Contents

1 Basic

1.1 .vimrc

linenumber, relative-linenumber, mouse, cindent, expandtab, shiftwidth, softtabstop, nowrap, ignorecase(when search), noVi compatible, backspace
nornu when enter insert mode

```
se nu rnu mouse=a cin et sw=2 sts=2 nowrap ic nocp bs=2
syn on
au InsertLeave * se rnu
au InsertEnter * se nornu
```

1.2 Default Code

所有模板的 define 都在這

```cpp
#include<bits/stdc++.h>
#define ll long long
#define ld long double
#define INF 0x3f3f3f3f
#define NINF 0xc1c1c1c1
#define LLINF 0x3f3f3f3f3f3f3f3f
#define NLLINF 0xc1c1c1c1c1c1c1c1
#define X first
#define Y second
#define PB emplace_back
#define pll pair<long, long>
using namespace std;
const int MXN = 4e5+5;

void sol(){}
int main(){
  int t=1;
  cin >> t;
  while(t--){
    sol(); } }
```

1.3 Common Sense

陣列過大時本機的指令:
windows: g++ -Wl,-stack,40000000 a.cpp
linux: ulimit -s unlimited
1e7 的 int 陣列 = 4e7 byte = 40 mb
STL 式模板函式名稱定義:
.init(n, ...) ⇒ 初始化並重置全部變數, 0-base
.addEdge(u, v, ...) ⇒ 加入一條邊, 有向圖為 $u \to v$, 無向圖為 $u \leftrightarrow v$
.run() ⇒ 執行並回傳答案
.build() ⇒ 查詢前處理
.query(...) ⇒ 查詢並回傳答案
memset 設-0x3f 的值是 -0x3e3e3e3f / 0xc1c1c1c1

2 flow

2.1 MinCostFlow

2.2 Dinic

求最大流 $O(N^2E)$，求二分最大匹配 $O(E\sqrt{N})$)
dinic.init(n, st, en) ⇒ 0-base
dinic.addEdge(u, v, f) ⇒ $u \to v$, flow f units
dinic.run() ⇒ return max flow from st to en
Dinic 玄學: 若 TLE，可以先加" 正向邊" 且每次都 run()，再全加一次每次都 run()。
範例 code 待補

```cpp
const int MXN = 10005;
struct Dinic{
  struct Edge{ ll v, f, re; };
  int n, s, t, lvl[MXN];
  vector<Edge> e[MXN];
  void init(int _n, int _s, int _t){
    n = _n; s = _s; t = _t;
    for(int i = 0; i < n; ++i) e[i].clear(); }
  void addEdge(int u, int v, ll f = 1){
    e[u].push_back({v, f, e[v].size()});
    e[v].push_back({u, 0, e[u].size() - 1}); }
  bool bfs(){
    memset(lvl, -1, n * 4);
    queue<int> q;
    q.push(s);
    lvl[s] = 0;
    while(!q.empty()){
      int u = q.front(); q.pop();
      for(auto &i : e[u])
        if(i.f > 0 && lvl[i.v] == -1)
          lvl[i.v] = lvl[u] + 1, q.push(i.v); }
    return lvl[t] != -1; }
  ll dfs(int u, ll nf){
    if(u == t) return nf;
    ll res = 0;
    for(auto &i : e[u])
      if(i.f > 0 && lvl[i.v] == lvl[u] + 1){
        int tmp = dfs(i.v, min(nf, i.f));
        res += tmp, nf -= tmp, i.f -= tmp;
        e[i.v][i.re].f += tmp;
        if(nf == 0) return res; }
    if(!res) lvl[u] = -1;
    return res; }
  ll run(ll res){
    while(bfs()) res += dfs(s, LLINF);
    return res; } };
```

2.3 Hungarian

2.4 Kuhn Munkres 最大完美二分匹配

2.5 Directed MST

```cpp
/* Edmond's algoirthm for Directed MST
 * runs in O(VE) */
const int MAXV = 10010;
const int MAXE = 10010;
const int INF  = 2147483647;
struct Edge{
  int u, v, c;
  Edge(int x=0, int y=0, int z=0) : u(x), v(y), c(z){}
};
int V, E, root;
Edge edges[MAXE];
inline int newV(){ return ++ V; }
inline void addEdge(int u, int v, int c)
{ edges[++E] = Edge(u, v, c); }
bool con[MAXV];
int mnInW[MAXV], prv[MAXV], cyc[MAXV], vis[MAXV];
inline int DMST(){
```

```
  fill(con, con+V+1, 0);
  int r1 = 0, r2 = 0;
  while(1){
    fill(mnInW, mnInW+V+1, INF);
    fill(prv, prv+V+1, -1);
    REP(i, 1, E){
      int u=edges[i].u, v=edges[i].v, c=edges[i].c;
      if(u != v && v != root && c < mnInW[v])
        mnInW[v] = c, prv[v] = u;
    }
    fill(vis, vis+V+1, -1);
    fill(cyc, cyc+V+1, -1);
    r1 = 0;
    bool jf = 0;
    REP(i, 1, V){
      if(con[i]) continue ;
      if(prv[i] == -1 && i != root) return -1;
      if(prv[i] > 0) r1 += mnInW[i];
      int s;
      for(s = i; s != -1 && vis[s] == -1; s = prv[s])
        vis[s] = i;
      if(s > 0 && vis[s] == i){
         // get a cycle
        jf = 1; int v = s;
        do{
          cyc[v] = s, con[v] = 1;
          r2 += mnInW[v]; v = prv[v];
        }while(v != s);
        con[s] = 0;
    } }
    if(!jf) break ;
    REP(i, 1, E){
      int &u = edges[i].u;
      int &v = edges[i].v;
      if(cyc[v] > 0) edges[i].c -= mnInW[edges[i].v];
      if(cyc[u] > 0) edges[i].u = cyc[edges[i].u];
      if(cyc[v] > 0) edges[i].v = cyc[edges[i].v];
      if(u == v) edges[i--] = edges[E--];
  } }
  return r1+r2;
}
```

3 Math

3.1 Fast Pow & Inverse & Combination

$fpow(a,b,m) = a^b \pmod m$
$fa[i] = i! \pmod{MOD}$
$fi[i] = i!^{-1} \equiv 1 \pmod{MOD}$
$c(a,b) = \binom{a}{b} \pmod{MOD}$

```
ll fpow(ll a, ll b, ll m){
  ll ret = 1;
  a %= m;
  while(b){
    if(b&1) ret = ret * a % m;
    a = a * a % m;
    b >>= 1; }
  return ret; }

ll fa[MXN], fi[MXN];
void init(){
  fa[0] = 1;
  for(ll i = 1; i < MXN; ++i)
    fa[i] = fa[i - 1] * i % MOD;
  fi[MXN - 1] = fpow(fa[MXN - 1], MOD - 2, MOD);
  for(ll i = MXN - 1; i > 0; --i)
    fi[i - 1] = fi[i] * i % MOD; }

ll c(ll a, ll b){
  return fa[a] * fi[b] % MOD * fi[a - b] % MOD; }
```

3.2 FFT

```
const int MAXN = 262144;
typedef long double ld;
typedef complex<ld> cplx; //real() ,imag()
const ld PI = acosl(-1);
const cplx I(0, 1);
cplx omega[MAXN+1];
void pre_fft(){
  for(int i=0; i<=MAXN; i++)
    omega[i] = exp(i * 2 * PI / MAXN * I);
}
void fft(int n, cplx a[], bool inv=false){
  int basic = MAXN / n;
  int theta = basic;
  for (int m = n; m >= 2; m >>= 1) {
    int mh = m >> 1;
    for (int i = 0; i < mh; i++) {
      cplx w = omega[inv ? MAXN-(i*theta%MAXN)
                         : i*theta%MAXN];
      for (int j = i; j < n; j += m) {
        int k = j + mh;
        cplx x = a[j] - a[k];
        a[j] += a[k];
        a[k] = w * x;
    } }
    theta = (theta * 2) % MAXN;
  }
  int i = 0;
  for (int j = 1; j < n - 1; j++) {
    for (int k = n >> 1; k > (i ^= k); k >>= 1);
    if (j < i) swap(a[i], a[j]);
  }
  if(inv) for (i = 0; i < n; i++) a[i] /= n;
}
cplx arr[MAXN+1];
inline void mul(int _n,ll a[],int _m,ll b[],ll ans[])
{
  int n=1,sum=_n+_m-1;
  while(n<sum)
    n<<=1;
  for(int i=0;i<n;i++)
  {
    double x=(i<_n?a[i]:0),y=(i<_m?b[i]:0);
    arr[i]=complex<double>(x+y,x-y);
  }
  fft(n,arr);
  for(int i=0;i<n;i++)
    arr[i]=arr[i]*arr[i];
  fft(n,arr,true);
  for(int i=0;i<sum;i++){
    ans[i]=(long long int)(arr[i].real()/4 +
              (arr[i].real() > 0 ? 0.5 : -0.5));
  }
}
```

3.3 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;
}
```

3.4 Chinese Remainder

```
LL x[N],m[N];
LL CRT(LL x1, LL m1, LL x2, LL m2) {
  LL g = __gcd(m1, m2);
  if((x2 - x1) % g) return -1;// no sol
  m1 /= g; m2 /= g;
  pair<LL,LL> p = gcd(m1, m2);
  LL lcm = m1 * m2 * g;
  LL res = p.first * (x2 - x1) * m1 + x1;
  return (res % lcm + lcm) % lcm;
}
LL solve(int n){ // n>=2,be careful with no solution
  LL res=CRT(x[0],m[0],x[1],m[1]),p=m[0]/__gcd(m[0],m
      [1])*m[1];
  for(int i=2;i<n;i++){
    res=CRT(res,p,x[i],m[i]);
    p=p/__gcd(p,m[i])*m[i];
  }
  return res;
}
```

3.5 Pollard Rho

```
// does not work when n is prime
LL f(LL x, LL mod){ return add(mul(x,x,mod),1,mod); }
LL pollard_rho(LL n) {
  if(!(n&1)) return 2;
  while(true){
    LL y=2, x=rand()%(n-1)+1, res=1;
    for(int sz=2; res==1; sz*=2) {
      for(int i=0; i<sz && res<=1; i++) {
        x = f(x, n);
        res = __gcd(abs(x-y), n);
      }
      y = x;
    }
    if (res!=0 && res!=n) return res;
} }
```

3.6 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;
}
```

4 Geometry

4.1 definition

```
typedef long double ld;
const ld eps = 1e-8;
int dcmp(ld x) {
  if(abs(x) < eps) return 0;
  else return x < 0 ? -1 : 1;
}
struct Pt {
  ld x, y;
  Pt(ld _x=0, ld _y=0):x(_x), y(_y) {}

  Pt operator+(const Pt &a) const {
    return Pt(x+a.x, y+a.y);
  }
  Pt operator-(const Pt &a) const {
    return Pt(x-a.x, y-a.y);
  }
  Pt operator*(const ld &a) const {
    return Pt(x*a, y*a);
  }
  Pt operator/(const ld &a) const {
    return Pt(x/a, y/a);
  }
  ld operator*(const Pt &a) const {
    return x*a.x + y*a.y;
  }
  ld operator^(const Pt &a) const {
    return x*a.y - y*a.x;
  }
  bool operator<(const Pt &a) const {
    return x < a.x || (x == a.x && y < a.y);
    //return dcmp(x-a.x) < 0 || (dcmp(x-a.x) == 0 &&
        dcmp(y-a.y) < 0);
  }
  bool operator==(const Pt &a) const {
    return dcmp(x-a.x) == 0 && dcmp(y-a.y) == 0;
  }
};
ld norm2(const Pt &a) {
  return a*a;
}
ld norm(const Pt &a) {
  return sqrt(norm2(a));
}
Pt perp(const Pt &a) {
  return Pt(-a.y, a.x);
}
Pt rotate(const Pt &a, ld ang) {
  return Pt(a.x*cos(ang)-a.y*sin(ang), a.x*sin(ang)+a.y
      *cos(ang));
}
struct Line {
  Pt s, e, v; // start, end, end-start
  ld ang;
  Line(Pt _s=Pt(0, 0), Pt _e=Pt(0, 0)):s(_s), e(_e) { v
       = e-s; ang = atan2(v.y, v.x); }

  bool operator<(const Line &L) const {
    return ang < L.ang;
  }
};
struct Circle {
  Pt o; ld r;
  Circle(Pt _o=Pt(0, 0), ld _r=0):o(_o), r(_r) {}
};
```

4.2 Convex Hull

```
double cross(Pt o, Pt a, Pt b){
  return (a-o) ^ (b-o);
}
vector<Pt> convex_hull(vector<Pt> pt){
  sort(pt.begin(),pt.end());
  int top=0;
  vector<Pt> stk(2*pt.size());
  for (int i=0; i<(int)pt.size(); i++){
    while (top >= 2 && cross(stk[top-2],stk[top-1],pt[i
        ]) <= 0)
      top--;
    stk[top++] = pt[i];
  }
  for (int i=pt.size()-2, t=top+1; i>=0; i--){
```

```
    while (top >= t && cross(stk[top-2],stk[top-1],pt[i]) <= 0)
      top--;
    stk[top++] = pt[i];
  }
  stk.resize(top-1);
  return stk;
}
```

4.3 Scan line

```
struct node1 {
  double l, r;
  double sum;
} cl[maxn << 3];
struct node2 {
  double x, y1, y2;
  int flag;
} p[maxn << 3];
bool cmp(node2 a, node2 b) { return a.x < b.x; }
void pushup(int rt) {
  if (lazy[rt] > 0) cl[rt].sum = cl[rt].r - cl[rt].l;
  else cl[rt].sum = cl[rt * 2].sum + cl[rt * 2 + 1].sum;}
void build(int rt, int l, int r) {
  if (r - l > 1) {
    cl[rt].l = s[l];
    cl[rt].r = s[r];
    build(rt * 2, l, (l + r) / 2);
    build(rt * 2 + 1, (l + r) / 2, r);
    pushup(rt);
  } else {
    cl[rt].l = s[l];
    cl[rt].r = s[r];
    cl[rt].sum = 0;
  }
  return;}
void update(int rt, double y1, double y2, int flag) {
  if (cl[rt].l == y1 && cl[rt].r == y2) {
    lazy[rt] += flag;
    pushup(rt);
    return;
  } else {
    if (cl[rt * 2].r > y1) update(rt * 2, y1, min(cl[rt * 2].r, y2), flag);
    if (cl[rt * 2 + 1].l < y2)
      update(rt * 2 + 1, max(cl[rt * 2 + 1].l, y1), y2, flag);
    pushup(rt);
  }}
```

5 Tree

5.1 LCA

求樹上兩點的最低共同祖先
lca.init(n) $\Rightarrow$ 0-base
lca.addEdge(u, v) $\Rightarrow u \leftrightarrow v$
lca.build(root, root) $\Rightarrow O(nlgn)$
lca.qlca(u, v) $\Rightarrow O(lgn)$ u, v 的 LCA
lca.qdis(u, v) $\Rightarrow O(lgn)$ u, v 的距離 (可用倍增法帶權)
lca.anc[u][i] $\Rightarrow$ u 的第 2^i 個祖先

```
const int MXN = 5e5+5;
struct LCA{
  int n, lgn, ti = 0;
  int anc[MXN][24], in[MXN], out[MXN];
  vector<int> g[MXN];
  void init(int _n){
    n = _n, lgn = __lg(n) + 5;
    for(int i = 0; i < n; ++i) g[i].clear(); }
  void addEdge(int u, int v){ g[u].PB(v), g[v].PB(u); }
  void build(int u, int f){
    in[u] = ti++;
    int cur = f;
    for(int i = 0; i < lgn; ++i)
      anc[u][i] = cur, cur = anc[cur][i];
    for(auto i : g[u]) if(i != f) build(i, u);
    out[u] = ti++; }
  bool isanc(int a, int u){
    return in[a] <= in[u] && out[a] >= out[u]; }
  int qlca(int u, int v){
    if(isanc(u, v)) return u;
    if(isanc(v, u)) return v;
    for(int i = lgn-1; i >= 0; --i)
      if(!isanc(anc[u][i], v)) u = anc[u][i];
    return anc[u][0]; }
  int qdis(int u, int v){
    int dis = !isanc(u, v) + !isanc(v, u);
    for(int i = lgn - 1; i >= 0; --i){
      if(!isanc(anc[u][i], v))
        u = anc[u][i], dis += 1<<i;
      if(!isanc(anc[v][i], u))
        v = anc[v][i], dis += 1<<i; }
    return dis; } };
```

6 Graph

6.1 DominatorTree

```
const int MAXN = 100010;
struct DominatorTree{
#define REP(i,s,e) for(int i=(s);i<=(e);i++)
#define REPD(i,s,e) for(int i=(s);i>=(e);i--)
  int n , m , s;
  vector< int > g[ MAXN ] , pred[ MAXN ];
  vector< int > cov[ MAXN ];
  int dfn[ MAXN ] , nfd[ MAXN ] , ts;
  int par[ MAXN ]; //idom[u] s到u的最後一個必經點
  int sdom[ MAXN ] , idom[ MAXN ];
  int mom[ MAXN ] , mn[ MAXN ];
  inline bool cmp( int u , int v )
  { return dfn[ u ] < dfn[ v ]; }
  int eval( int u ){
    if( mom[ u ] == u ) return u;
    int res = eval( mom[ u ] );
    if(cmp( sdom[ mn[ mom[ u ] ] ] , sdom[ mn[ u ] ] ))
      mn[ u ] = mn[ mom[ u ] ];
    return mom[ u ] = res;
  }
  void init( int _n , int _m , int _s ){
    ts = 0; n = _n; m = _m; s = _s;
    REP( i, 1, n ) g[ i ].clear(), pred[ i ].clear();
  }
  void addEdge( int u , int v ){
    g[ u ].push_back( v );
    pred[ v ].push_back( u );
  }
  void dfs( int u ){
    ts++;
    dfn[ u ] = ts;
    nfd[ ts ] = u;
    for( int v : g[ u ] ) if( dfn[ v ] == 0 ){
      par[ v ] = u;
      dfs( v );
    }
  }
  void build(){
    REP( i , 1 , n ){
      dfn[ i ] = nfd[ i ] = 0;
      cov[ i ].clear();
      mom[ i ] = mn[ i ] = sdom[ i ] = i;
    }
    dfs( s );
    REPD( i , n , 2 ){
      int u = nfd[ i ];
      if( u == 0 ) continue ;
      for( int v : pred[ u ] ) if( dfn[ v ] ){
        eval( v );
        if( cmp( sdom[ mn[ v ] ] , sdom[ u ] ) )
          sdom[ u ] = sdom[ mn[ v ] ];
      }
      cov[ sdom[ u ] ].push_back( u );
      mom[ u ] = par[ u ];
      for( int w : cov[ par[ u ] ] ){
        eval( w );
        if( cmp( sdom[ mn[ w ] ] , par[ u ] ) )
          idom[ w ] = mn[ w ];
        else idom[ w ] = par[ u ];
      }
      cov[ par[ u ] ].clear();
    }
    REP( i , 2 , n ){
      int u = nfd[ i ];
      if( u == 0 ) continue ;
      if( idom[ u ] != sdom[ u ] )
```

```
        idom[ u ] = idom[ idom[ u ] ];
    }
  }
} domT;
```

6.2 MaximumClique 最大團

```
#define N 111
struct MaxClique{ // 0-base
  typedef bitset<N> Int;
  Int linkto[N] , v[N];
  int n;
  void init(int _n){
    n = _n;
    for(int i = 0 ; i < n ; i ++){
      linkto[i].reset(); v[i].reset();
    }
  }
  void addEdge(int a , int b)
  { v[a][b] = v[b][a] = 1; }
  int popcount(const Int& val)
  { return val.count(); }
  int lowbit(const Int& val)
  { return val._Find_first(); }
  int ans , stk[N];
  int id[N] , di[N] , deg[N];
  Int cans;
  void maxclique(int elem_num, Int candi){
    if(elem_num > ans){
      ans = elem_num; cans.reset();
      for(int i = 0 ; i < elem_num ; i ++)
        cans[id[stk[i]]] = 1;
    }
    int potential = elem_num + popcount(candi);
    if(potential <= ans) return;
    int pivot = lowbit(candi);
    Int smaller_candi = candi & (~linkto[pivot]);
    while(smaller_candi.count() && potential > ans){
      int next = lowbit(smaller_candi);
      candi[next] = !candi[next];
      smaller_candi[next] = !smaller_candi[next];
      potential --;
      if(next == pivot || (smaller_candi & linkto[next
          ]).count()){
        stk[elem_num] = next;
        maxclique(elem_num + 1, candi & linkto[next]);
      }
    }
  }
  int solve(){
    for(int i = 0 ; i < n ; i ++){
      id[i] = i; deg[i] = v[i].count();
    }
    sort(id , id + n , [&](int id1, int id2){
          return deg[id1] > deg[id2]; });
    for(int i = 0 ; i < n ; i ++) di[id[i]] = i;
    for(int i = 0 ; i < n ; i ++)
      for(int j = 0 ; j < n ; j ++)
        if(v[i][j]) linkto[di[i]][di[j]] = 1;
    Int cand; cand.reset();
    for(int i = 0 ; i < n ; i ++) cand[i] = 1;
    ans = 1;
    cans.reset(); cans[0] = 1;
    maxclique(0, cand);
    return ans;
  }
} solver;
```

6.3 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++;
      }
  }
};
```

7 String

7.1 Rolling Hash

```
struct RollingHash {
  const int p1 = 44129; // 65537, 40961, 90001, 971651
  vector<ll> pre;
  void init(string s) {
    pre.resize(s.size() + 1); pre[0] = 0;
    for (int i = 0; i < (int)s.size(); i++)
      pre[i + 1] = (pre[i] * p1 + s[i]) % MOD;
  }
  ll query(int l, int r) {return (pre[r + 1] - pre[l] *
       fpow(p1, r - l + 1));}
};
```

7.2 KMP

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

failure[k]:
failure[k]為次長相同前綴後綴
如果我們不只想求最多，而且以0-base做為考量
，那可能的長度由大到小會是
failuer[k]、failure[failuer[k]-1]
、failure[failure[failuer[k]-1]-1]..
直到有值為0為止
*/
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];
}   }   }
```

7.3 Aho-Corasick

```
struct ACautomata{
  struct Node{
    int cnt,i;
```

```
    Node *go[26], *fail, *dic;
    Node (){
      cnt = 0; fail = 0; dic=0;
      memset(go,0,sizeof(go));
    }
  }pool[1048576],*root;
  int nMem,n_pattern;
  Node* new_Node(){
    pool[nMem] = Node();
    return &pool[nMem++];
  }
  void init() {nMem=0;root=new_Node();n_pattern=0;}
  void add(const string &str) { insert(root,str,0); }
  void insert(Node *cur, const string &str, int pos){
    for(int i=pos;i<str.size();i++){
      if(!cur->go[str[i]-'a'])
        cur->go[str[i]-'a'] = new_Node();
      cur=cur->go[str[i]-'a'];
    }
    cur->cnt++; cur->i=n_pattern++;
  }
  void make_fail(){
    queue<Node*> que;
    que.push(root);
    while (!que.empty()){
      Node* fr=que.front(); que.pop();
      for (int i=0; i<26; i++){
        if (fr->go[i]){
          Node *ptr = fr->fail;
          while (ptr && !ptr->go[i]) ptr = ptr->fail;
          fr->go[i]->fail=ptr=(ptr?ptr->go[i]:root);
          fr->go[i]->dic=(ptr->cnt?ptr:ptr->dic);
          que.push(fr->go[i]);
  } } } }
  void query(string s){
      Node *cur=root;
      for(int i=0;i<(int)s.size();i++){
          while(cur&&!cur->go[s[i]-'a']) cur=cur->fail;
          cur=(cur?cur->go[s[i]-'a']:root);
          if(cur->i>=0) ans[cur->i]++;
          for(Node *tmp=cur->dic;tmp;tmp=tmp->dic)
              ans[tmp->i]++;
  } }// ans[i] : number of occurrence of pattern i
}AC;
```

7.4 Z Value

```
char s[MAXN];
int len,z[MAXN];
void Z_value() { //z[i] = lcp(s[1...],s[i...])
  int i,j,left,right;
  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;
}   }   }
```

7.5 sa

```
#include <bits/stdc++.h>
using namespace std;
#define N 100010
char T[ N ];
int n , RA[ N ], tempRA[ N ] , SA[ N ], tempSA[ N ] , c[ N ];
void countingSort( int k ){
  int i , sum , maxi = max( 300 , n ) ;
  memset( c , 0 , sizeof c ) ;
  for ( i = 0 ; i < n ; i ++ ) c[ ( i + k < n ) ? RA[i + k] : 0 ] ++ ;
  for ( i = sum = 0 ; i < maxi ; i ++ ) { int t = c[i] ; c[i] = sum ; sum += t ; }
  for ( i = 0 ; i < n ; i ++ )
    tempSA[ c[ ( SA[ i ] + k < n ) ? RA[ SA[ i ] + k ] : 0 ] ++ ] = SA[ i ] ;
  for ( i = 0 ; i < n ; i ++ ) SA[ i ] = tempSA[ i ] ;
}
void constructSA(){
  int r;
  for ( int i = 0 ; i < n ; i ++ ) RA[ i ] = T[ i ] - '.' ;
  for ( int i = 0 ; i < n ; i ++ ) SA[ i ] = i ;
  for ( int k = 1 ; k < n ; k <<= 1 ) {
    countingSort( k ) ; countingSort( 0 ) ;
    tempRA[ SA[ 0 ] ] = r = 0;
    for ( int i = 1 ; i < n ; i ++ )
      tempRA[ SA[ i ] ] = ( RA[ SA[ i ] ] == RA[ SA[ i - 1 ] ] && RA[ SA[ i ] + k ] == RA[ SA[ i - 1 ] + k ] ) ? r : ++ r ;
    for ( int i = 0 ; i < n ; i ++ ) RA[ i ] = tempRA[ i ] ;
  }
}
int main() {
  n = (int)strlen( gets( T ) ) ;
  T[ n ++ ] = '.' ; // important bug fix!
  constructSA() ;
  return 0;
}
```

7.6 ParTree

```
// 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;
```

8 Data Structure

8.1 Treap

```
Treap *th = nullptr
th = merge(th, new Treap(val)) ⇒ 新增元素到 th
th = merge(merge(tl, tm), tr) ⇒ 合併 tl,tm,tr 到 th
split(th, k, tl, tr) ⇒ 分割 th, tl 的元素 ≤ k (失去 BST 性質後不能用)
kth(th, k, tl, tr) ⇒ 分割 th, gsz(tl) ≤ k ( < when gsz(th) < k)
gsz ⇒ get size | gsum ⇒ get sum | th->rev ^= 1 ⇒ 反轉 th
帶懶標版本, 並示範 sum/rev 如何 pull/push
注意 Treap 複雜度好但常數大, 動作能用其他方法就用, 並做 io 等優化
```

```
struct Treap{
```

```
  Treap *l, *r;
  int pri, sz, rev;
  ll val, sum;
  Treap(int _val): l(nullptr), r(nullptr),
    pri(rand()), sz(1), rev(0),
    val(_val), sum(_val){} };

ll gsz(Treap *x){ return x ? x->sz : 0; }
ll gsum(Treap *x){ return x ? x->sum : 0; }

Treap* pull(Treap *x){
  x->sz = gsz(x->l) + gsz(x->r) + 1;
  x->sum = x->val + gsum(x->l) + gsum(x->r);
  return x; }
void push(Treap *x){
  if(x->rev){
    swap(x->l, x->r);
    if(x->l) x->l->rev ^= 1;
    if(x->r) x->r->rev ^= 1;
    x->rev = 0; } }

Treap* merge(Treap* a, Treap* b){
  if(!a || !b) return a ? a : b;
  push(a), push(b);
  if(a->pri > b->pri){
    a->r = merge(a->r, b);
    return pull(a); }
  else{
    b->l = merge(a, b->l);
    return pull(b); } }

void split(Treap *x, int k, Treap *&a, Treap *&b){
  if(!x) a = b = nullptr;
  else{
    push(x);
    if(x->val <= k) a = x, split(x->r, k, a->r, b);
    else            b = x, split(x->l, k, a, b->l);
    pull(x); } }

void kth(Treap *x, int k, Treap *&a, Treap *&b){
  if(!x) a = b = nullptr;
  else{
    push(x);
    if(gsz(x->l) < k)
         a = x, kth(x->r, k - gsz(x->l) - 1, a->r, b);
    else b = x, kth(x->l, k, a, b->l);
    pull(x); } }
```

9 Others

Prime	Root	Prime	Root
7681	17	167772161	3
12289	11	104857601	3
40961	3	985661441	3
65537	3	998244353	3
786433	10	1107296257	10
5767169	3	2013265921	31
7340033	3	2810183681	11
23068673	3	2885681153	3
469762049	3	605028353	3