

Contents

1	Basic
1.1	.vimrc
1.2	Default Code
1.3	Common Sense
2	flow
2.1	MinCostFlow
2.2	Dinic
2.3	Hungarian
2.4	Kuhn Munkres 最大完美二分匹配
2.5	Directed MST
3	Math
3.1	Fast Pow & Inverse & Combination
3.2	FFT
3.3	Miller Rabin
3.4	Chinese Remainder
3.5	Pollard Rho
3.6	Primes
4	Geometry
4.1	definition
4.2	Convex Hull
4.3	Scan line
5	Tree
5.1	LCA
6	Graph
6.1	DominatorTree
6.2	MaximumClique 最大團
6.3	Strongly Connected Component
7	String
7.1	KMP
7.2	Aho-Corasic
7.3	Z Value
7.4	sa
7.5	ParTree
8	Data Structure
8.1	Treap
9	Others

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

```

```

1 se nu rnu mouse=a cin et sw=2 sts=2 nowrap ic nosp bs=2
2 syn on
3 au InsertLeave * se rnu
4 au InsertEnter * se nornu

```

1.2 Default Code

所有模板的 define 都在這

```

1 #include<bits/stdc++.h>
2 #define ll long long
3 #define ld long double
4 #define INF 0x3f3f3f3f
5 #define NINF 0xc1c1c1c1
6 #define LLINF 0x3f3f3f3f3f3f3f3f
7 #define NLLINF 0xc1c1c1c1c1c1c1c1
8 #define X first
9 #define Y second
10 #define PB emplace_back
11 #define pll pair<long, long>
12 using namespace std;
13 const int MXN = 4e5+5;
14
15 void sol(){}
16 int main(){
17     int t=1;
18     cin >> t;
19     while(t--){
20         sol(); } }

```

1.3 Common Sense

陣列過大時本機的指令：

```

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

```

2 flow

2.1 MinCostFlow

2.2 Dinic

求最大流 $O(N^2 E)$, 求二分最大匹配 $O(E\sqrt{N})$

```

2 dinic.init(n, st, en) => 0-base
2 dinic.addEdge(u, v, f) => u -> v, flow f units
2 dinic.run() => return max flow from st to en
2 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()。
3 範例 code 待補

```

```

1 const int MXN = 10005;
2 struct Dinic{
3     struct Edge{ ll v, f, re; };
4     int n, s, t, lvl[MXN];
5     vector<Edge> e[MXN];
6     void init(int _n, int _s, int _t){
7         n = _n; s = _s; t = _t;
8         for(int i = 0; i < n; ++i) e[i].clear(); }
9     void addEdge(int u, int v, ll f = 1){
10        e[u].push_back({v, f, e[v].size()});
11        e[v].push_back({u, 0, e[u].size() - 1}); }
12    bool bfs(){
13        memset(lvl, -1, n * 4);
14        queue<int> q;
15        q.push(s);
16        lvl[s] = 0;
17        while(!q.empty()){
18            int u = q.front(); q.pop();
19            for(auto &i : e[u])
20                if(i.f > 0 && lvl[i.v] == -1)
21                    lvl[i.v] = lvl[u] + 1, q.push(i.v); }
22        return lvl[t] != -1; }
23    ll dfs(int u, ll nf){
24        if(u == t) return nf;
25        ll res = 0;
26        for(auto &i : e[u])
27            if(i.f > 0 && lvl[i.v] == lvl[u] + 1){
28                int tmp = dfs(i.v, min(nf, i.f));
29                res += tmp, nf -= tmp, i.f -= tmp;
30                e[i.v][i.re].f += tmp;
31                if(nf == 0) return res; }
32        if(!res) lvl[u] = -1;
33        return res; }
34    ll run(ll res){
35        while(bfs()) res += dfs(s, LLINF);
36        return res; } }

```

2.3 Hungarian

2.4 Kuhn Munkres 最大完美二分匹配

2.5 Directed MST

```

1 /* Edmond's algoirthm for Directed MST
2  * runs in O(VE) */
3 const int MAXV = 10010;
4 const int MAXE = 10010;
5 const int INF = 2147483647;
6 struct Edge{
7     int u, v, c;
8     Edge(int x=0, int y=0, int z=0) : u(x), v(y), c(z){}
9 };
10 int V, E, root;
11 Edge edges[MAXE];
12 inline int newV(){ return ++ V; }
13 inline void addEdge(int u, int v, int c)
14 { edges[++E] = Edge(u, v, c); }
15 bool con[MAXV];
16 int mnInW[MAXV], prv[MAXV], cyc[MAXV], vis[MAXV];
17 inline int DMST(){

```

```

18 fill(con, con+V+1, 0);
19 int r1 = 0, r2 = 0;
20 while(1){
21     fill(mnInW, mnInW+V+1, INF);
22     fill(prv, prv+V+1, -1);
23     REP(i, 1, E){
24         int u=edges[i].u, v=edges[i].v, c=edges[i].c;
25         if(u != v && v != root && c < mnInW[v])
26             mnInW[v] = c, prv[v] = u;
27     }
28     fill(vis, vis+V+1, -1);
29     fill(cyc, cyc+V+1, -1);
30     r1 = 0;
31     bool jf = 0;
32     REP(i, 1, V){
33         if(con[i]) continue;
34         if(prv[i] == -1 && i != root) return -1;
35         if(prv[i] > 0) r1 += mnInW[i];
36         int s;
37         for(s = i; s != -1 && vis[s] == -1; s = prv[s])
38             vis[s] = i;
39         if(s > 0 && vis[s] == i){
40             // get a cycle
41             jf = 1; int v = s;
42             do{
43                 cyc[v] = s, con[v] = 1;
44                 r2 += mnInW[v]; v = prv[v];
45             }while(v != s);
46             con[s] = 0;
47         }
48         if(!jf) break;
49         REP(i, 1, E){
50             int &u = edges[i].u;
51             int &v = edges[i].v;
52             if(cyc[v] > 0) edges[i].c -= mnInW[edges[i].v];
53             if(cyc[u] > 0) edges[i].u = cyc[edges[i].u];
54             if(cyc[v] > 0) edges[i].v = cyc[edges[i].v];
55             if(u == v) edges[i--] = edges[E--];
56         }
57     }
58     return r1+r2;
59 }

```

3 Math

3.1 Fast Pow & Inverse & Combination

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

```

1 ll fpow(ll a, ll b, ll m){
2     ll ret = 1;
3     a %= m;
4     while(b){
5         if(b&1) ret = ret * a % m;
6         a = a * a % m;
7         b >>= 1; }
8     return ret; }
9
10 ll fa[MXN], fi[MXN];
11 void init(){
12     fa[0] = 1;
13     for(ll i = 1; i < MXN; ++i)
14         fa[i] = fa[i-1] * i % MOD;
15     fi[MXN-1] = fpow(fa[MXN-1], MOD-2, MOD);
16     for(ll i = MXN-1; i > 0; --i)
17         fi[i-1] = fi[i] * i % MOD; }
18
19 ll c(ll a, ll b){
20     return fa[a] * fi[b] % MOD * fi[a-b] % MOD; }

```

3.2 FFT

```

1 const int MAXN = 262144;
2 typedef long double ld;
3 typedef complex<ld> cplx; //real() ,imag()
4 const ld PI = acos(-1);
5 const cplx I(0, 1);
6 cplx omega[MAXN+1];
7 void pre_fft(){
8     for(int i=0; i<=MAXN; i++)
9         omega[i] = exp(i * 2 * PI / MAXN * I);

```

```

10 }
11 void fft(int n, cplx a[], bool inv=false){
12     int basic = MAXN / n;
13     int theta = basic;
14     for (int m = n; m >= 2; m >>= 1) {
15         int mh = m >> 1;
16         for (int i = 0; i < mh; i++) {
17             cplx w = omega[inv ? MAXN-(i*theta%MAXN)
18                             : i*theta%MAXN];
19             for (int j = i; j < n; j += m) {
20                 int k = j + mh;
21                 cplx x = a[j] - a[k];
22                 a[j] += a[k];
23                 a[k] = w * x;
24             }
25             theta = (theta * 2) % MAXN;
26         }
27         int i = 0;
28         for (int j = 1; j < n - 1; j++) {
29             for (int k = n >> 1; k > (i ^ k); k >>= 1);
30             if (j < i) swap(a[i], a[j]);
31         }
32         if(inv) for (i = 0; i < n; i++) a[i] /= n;
33     }
34     cplx arr[MAXN+1];
35     inline void mul(int _n, ll a[], int _m, ll b[], ll ans[])
36     {
37         int n=1, sum=_n+_m-1;
38         while(n<sum)
39             n<<=1;
40         for(int i=0; i<n; i++)
41         {
42             double x=(i<_n?a[i]:0), y=(i<_m?b[i]:0);
43             arr[i]=complex<double>(x+y, x-y);
44         }
45         fft(n, arr);
46         for(int i=0; i<n; i++)
47             arr[i]=arr[i]*arr[i];
48         fft(n, arr, true);
49         for(int i=0; i<sum; i++){
50             ans[i]=(long long int)(arr[i].real()/4 +
51                                   (arr[i].real() > 0 ? 0.5 : -0.5));
52         }
53     }

```

3.3 Miller Rabin

```

1 // n < 4,759,123,141      3 : 2, 7, 61
2 // n < 1,122,004,669,633  4 : 2, 13, 23, 1662803
3 // n < 3,474,749,660,383  6 : pimes <= 13
4 // n < 2^64               7 :
5 // 2, 325, 9375, 28178, 450775, 9780504, 1795265022
6 // Make sure testing integer is in range [2, n-2] if
7 // you want to use magic.
8 LL magic[]={}
9 bool witness(LL a, LL n, LL u, int t){
10     if(!a) return 0;
11     LL x=mpow(a, u, n);
12     for(int i=0; i<t; i++) {
13         LL nx=mul(x, x, n);
14         if(nx==1&&x!=1&&x!=n-1) return 1;
15         x=nx;
16     }
17     return x!=1;
18 }
19 bool miller_rabin(LL n) {
20     int s=(magic number size)
21     // iterate s times of witness on n
22     if(n<2) return 0;
23     if(!(n&1)) return n==2;
24     ll u=n-1; int t=0;
25     // n-1 = u*2^t
26     while(!(u&1)) u>>=1, t++;
27     while(s--){
28         LL a=magic[s]%n;
29         if(witness(a, n, u, t)) return 0;
30     }
31     return 1;
32 }

```

3.4 Chinese Remainder

```

1 LL x[N],m[N];
2 LL CRT(LL x1, LL m1, LL x2, LL m2) {
3     LL g = __gcd(m1, m2);
4     if((x2 - x1) % g) return -1; // no sol
5     m1 /= g; m2 /= g;
6     pair<LL,LL> p = gcd(m1, m2);
7     LL lcm = m1 * m2 * g;
8     LL res = p.first * (x2 - x1) * m1 + x1;
9     return (res % lcm + lcm) % lcm;
10 }
11 LL solve(int n){ // n>=2, be careful with no solution
12     LL res=CRT(x[0],m[0],x[1],m[1]),p=m[0]/__gcd(m[0],m
13         [1])*m[1];
14     for(int i=2;i<n;i++){
15         res=CRT(res,p,x[i],m[i]);
16         p=p/__gcd(p,m[i])*m[i];
17     }
18     return res;

```

3.5 Pollard Rho

```

1 // does not work when n is prime
2 LL f(LL x, LL mod){ return add(mul(x,x,mod),1,mod); }
3 LL pollard_rho(LL n) {
4     if(!(n&1)) return 2;
5     while(true){
6         LL y=2, x=rand()%(n-1)+1, res=1;
7         for(int sz=2; res==1; sz*=2) {
8             for(int i=0; i<sz && res<=1; i++) {
9                 x = f(x, n);
10                res = __gcd(abs(x-y), n);
11            }
12            y = x;
13        }
14        if (res!=0 && res!=n) return res;
15    } }

```

3.6 Primes

```

1 /* 12721, 13331, 14341, 75577, 123457, 222557, 556679
2 * 999983, 1097774749, 1076767633, 100102021, 999997771
3 * 1001010013, 1000512343, 987654361, 999991231
4 * 999888733, 98789101, 987777733, 999991921, 1010101333
5 * 1010102101, 1000000000039, 100000000000037
6 * 2305843009213693951, 4611686018427387847
7 * 9223372036854775783, 18446744073709551557 */
8 int mu[ N ], p_tbl[ N ];
9 vector<int> primes;
10 void sieve() {
11     mu[ 1 ] = p_tbl[ 1 ] = 1;
12     for( int i = 2 ; i < N ; i ++ ){
13         if( !p_tbl[ i ] ){
14             p_tbl[ i ] = i;
15             primes.push_back( i );
16             mu[ i ] = -1;
17         }
18         for( int p : primes ){
19             int x = i * p;
20             if( x >= M ) break;
21             p_tbl[ x ] = p;
22             mu[ x ] = -mu[ i ];
23             if( i % p == 0 ){
24                 mu[ x ] = 0;
25                 break;
26             }
27         }
28     }
29     vector<int> fac( int x ){
30         vector<int> fac{ 1 };
31         while( x > 1 ){
32             int fn = SZ(fac), p = p_tbl[ x ], pos = 0;
33             while( x % p == 0 ){
34                 x /= p;
35                 for( int i = 0 ; i < fn ; i ++ )
36                     fac.PB( fac[ pos ++ ] * p );
37             }
38         }
39         return fac;
40     }
41 }

```

4 Geometry

4.1 definition

```

1 typedef long double ld;
2 const ld eps = 1e-8;
3 int dcmp(ld x) {
4     if(abs(x) < eps) return 0;
5     else return x < 0 ? -1 : 1;
6 }
7 struct Pt {
8     ld x, y;
9     Pt(ld _x=0, ld _y=0):x(_x), y(_y) {}
10 }
11 Pt operator+(const Pt &a) const {
12     return Pt(x+a.x, y+a.y);
13 }
14 Pt operator-(const Pt &a) const {
15     return Pt(x-a.x, y-a.y);
16 }
17 Pt operator*(const ld &a) const {
18     return Pt(x*a, y*a);
19 }
20 Pt operator/(const ld &a) const {
21     return Pt(x/a, y/a);
22 }
23 ld operator*(const Pt &a) const {
24     return x*a.x + y*a.y;
25 }
26 ld operator^(const Pt &a) const {
27     return x*a.y - y*a.x;
28 }
29 bool operator<(const Pt &a) const {
30     return x < a.x || (x == a.x && y < a.y);
31     //return dcmp(x-a.x) < 0 || (dcmp(x-a.x) == 0 &&
32         dcmp(y-a.y) < 0);
33 }
34 bool operator==(const Pt &a) const {
35     return dcmp(x-a.x) == 0 && dcmp(y-a.y) == 0;
36 }
37 ld norm2(const Pt &a) {
38     return a*a;
39 }
40 ld norm(const Pt &a) {
41     return sqrt(norm2(a));
42 }
43 Pt perp(const Pt &a) {
44     return Pt(-a.y, a.x);
45 }
46 Pt rotate(const Pt &a, ld ang) {
47     return Pt(a.x*cos(ang)-a.y*sin(ang), a.x*sin(ang)+a.y
48         *cos(ang));
49 }
50 struct Line {
51     Pt s, e, v; // start, end, end-start
52     ld ang;
53     Line(Pt _s=Pt(0, 0), Pt _e=Pt(0, 0)):s(_s), e(_e) { v
54         = e-s; ang = atan2(v.y, v.x); }
55     bool operator<(const Line &L) const {
56         return ang < L.ang;
57     }
58 }
59 struct Circle {
60     Pt o; ld r;
61     Circle(Pt _o=Pt(0, 0), ld _r=0):o(_o), r(_r) {}

```

4.2 Convex Hull

```

1 double cross(Pt o, Pt a, Pt b){
2     return (a-o) ^ (b-o);
3 }
4 vector<Pt> convex_hull(vector<Pt> pt){
5     sort(pt.begin(),pt.end());
6     int top=0;
7     vector<Pt> stk(2*pt.size());
8     for (int i=0; i<(int)pt.size(); i++){
9         while (top >= 2 && cross(stk[top-2],stk[top-1],pt[i]
10             ]) <= 0)
11             top--;
12         stk[top++] = pt[i];
13     }
14     for (int i=pt.size()-2, t=top+1; i>=0; i--){

```

```

14 while (top >= t && cross(stk[top-2],stk[top-1],pt[i22
    ]) <= 0)
15     top--;
16     stk[top++] = pt[i];
17 }
18 stk.resize(top-1);
19 return stk;
20 }

```

4.3 Scan line

```

1 struct node1 {
2     double l, r;
3     double sum;
4 } cl[maxn << 3];
5 struct node2 {
6     double x, y1, y2;
7     int flag;
8 } p[maxn << 3];
9 bool cmp(node2 a, node2 b) { return a.x < b.x; }
10 void pushup(int rt) {
11     if (lazy[rt] > 0) cl[rt].sum = cl[rt].r - cl[rt].l;
12     else cl[rt].sum = cl[rt * 2].sum + cl[rt * 2 + 1].sum;
13 }
14 void build(int rt, int l, int r) {
15     if (r - l > 1) {
16         cl[rt].l = s[l];
17         cl[rt].r = s[r];
18         build(rt * 2, l, (l + r) / 2);
19         build(rt * 2 + 1, (l + r) / 2, r);
20         pushup(rt);
21     } else {
22         cl[rt].l = s[l];
23         cl[rt].r = s[r];
24         cl[rt].sum = 0;
25     }
26 }
27 void update(int rt, double y1, double y2, int flag) {
28     if (cl[rt].l == y1 && cl[rt].r == y2) {
29         lazy[rt] += flag;
30         pushup(rt);
31         return;
32     } else {
33         if (cl[rt * 2].r > y1) update(rt * 2, y1, min(cl[rt * 2].r, y2), flag);
34         if (cl[rt * 2 + 1].l < y2) update(rt * 2 + 1, max(cl[rt * 2 + 1].l, y1), y2, flag);
35         pushup(rt);
36     }
37 }

```

5 Tree

5.1 LCA

求樹上兩點的最低共同祖先

$lca.init(n) \Rightarrow 0\text{-base}$

$lca.addEdge(u, v) \Rightarrow u \leftrightarrow v$

$lca.build(root, root) \Rightarrow O(n \lg n)$

$lca.qlca(u, v) \Rightarrow O(\lg n)$ u, v 的 LCA

$lca.qdis(u, v) \Rightarrow O(\lg n)$ 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

```

1 const int MXN = 5e5+5;
2 struct LCA{
3     int n, lgn, ti = 0;
4     int anc[MXN][24], in[MXN], out[MXN];
5     vector<int> g[MXN];
6     void init(int _n){
7         n = _n, lgn = __lg(n) + 5;
8         for(int i = 0; i < n; ++i) g[i].clear();
9         void addEdge(int u, int v){ g[u].PB(v), g[v].PB(u); }
10        void build(int u, int f){
11            in[u] = ti++;
12            int cur = f;
13            for(int i = 0; i < lgn; ++i)
14                anc[u][i] = cur, cur = anc[cur][i];
15            for(auto i : g[u]) if(i != f) build(i, u);
16            out[u] = ti++;
17        }
18        bool isanc(int a, int u){
19            return in[a] <= in[u] && out[a] >= out[u];
20        }
21        int qlca(int u, int v){
22            if(isanc(u, v)) return u;
23            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

```

1 const int MAXN = 100010;
2 struct DominatorTree{
3     #define REP(i,s,e) for(int i=(s);i<=(e);i++)
4     #define REPD(i,s,e) for(int i=(s);i>=(e);i--)
5     int n, m, s;
6     vector<int> g[ MAXN ], pred[ MAXN ];
7     vector<int> cov[ MAXN ];
8     int dfn[ MAXN ], nfd[ MAXN ], ts;
9     int par[ MAXN ]; //idom[u] s到u的最後一個必經點
10    int sdom[ MAXN ], idom[ MAXN ];
11    int mom[ MAXN ], mn[ MAXN ];
12    inline bool cmp( int u, int v )
13    { return dfn[ u ] < dfn[ v ]; }
14    int eval( int u ){
15        if( mom[ u ] == u ) return u;
16        int res = eval( mom[ u ] );
17        if( cmp( sdom[ mn[ mom[ u ] ] ], sdom[ mn[ u ] ] ) )
18            mn[ u ] = mn[ mom[ u ] ];
19        return mom[ u ] = res;
20    }
21    void init( int _n, int _m, int _s ){
22        ts = 0; n = _n; m = _m; s = _s;
23        REP( i, 1, n ) g[ i ].clear(), pred[ i ].clear();
24    }
25    void addEdge( int u, int v ){
26        g[ u ].push_back( v );
27        pred[ v ].push_back( u );
28    }
29    void dfs( int u ){
30        ts++;
31        dfn[ u ] = ts;
32        nfd[ ts ] = u;
33        for( int v : g[ u ] ) if( dfn[ v ] == 0 ){
34            par[ v ] = u;
35            dfs( v );
36        }
37    }
38    void build(){
39        REP( i, 1, n ){
40            dfn[ i ] = nfd[ i ] = 0;
41            cov[ i ].clear();
42            mom[ i ] = mn[ i ] = sdom[ i ] = i;
43        }
44        dfs( s );
45        REPD( i, n, 2 ){
46            int u = nfd[ i ];
47            if( u == 0 ) continue;
48            for( int v : pred[ u ] ) if( dfn[ v ] ){
49                eval( v );
50                if( cmp( sdom[ mn[ v ] ], sdom[ u ] ) )
51                    sdom[ u ] = sdom[ mn[ v ] ];
52            }
53            cov[ sdom[ u ] ].push_back( u );
54            mom[ u ] = par[ u ];
55            for( int w : cov[ par[ u ] ] ){
56                eval( w );
57                if( cmp( sdom[ mn[ w ] ], par[ u ] ) )
58                    idom[ w ] = mn[ w ];
59                else idom[ w ] = par[ u ];
60            }
61            cov[ par[ u ] ].clear();
62        }
63        REP( i, 2, n ){
64            int u = nfd[ i ];
65            if( u == 0 ) continue;
66            if( idom[ u ] != sdom[ u ] )

```

```

67     idom[ u ] = idom[ idom[ u ] ];
68 }
69 }
70 } domT;

```

6.2 MaximumClique 最大團

```

1 #define N 111
2 struct MaxClique{ // 0-base
3     typedef bitset<N> Int;
4     Int linkto[N] , v[N];
5     int n;
6     void init(int _n){
7         n = _n;
8         for(int i = 0 ; i < n ; i ++){
9             linkto[i].reset(); v[i].reset();
10        }
11    }
12    void addEdge(int a , int b)
13    { v[a][b] = v[b][a] = 1; }
14    int popcount(const Int& val)
15    { return val.count(); }
16    int lowbit(const Int& val)
17    { return val._Find_first(); }
18    int ans , stk[N];
19    int id[N] , di[N] , deg[N];
20    Int cans;
21    void maxclique(int elem_num, Int candi){
22        if(elem_num > ans){
23            ans = elem_num; cans.reset();
24            for(int i = 0 ; i < elem_num ; i ++){
25                cans[id[stk[i]]] = 1;
26            }
27            int potential = elem_num + popcount(candi);
28            if(potential <= ans) return;
29            int pivot = lowbit(candi);
30            Int smaller_candi = candi & (~linkto[pivot]);
31            while(smaller_candi.count() && potential > ans){
32                int next = lowbit(smaller_candi);
33                candi[next] = !candi[next];
34                smaller_candi[next] = !smaller_candi[next];
35                potential --;
36                if(next == pivot || (smaller_candi & linkto[next
37                    ]).count()){
38                    stk[elem_num] = next;
39                    maxclique(elem_num + 1, candi & linkto[next]);
40                }
41            }
42            int solve(){
43                for(int i = 0 ; i < n ; i ++){
44                    id[i] = i; deg[i] = v[i].count();
45                }
46                sort(id , id + n , [&](int id1, int id2){
47                    return deg[id1] > deg[id2]; });
48                for(int i = 0 ; i < n ; i ++){ di[id[i]] = i;
49                }
50                for(int i = 0 ; i < n ; i ++){
51                    for(int j = 0 ; j < n ; j ++){
52                        if(v[i][j]) linkto[di[i]][di[j]] = 1;
53                    }
54                }
55                Int cand; cand.reset();
56                for(int i = 0 ; i < n ; i ++){ cand[i] = 1;
57                }
58                ans = 1;
59                cans.reset(); cans[0] = 1;
60                maxclique(0, cand);
61                return ans;
62            }
63        }
64    } solver;

```

6.3 Strongly Connected Component

```

1 struct Scc{
2     int n, nScc, vst[MXN], bln[MXN];
3     vector<int> E[MXN], rE[MXN], vec;
4     void init(int _n){
5         n = _n;
6         for (int i=0; i<MXN; i++){
7             E[i].clear(), rE[i].clear();
8         }
9     }
10    void addEdge(int u, int v){
11        E[u].PB(v); rE[v].PB(u);
12    }
13    void DFS(int u){

```

```

13     vst[u]=1;
14     for (auto v : E[u]) if (!vst[v]) DFS(v);
15     vec.PB(u);
16 }
17 void rDFS(int u){
18     vst[u] = 1; bln[u] = nScc;
19     for (auto v : rE[u]) if (!vst[v]) rDFS(v);
20 }
21 void solve(){
22     nScc = 0;
23     vec.clear();
24     FZ(vst);
25     for (int i=0; i<n; i++){
26         if (!vst[i]) DFS(i);
27         reverse(vec.begin(),vec.end());
28         FZ(vst);
29         for (auto v : vec)
30             if (!vst[v]){
31                 rDFS(v); nScc++;
32             }
33     }
34 }

```

7 String

7.1 KMP

```

1 /*
2 len-failure[k]:
3 在k結尾的情況下，這個子字串可以由開頭
4 長度為(len-failure[k])的部分重複出現來表達
5
6 failure[k]:
7 failure[k]為次長相同前綴後綴
8 如果我們不只想求最多，而且以0-base做為考量
9 ，那可能的長度由大到小會是
10 failuer[k]、failure[failuer[k]-1]
11 、failure[failure[failuer[k]-1]-1]..
12 直到有值為0為止
13 */
14 int failure[MXN];
15 void KMP(string& t, string& p)
16 {
17     if (p.size() > t.size()) return;
18     for (int i=1, j=failure[0]=-1; i<p.size(); ++i)
19     {
20         while (j >= 0 && p[j+1] != p[i])
21             j = failure[j];
22         if (p[j+1] == p[i]) j++;
23         failure[i] = j;
24     }
25     for (int i=0, j=-1; i<t.size(); ++i)
26     {
27         while (j >= 0 && p[j+1] != t[i])
28             j = failure[j];
29         if (p[j+1] == t[i]) j++;
30         if (j == p.size()-1)
31         {
32             cout << i - p.size() + 1<<" ";
33             j = failure[j];
34         }
35     }
36 }

```

7.2 Aho-Corasick

```

1 struct ACautomata{
2     struct Node{
3         int cnt,i;
4         Node *go[26], *fail, *dic;
5         Node (){
6             cnt = 0; fail = 0; dic=0;
7             memset(go,0,sizeof(go));
8         }
9     }pool[1048576],*root;
10    int nMem,n_pattern;
11    Node* new_Node(){
12        pool[nMem] = Node();
13        return &pool[nMem++];
14    }
15    void init() {nMem=0;root=new_Node();n_pattern=0;}
16    void add(const string &str) { insert(root,str,0); }
17    void insert(Node *cur, const string &str, int pos){

```



```

18 for(int i=pos;i<str.size();i++){
19     if(!cur->go[str[i]-'a'])
20         cur->go[str[i]-'a'] = new_Node();
21     cur=cur->go[str[i]-'a'];
22 }
23 cur->cnt++; cur->i=n_pattern++;
24 }
25 void make_fail(){
26     queue<Node*> que;
27     que.push(root);
28     while (!que.empty()){
29         Node* fr=que.front(); que.pop();
30         for (int i=0; i<26; i++){
31             if (fr->go[i]){
32                 Node *ptr = fr->fail;
33                 while (ptr && !ptr->go[i]) ptr = ptr->fail;
34                 fr->go[i]->fail=ptr=(ptr?ptr->go[i]:root);
35                 fr->go[i]->dic=(ptr->cnt?ptr:ptr->dic);
36                 que.push(fr->go[i]);
37             } } } }
38 void query(string s){
39     Node *cur=root;
40     for(int i=0;i<(int)s.size();i++){
41         while(cur&&!cur->go[s[i]-'a']) cur=cur->fail;
42         cur=(cur?cur->go[s[i]-'a']:root);
43         if(cur->i>=0) ans[cur->i]++;
44         for(Node *tmp=cur->dic;tmp;tmp=tmp->dic)
45             ans[tmp->i]++;
46     } } // ans[i] : number of occurrence of pattern i
47 }AC;

```

7.3 Z Value

```

1 char s[MAXN];
2 int len,z[MAXN];
3 void Z_value() { //z[i] = lcp(s[1...],s[i...])
4     int i,j,left,right;
5     left=right=0; z[0]=len;
6     for(i=1;i<len;i++) {
7         j=max(min(z[i-left],right-i),0);
8         for(;i+j<len&&s[i+j]==s[j];j++);
9         z[i]=j;
10        if(i+z[i]>right) {
11            right=i+z[i];
12            left=i;
13        } } }

```

7.4 sa

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 #define N 100010
4 char T[ N ];
5 int n , RA[ N ] , tempRA[ N ] , SA[ N ] , tempSA[ N ] , c44
   [ N ];
6 void countingSort( int k ){
7     int i , sum , maxi = max( 300 , n ) ;
8     memset( c , 0 , sizeof c ) ;
9     for ( i = 0 ; i < n ; i ++ ) c[ ( i + k < n ) ? RA[i]
10        + k ] : 0 ] ++ ;
11     for ( i = sum = 0 ; i < maxi ; i ++ ) { int t = c[i]
12        ; c[i] = sum ; sum += t ; }
13     for ( i = 0 ; i < n ; i ++ )
14         tempSA[ c[ ( SA[ i ] + k < n ) ? RA[ SA[ i ] ] + k ]
15            : 0 ] ++ ] = SA[ i ] ;
16     for ( i = 0 ; i < n ; i ++ ) SA[ i ] = tempSA[ i ] ;
17 }
18 void constructSA(){
19     int r;
20     for ( int i = 0 ; i < n ; i ++ ) RA[ i ] = T[ i ] - '
21        ' ;
22     for ( int i = 0 ; i < n ; i ++ ) SA[ i ] = i ;
23     for ( int k = 1 ; k < n ; k <= 1 ) {
24         countingSort( k ) ; countingSort( 0 ) ;
25         tempRA[ SA[ 0 ] ] = r = 0 ;
26         for ( int i = 1 ; i < n ; i ++ )
27             tempRA[ SA[ i ] ] = ( RA[ SA[ i ] ] == RA[ SA[ i -
28                1 ] ] && RA[ SA[ i ] + k ] == RA[ SA[ i -
29                1 ] + k ] ) ? r : ++ r ;
30         for ( int i = 0 ; i < n ; i ++ ) RA[ i ] = tempRA[
31            i ] ;
32     }
33 }

```

```

26 }
27 int main() {
28     n = (int)strlen( gets( T ) ) ;
29     T[ n ++ ] = '.' ; // important bug fix!
30     constructSA() ;
31     return 0;
32 }

```

7.5 ParTree

```

1 // len[s]是對應的回文長度
2 // num[s]是有幾個回文後綴
3 // cnt[s]是這個回文字字串在整個字串中的出現次數
4 // fail[s]是他長度次長的回文後綴，aba的fail是a
5 const int MXN = 1000010;
6 struct PalT{
7     int nxt[MXN][26],fail[MXN],len[MXN];
8     int tot,lst,n,state[MXN],cnt[MXN],num[MXN];
9     int diff[MXN],sfail[MXN],fac[MXN],dp[MXN];
10    char s[MXN]={-1};
11    int newNode(int l,int f){
12        len[tot]=l,fail[tot]=f,cnt[tot]=num[tot]=0;
13        memset(nxt[tot],0,sizeof(nxt[tot]));
14        diff[tot]=(l>0?l-len[f]:0);
15        sfail[tot]=(l>0&&diff[tot]==diff[f]?sfail[f]:f);
16        return tot++;
17    }
18    int getfail(int x){
19        while(s[n-len[x]-1]!=s[n]) x=fail[x];
20        return x;
21    }
22    int getmin(int v){
23        dp[v]=fac[n-len[sfail[v]]-diff[v]];
24        if(diff[v]==diff[fail[v]])
25            dp[v]=min(dp[v],dp[fail[v]]);
26        return dp[v]+1;
27    }
28    int push(){
29        int c=s[n]-'a',np=getfail(lst);
30        if(!(lst=nxt[np][c])){
31            lst=newNode(len[np]+2,nxt[getfail(fail[np])][c]);
32            nxt[np][c]=lst; num[lst]=num[fail[lst]]+1;
33        }
34        fac[n]=n;
35        for(int v=lst;len[v]>0;v=sfail[v])
36            fac[n]=min(fac[n],getmin(v));
37        return ++cnt[lst],lst;
38    }
39    void init(const char *_s){
40        tot=lst=n=0;
41        newNode(0,1),newNode(-1,1);
42        for(;s[n];) s[n+1]=s[n],++n,state[n-1]=push();
43        for(int i=tot-1;i>1;i--) cnt[fail[i]]+=cnt[i];
44    }
45 }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

```

1 struct Treap{
2     Treap *l, *r;
3     int pri, sz, rev;
4     ll val, sum;
5     Treap(int _val): l(nullptr), r(nullptr),
6         pri(rand()), sz(1), rev(0),
7         val(_val), sum(_val){} };
8
9 ll gsz(Treap *x){ return x ? x->sz : 0; }
10 ll gsum(Treap *x){ return x ? x->sum : 0; }
11
12 Treap* pull(Treap *x){
13     x->sz = gsz(x->l) + gsz(x->r) + 1;
14     x->sum = x->val + gsum(x->l) + gsum(x->r);
15     return x; }

```

```
16 void push(Treap *x){
17     if(x->rev){
18         swap(x->l, x->r);
19         if(x->l) x->l->rev ^= 1;
20         if(x->r) x->r->rev ^= 1;
21         x->rev = 0; } }
22
23 Treap* merge(Treap* a, Treap* b){
24     if(!a || !b) return a ? a : b;
25     push(a), push(b);
26     if(a->pri > b->pri){
27         a->r = merge(a->r, b);
28         return pull(a); }
29     else{
30         b->l = merge(a, b->l);
31         return pull(b); } }
32
33 void split(Treap *x, int k, Treap *&a, Treap *&b){
34     if(!x) a = b = nullptr;
35     else{
36         push(x);
37         if(x->val <= k) a = x, split(x->r, k, a->r, b);
38         else b = x, split(x->l, k, a, b->l);
39         pull(x); } }
40
41 void kth(Treap *x, int k, Treap *&a, Treap *&b){
42     if(!x) a = b = nullptr;
43     else{
44         push(x);
45         if(gsz(x->l) < k)
46             a = x, kth(x->r, k - gsz(x->l) - 1, a->r, b);
47         else b = x, kth(x->l, k, a, b->l);
48         pull(x); } }
```

9 Others