+1] != t[i])
            j = failure[j];
        if (p[j+1] == t[i]) j++;
        if (j == p.size()-1)
        {
            cout << i - p.size() + 1<<" ";
            j = failure[j];
}   }   }
```

6.3 Z Value

```cpp
int z[MAXN];
void Z_value(const string& s) { //z[i] = lcp(s[1...],s[
    i...])
  int i, j, left, right, len = s.size();
  left=right=0; z[0]=len;
  for(i=1;i<len;i++) {
    j=max(min(z[i-left],right-i),0);
    for(;i+j<len&&s[i+j]==s[j];j++);
    z[i]=j;
    if(i+z[i]>right) {
      right=i+z[i];
      left=i;
}   }   }
```

6.4 Cyclic LCS

```cpp
#define L 0
#define LU 1
#define U 2
const int mov[3][2]={0,-1, -1,-1, -1,0};
int al,bl;
char a[MAXL*2],b[MAXL*2]; // 0-indexed
int dp[MAXL*2][MAXL];
char pred[MAXL*2][MAXL];
inline int lcs_length(int r) {
  int i=r+al,j=bl,l=0;
  while(i>r) {
    char dir=pred[i][j];
    if(dir==LU) l++;
    i+=mov[dir][0];
    j+=mov[dir][1];
  }
  return l;
}
inline void reroot(int r) { // r = new base row
  int i=r,j=1;
  while(j<=bl&&pred[i][j]!=LU) j++;
  if(j>bl) return;
  pred[i][j]=L;
  while(i<2*al&&j<=bl) {
    if(pred[i+1][j]==U) {
      i++;
      pred[i][j]=L;
    } else if(j<bl&&pred[i+1][j+1]==LU) {
      i++;
      j++;
      pred[i][j]=L;
    } else {
      j++;
} } }
int cyclic_lcs() {
  // a, b, al, bl should be properly filled
  // note: a WILL be altered in process
  //         -- concatenated after itself
  char tmp[MAXL];
  if(al>bl) {
    swap(al,bl);
    strcpy(tmp,a);
    strcpy(a,b);
    strcpy(b,tmp);
  }
  strcpy(tmp,a);
  strcat(a,tmp);
  // basic lcs
  for(int i=0;i<=2*al;i++) {
    dp[i][0]=0;
    pred[i][0]=U;
  }
  for(int j=0;j<=bl;j++) {
    dp[0][j]=0;
    pred[0][j]=L;
  }
  for(int i=1;i<=2*al;i++) {
    for(int j=1;j<=bl;j++) {
      if(a[i-1]==b[j-1]) dp[i][j]=dp[i-1][j-1]+1;
      else dp[i][j]=max(dp[i-1][j],dp[i][j-1]);
      if(dp[i][j-1]==dp[i][j]) pred[i][j]=L;
      else if(a[i-1]==b[j-1]) pred[i][j]=LU;
      else pred[i][j]=U;
  } }
  // do cyclic lcs
  int clcs=0;
```

```
  for(int i=0;i<al;i++) {
    clcs=max(clcs,lcs_length(i));
    reroot(i+1);
  }
  // recover a
  a[al]='\0';
  return clcs;
}
```

7 Data Structure

7.1 DSU

```
void init()
{
    for(ll i=1;i<=n;i++) p[i]=i;
}
ll find(ll x)
{
    if(p[x]!=x) p[x]=find(p[x]);
    return p[x];
}
void merge(ll a,ll b)
{
    if(find(a)!=find(b))
    {
        p[find(a)]=find(b);
    }
}
```

7.2 trie

```
//在給定的 N 個整數 A1, A2……AN 中選出兩個進行 xor運
    算，得到的結果最大是多少？
//第一行輸入一個整數 N。
// 第二行輸入 N 個整數 A1～AN。
#include<bits/stdc++.h>
#define IO cin.tie(0);ios_base::sync_with_stdio(false)
#define ll long long
using namespace std;
const ll MAXN = 1e5+10;
ll n,ans,idx;
ll a[MAXN],son[31*MAXN][2];
void insert(int x)
{
    int p=0;
    for(ll i=30;i>=0;i--)
    {
        int u=x>>i&1; // x第i位的二進制數
        if(!son[p][u])  son[p][u]=++idx; // 如果沒路,
            就建新的路
        p=son[p][u]; // p指向idx所指的下標
    }
}
ll query(ll x) //返回第i元素前與二進制a[i]相比最多位數
    不同的數
{
    int p=0;
    int res=0;
    for(ll i=30;i>=0;i--)
    {
        ll u=x>>i&1;
        if(son[p][!u])
        {
            p=son[p][!u];
            res=res*2+!u;
        }
        else
        {
            p=son[p][u];
            res=res*2+u;
        }
    }
    return res;
}
int main()
{
    IO;
    cin>>n;
    for(ll i=0;i<n;i++)
    {
        cin>>a[i];
        insert(a[i]);//建樹
        int t=query(a[i]);//返回第i元素前與二進制a[i]相
            比最多位數不同的數
        ans=max(ans,a[i]^t);
    }
    cout<<ans<<'\n';
    return 0;
}
```

7.3 trie

```
// //给定一个字符串 S，以及一个模式串 P，所有字符串中只
    包含大小写英文字母以及阿拉伯数字。
// 模式串 P 在字符串 S 中多次作为子串出现。
// 求出模式串 P 在字符串 S 中所有出现的位置的起始下标。
// 3
// aba
// 5
// ababa
// ======
// 0 2
#include <iostream>
using namespace std;
const int N = 1e5 + 10, M = 1e6 + 10;
int n, m;
int ne[N];
char s[M], p[N];
void get_next() {
    ne[1] = 0;  //我们的下标从1开始，题目中的下标从0开
        始
    for (int i = 2, j = 0; i <= n; i++) {
        while (j && p[i] != p[j + 1]) {
            j = ne[j];
        }
        if (p[i] == p[j + 1]) {
            j++;
        }
        ne[i] = j;
    }
}
int main() {
    cin >> n >> (p + 1) >> m >> (s + 1);
    //求next数组
    get_next();
    // KMP匹配过程
    for (int i = 1, j = 0; i <= m; i++) {
        while (j && s[i] != p[j + 1]) {
            j = ne[j];
        }
        if (s[i] == p[j + 1]) {
            j++;
        }
        if (j == n) {
            cout << i - n << " ";
            j = ne[j];
        }
    }
    return 0;
}
```

7.4 Segment tree

```
struct seg_tree{
  ll a[MXN],val[MXN*4],tag[MXN*4],NO_TAG=0;
  void push(int i,int l,int r){
    if(tag[i]!=NO_TAG){
      val[i]+=tag[i]; // update by tag
      if(l!=r){
        tag[cl(i)]+=tag[i]; // push
        tag[cr(i)]+=tag[i]; // push
      }
      tag[i]=NO_TAG;
  } }
  void pull(int i,int l,int r){
    int mid=(l+r)>>1;
    push(cl(i),l,mid);push(cr(i),mid+1,r);
    val[i]=max(val[cl(i)],val[cr(i)]); // pull
  }
  void build(int i,int l,int r){
    if(l==r){
      val[i]=a[l]; // set value
      return;
```

```
    }
    int mid=(l+r)>>1;
    build(cl(i),l,mid);build(cr(i),mid+1,r);
    pull(i,l,r);
  }
  void update(int i,int l,int r,int ql,int qr,int v){
    push(i,l,r);
    if(ql<=l&&r<=qr){
      tag[i]+=v; // update tag
      return;
    }
    int mid=(l+r)>>1;
    if(ql<=mid) update(cl(i),l,mid,ql,qr,v);
    if(qr>mid) update(cr(i),mid+1,r,ql,qr,v);
    pull(i,l,r);
  }
  ll query(int i,int l,int r,int ql,int qr){
    push(i,l,r);
    if(ql<=l&&r<=qr)
      return val[i]; // update answer
      ll mid=(l+r)>>1,ret=0;
    if(ql<=mid) ret=max(ret,query(cl(i),l,mid,ql,qr));
    if(qr>mid) ret=max(ret,query(cr(i),mid+1,r,ql,qr));
    return ret;
} }tree;
```

7.5 Treap

```
struct Treap{
  int sz , val , pri , tag;
  Treap *l , *r;
  Treap( int _val ){
    val = _val; sz = 1;
    pri = rand(); l = r = NULL; tag = 0;
  }
};
void push( Treap * a ){
  if( a->tag ){
    Treap *swp = a->l; a->l = a->r; a->r = swp;
    int swp2;
    if( a->l ) a->l->tag ^= 1;
    if( a->r ) a->r->tag ^= 1;
    a->tag = 0;
} }
inline int Size( Treap * a ){ return a ? a->sz : 0; }
void pull( Treap * a ){
  a->sz = Size( a->l ) + Size( a->r ) + 1;
}
Treap* merge( Treap *a , Treap *b ){
  if( !a || !b ) return a ? a : b;
  if( a->pri > b->pri ){
    push( a );
    a->r = merge( a->r , b );
    pull( a );
    return a;
  }else{
    push( b );
    b->l = merge( a , b->l );
    pull( b );
    return b;
} }
void split_kth( Treap *t , int k, Treap*&a, Treap*&b ){
  if( !t ){ a = b = NULL; return; }
  push( t );
  if( Size( t->l ) + 1 <= k ){
    a = t;
    split_kth( t->r , k - Size( t->l ) - 1 , a->r , b )
        ;
    pull( a );
  }else{
    b = t;
    split_kth( t->l , k , a , b->l );
    pull( b );
} }
void split_key(Treap *t, int k, Treap*&a, Treap*&b){
  if(!t){ a = b = NULL; return; }
  push(t);
  if(k<=t->val){
    b = t;
    split_key(t->l,k,a,b->l);
    pull(b);
  }
  else{
    a = t;
    split_key(t->r,k,a->r,b);
    pull(a);
} }
```

8 Others