

BEIJING NORMAL UNIVERSITY

Template

Pray Never

2025 年 4 月 19 日

目录

1	data structure	3
1.1	01tries.cpp	3
1.2	persistant-segement-tree.cpp	4
1.3	trie.cpp	4
2	geometry	6
2.1	convex.cpp	6
3	graph	7
3.1	Dinic.cpp	7
3.2	EK.cpp	8
3.3	ISAP.cpp	9
3.4	Johnson.cpp	11
3.5	LCA.cpp	13
3.6	lcd.cpp	14
3.7	lcd2.cpp	17
3.8	tarjan.cpp	18
3.9	tree divide and conquer.cpp	19
4	heading	21
4.1	debug.h	21
4.2	duipai.cpp	21
4.3	heading.cpp	22
5	math	23
5.1	Eratost.cpp	23
5.2	Euler.cpp	23
5.3	FFT.cpp	24
5.4	Gause _X or.cpp	24
5.5	NTT.cpp	25
5.6	pollar-rho.cpp	27
5.7	Xor _b ase.cpp	29
6	sort	30
6.1	merge-sort.cpp	30

7	string	31
7.1	<i>AC_automation.cpp</i>	31
7.2	KMP.cpp	32
7.3	Manacher.cpp	33
7.4	PAM.cpp	34
7.5	<i>z_function.cpp</i>	35

1 data structure

1.1 01tries.cpp

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  typedef long long ll;
4  const int N = 1e5 + 10;
5  vector<pair<int, int> > v[N];
6  int cnt;
7  int node[5 * N][2];
8  bool isval[5 * N];
9  void insert(int val){
10     int p = 0;
11     stack<int> s;
12     for(int i = 0; i < 31; ++i){
13         s.push(val % 2);
14         val /= 2;
15     }
16     for(int i = 0; i < 31; i++){
17         int c = s.top();
18         s.pop();
19         if(!node[p][c]) node[p][c] = ++cnt;
20         p = node[p][c];
21     }
22     isval[p]++;
23 }
24 int a[N];
25 void dfs(int pos, int far){
26     for(auto[u, w]: v[pos]){
27         if(u == far);
28         continue;
29         a[u] = a[pos] ^ w;
30         dfs(u, pos);
31     }
32 }
33 int query(int val){
34
35 }
36 int main(){
37     int n;
38     cin >> n;
39     for(int i = 1; i < n; ++i){
40         int x, y, z;
41         cin >> x >> y >> z;
42         v[x].emplace_back(y, z);
43         v[y].emplace_back(x, z);
44     }
45     dfs(1, 0);
46     int maxy = 0;
47     for(int i = 1; i <= n; ++i){
48         maxy = max(maxy, query(a[i]));

```

```

49     }
50 }

```

1.2 persistant-segement-tree.cpp

```

1  #ifdef IGNORE_THIS_FILE
2  struct PTR{
3      int n, tot;
4      vector<int> ls, rs, sum, root;
5      PTR(int _n){
6          n = _n;
7          tot = 0;
8          ls = vector<int>(n << 6), rs = vector<int>(n << 6), sum = vector<int>(n << 6);
9          root = vector<int>(n + 1);
10     }
11     int update(int q, int l, int r, int rt){
12         int dir = ++tot;
13         ls[dir] = ls[rt], rs[dir] = rs[rt];
14         if(l == r){
15             sum[dir] = sum[rt] + 1;
16             return dir;
17         }
18         int mid = (l + r) >> 1;
19         if(q <= mid)
20             ls[dir] = update(q, l, mid, ls[rt]);
21         else
22             rs[dir] = update(q, mid + 1, r, rs[rt]);
23         sum[dir] = sum[ls[dir]] + sum[rs[dir]];
24         return dir;
25     };
26     int query(int q, int l, int r, int tl, int tr){
27         if(l == r)
28             return l;
29         int mid = (l + r) >> 1;
30         int sum1 = sum[ls[tr]] - sum[ls[tl]];
31         if(q <= sum1)
32             return query(q, l, mid, ls[tl], ls[tr]);
33         else
34             return query(q - sum1, mid + 1, r, rs[tl], rs[tr]);
35     };
36 };
37
38 #endif
39
40

```

1.3 trie.cpp

```

1  #ifdef IGNORE_THIS_FILE
2  struct Trie {
3      static const int N = 5000100; // 预设最大节点数

```

```
4     int cnt = 0;
5     vector<vector<int>> node{N, vector<int>(26)};
6     vector<int> isval{N};
7
8     void insert(string& s, int len) {
9         int p = 0;
10        for(int i = 0; i < len; ++i){
11            int c = s[i] - 'a';
12            if(!node[p][c]){
13                node[p][c] = ++cnt;
14            }
15            p = node[p][c];
16        }
17        ++isval[p];
18    }
19
20    int find(string& s, int len) {
21        int p = 0;
22        for(int i = 0; i < len; ++i){
23            int c = s[i] - 'a';
24            if(!node[p][c]) return 0;
25            p = node[p][c];
26        }
27        return isval[p];
28    }
29 };
30 #endif
```

2 geometry

2.1 convex.cpp

```

1  #ifdef IGNORE_THIS_FILE
2      struct Point {
3          ll x,y;
4      };
5      auto andrew = [](vector<Point>& p) -> vector<Point> {    // 传入下标从零开始的点数组，返回凸包数组
6          auto cmp = [](Point &a, Point &b) -> bool {
7              if(a.x != b.x) return a.x < b.x;
8              return a.y < b.y;
9          };
10         auto cross = [](Point &u, Point &v, Point &w) -> bool {
11             ll x1 = u.x - v.x, y1 = u.y - v.y;
12             ll x2 = w.x - v.x, y2 = w.y - v.y;
13             return x1 * y2 - x2 * y1 > 0; //如果不希望在凸包的边上有输入点。把 > 改成 >=
14         };
15         sort(p.begin(), p.end(), cmp);
16         int n = p.size(), m = 0;
17         vector<Point> res(n + 1);
18         for(int i = 0; i < n; ++i){
19             while(m > 1 && !cross(res[m - 1], res[m - 2], p[i])) --m;
20             res[m++] = p[i];
21         }
22         int kk = m;
23         for(int i = n - 2; i >= 0; i--){
24             while(m > kk && !cross(res[m - 1], res[m - 2], p[i])) --m;
25             res[m++] = p[i];
26         }
27         if(n > 1) --m; //凸包有 m 个顶点
28         res.erase(res.begin() + m, res.end());
29         return res;
30     };
31 #endif

```

3 graph

3.1 Dinic.cpp

```

1  #ifndef IGNORE_THIS_FILE
2  constexpr int N = 500;
3  constexpr ll INF = 0x3fffffffffffffff;
4  struct edge {
5      int from, to;
6      ll can_flow;
7      edge(int f, int t, ll can_f) : from(f), to(t), can_flow(can_f){};
8  };
9  struct Dinic {
10     vector<edge> e;
11     vector<int> G[N];
12     int dep[N], cur[N];
13     int n, m;
14     void init(int n) {
15         this->n = n;
16         for(int i = 0; i <= n; ++i) G[i].clear();
17         e.clear();
18     }
19
20     void addedge(int from, int to, ll cap) {
21         e.emplace_back(from, to, cap);
22         e.emplace_back(to, from, 0);
23         m = e.size();
24         G[from].push_back(m - 2);
25         G[to].push_back(m - 1);
26     }
27
28     bool bfs(int S, int T) {
29         queue<int> q;
30         memset(dep, 0, sizeof(int) * (n + 1));
31
32         dep[S] = 1;
33         q.push(S);
34         while (q.size()) {
35             int u = q.front();
36             q.pop();
37             for(int i = 0; i < G[u].size(); ++i){
38                 int id = G[u][i];
39                 const auto&[from, to, can_flow] = e[id];
40                 if ((!dep[to]) && can_flow) {
41                     dep[to] = dep[u] + 1;
42                     q.push(to);
43                 }
44             }
45         }
46         return dep[T];
47     }
48

```



```

49     ll dfs(int u, int T, ll last_flow) {
50         if (u == T || !last_flow) return last_flow;
51         ll flow = 0;
52         ll f;
53         for (int& i = cur[u]; i < G[u].size(); ++i) {
54             int id = G[u][i];
55             const auto&[from, to, can_flow] = e[id];
56             if (dep[u] + 1 == dep[to] && (f = dfs(to, T, min(last_flow, can_flow))) > 0) {
57                 e[id].can_flow -= f;
58                 e[id ^ 1].can_flow += f;
59                 flow += f;
60                 last_flow -= f;
61                 if (!last_flow) break;
62             }
63         }
64         return flow;
65     }
66
67     ll dinic(int S, int T) {
68         ll maxflow = 0;
69         while (bfs(S, T)) {
70             memset(cur, 0, sizeof(cur));
71             maxflow += dfs(S, T, INF);
72         }
73         return maxflow;
74     }
75 };
76 #endif

```

3.2 EK.cpp

```

1  #ifdef IGNORE_THIS_FILE
2  constexpr int N = 250;
3  constexpr ll INF = 0x3fffffffffffffff;
4  struct Edge {
5      int from, to;
6      ll can_flow;
7      Edge(int u, int v, ll can_f) : from(u), to(v), can_flow(can_f) {}
8  };
9  struct EK {
10     int n, m;           // n: 点数, m: 边数
11     vector<Edge> e;      // e: 所有边的集合
12     vector<int> G[N];    // G: 点 x -> x 的所有边在 e 中的下标
13     ll a[N];            // a: 点 x -> BFS 过程中最近接近点 x 的边给它的最大流
14     int p[N];           // p: 点 x -> BFS 过程中最近接近点 x 的边
15
16     void init(int n) {
17         for (int i = 0; i <= n; i++) G[i].clear();
18         e.clear();
19     }
20
21     void AddEdge(int from, int to, ll cap) {

```

```

22     e.emplace_back(from, to, cap);
23     e.emplace_back(to, from, 0);
24     m = e.size();
25     G[from].push_back(m - 2);
26     G[to].push_back(m - 1);
27 }
28
29 ll Maxflow(int s, int t) {
30     ll max_flow = 0;
31     while(true) {
32         memset(a, 0, sizeof(a));
33         queue<int> qu;
34         qu.push(s);
35         a[s] = INF;
36         while (!qu.empty()) {
37             int u = qu.front();
38             qu.pop();
39             for (int i: G[u]) { // 遍历以 u 作为起点的边
40                 const auto& [from, to, can_flow] = e[i];
41                 if (!a[to] && can_flow) {
42                     p[to] = i; // e[i] 是最近接近点 to 的边
43                     a[to] = min(a[u], can_flow); // 最近接近点 to 的边赋给它的流
44                     qu.push(to);
45                 }
46             }
47             if (a[t]) break; // 如果汇点接受到了流, 就退出 BFS
48         }
49         if (!a[t])
50             break; // 如果汇点没有接受到流, 说明源点和汇点不在同一个连通分量上
51         for (int u = t; u != s; u = e[p[u]].from) { // 通过 u 追寻 BFS 过程中 s -> t 的路径
52             e[p[u]].can_flow -= a[t]; // 减少路径上边的 can_flow 值
53             e[p[u] ^ 1].can_flow += a[t]; // 增加反向路径的 can_flow 值
54         }
55         max_flow += a[t];
56     }
57     return max_flow;
58 }
59 };
60 #endif

```

3.3 ISAP.cpp

```

1  #ifndef IGNORE_THIS_FILE
2  const int N = 1000;
3  const ll INF = 0x3ffffffffffff;
4  struct Edge {
5      int from, to;
6      ll can_flow;
7      Edge(int f, int t, ll cap) : from(f), to(t), can_flow(cap){};
8  };
9  struct ISAP {
10     int n, m;

```

```

11     int s, t;
12     vector<int> G[N];
13     vector<Edge> e;
14     int dep[N], gap[N], cur[N];
15
16     void init(int _n, int _s, int _t) {
17         this->n = _n, this->s = _s, this->t = _t;
18         for(int i = 0; i <= n; ++i){
19             G[i].clear();
20         }
21     }
22
23     void add_edge(int u, int v, ll cap) {
24         e.emplace_back(u, v, cap);
25         e.emplace_back(v, u, 0);
26         m = e.size();
27         G[u].push_back(m - 2);
28         G[v].push_back(m - 1);
29     }
30
31     void bfs() {
32         memset(dep, 0, sizeof(dep));
33         queue<int> q;
34         q.push(t);
35         dep[t] = 1;
36         gap[1] = 1;
37         while (!q.empty()) {
38             int u = q.front();
39             q.pop();
40             for (int i : G[u]) {
41                 int v = e[i].to;
42                 if (!dep[v]) {
43                     dep[v] = dep[u] + 1;
44                     gap[dep[v]]++;
45                     q.push(v);
46                 }
47             }
48         }
49     }
50
51     ll dfs(int u, ll flow) {
52         if (u == t || !flow) return flow;
53         ll used = 0;
54         for (int &i = cur[u]; i < G[u].size(); ++i) {
55             int id = G[u][i];
56             auto&[from, to, can_flow] = e[id];
57             if (can_flow && dep[u] == dep[to] + 1) {
58                 ll tmp = dfs(to, min(flow - used, can_flow));
59                 if (tmp) {
60                     e[id].can_flow -= tmp;
61                     e[id ^ 1].can_flow += tmp;
62                     used += tmp;
63                 }
64                 if (used == flow) return used;

```

```

65         }
66     }
67
68     --gap[dep[u]];
69     if (!gap[dep[u]]) dep[s] = n + 1;
70     ++dep[u];
71     ++gap[dep[u]];
72     return used;
73 }
74
75 ll isap() {
76     ll max_flow = 0;
77     bfs();
78     while (dep[s] <= n) {
79         memset(cur, 0, sizeof(cur));
80         max_flow += dfs(s, INF);
81     }
82     return max_flow;
83 }
84 };
85 #endif

```

3.4 Johnson.cpp

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  using ll = long long;
4  const ll N = 3e3 + 10;
5  ll h[N];
6  struct edge{
7      ll u;
8      ll w;
9      ll t;
10 };
11 vector<edge> v;
12 struct node{
13     ll u;
14     ll t;
15     bool operator<(const node& other) const{
16         return t < other.t;
17     }
18     bool operator>(const node& other) const{
19         return t > other.t;
20     }
21 };
22 priority_queue<node, vector<node>, greater<node>> q;
23 vector<node> s[N];
24 bool bellmanford(ll n){
25     bool flag = false;
26     for(ll i = 1; i <= n; ++i){
27         flag = false;
28         for(auto j: v){

```

```

29     if(h[j.w] > h[j.u] + j.t){
30         h[j.w] = h[j.u] + j.t;
31         flag = true;
32     }
33 }
34 if(!flag){
35     break;
36 }
37 }
38 return flag;
39 }
40 ll w[N][N];
41 void Dijskal(ll st, ll n){
42     vector<ll> isval(n + 10);
43     w[st][st] = 0;
44     priority_queue<node, vector<node>, greater<node>> q;
45     q.emplace(st, 0);
46     while(!q.empty()){
47         auto u = q.top();
48         q.pop();
49         if(isval[u.u])
50             continue;
51         isval[u.u] = 1;
52         for(auto i: s[u.u]){
53             if(w[st][i.u] > w[st][u.u] + i.t){
54                 w[st][i.u] = w[st][u.u] + i.t;
55                 q.emplace(i.u, w[st][i.u]);
56             }
57         }
58     }
59     for(int i = 1; i <= n; ++i){
60         w[st][i] = w[st][i] - h[st] + h[i];
61     }
62 }
63 int main(){
64     ll n, m;
65     cin >> n >> m;
66     for(ll i = 0; i < m; ++i){
67         ll u, w, t;
68         cin >> u >> w >> t;
69         v.emplace_back(u, w, t);
70     }
71     for(ll i = 1; i <= n; ++i){
72         v.emplace_back(0, i, 0);
73     }
74     if(bellmanford(n)){
75         cout << -1;
76         return 0;
77     }
78     for(auto& i: v){
79         auto& [u, w, t] = i;
80         t += h[u] - h[w];
81     }
82     for(ll i = 0; i < m; ++i){

```

```

83     auto&[u, w, t] = v[i];
84     s[u].emplace_back(w, t);
85 }
86 memset(w, 0x3f, sizeof(w));
87 for(ll i = 1; i <= n; ++i){
88     Dijskal(i, n);
89 }
90 for(ll i = 1; i <= n; ++i){
91     ll sum = 0;
92     for(ll j = 1; j <= n; ++j){
93         if(w[i][j] > ll(1e9)){
94             sum += j * ll(1e9);
95         }
96         else
97             sum += w[i][j] * j;
98     }
99     cout << sum << '\n';
100 }
101 }

```

3.5 LCA.cpp

```

1  #ifndef IGNORE_THIS_FILE
2  vector<vector<int>> a(n + 1, vector<int>(20)), v(n + 1);
3  vector<int> dep(n + 1);
4  auto build = [&](int u, int fa, auto&& self) -> void {
5      dep[u] = dep[fa] + 1, a[u][0] = fa;
6      for(int i = 1; i <= 19; ++i)
7          a[u][i] = a[a[u][i - 1]][i - 1];
8      for(int i: v[u]){
9          if(i == fa) continue;
10         self(i, u, self);
11     }
12 };
13 auto lca = [&](int x, int y) -> int {
14     if(dep[y] > dep[x]) swap(x, y);
15     for(int i = 19; i >= 0; --i){
16         if(dep[a[x][i]] >= dep[y])
17             x = a[x][i];
18     }
19     if(x == y) return x;
20     for(int i = 19; i >= 0; --i){
21         if(a[x][i] != a[y][i])
22             x = a[x][i], y = a[y][i];
23     }
24     return a[x][0];
25 };
26 #endif
27
28
29

```

3.6 lcd.cpp

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  typedef long long ll;
4  const ll N = 1e6 + 10;
5  vector<ll> v[N];
6  ll fa[N], dep[N], son[N], sz[N], top[N];
7  ll dfn[N], rid[N]; //dfn 序列, dfn-> 标号
8  ll bot[N]; //维护子树 dfn 序结束编号
9  ll vl[N];
10 ll n, m, r, mod;
11 ll cnt = 0;
12
13 void dfs1(ll u, ll far){
14     fa[u] = far, dep[u] = dep[far] + 1, sz[u] = 1;
15     for(auto i: v[u]){
16         if(i == far)
17             continue;
18         dfs1(i, u);
19         sz[u] += sz[i];
20         if(sz[son[u]] < sz[i])
21             son[u] = i;
22     }
23 }
24 void dfs2(ll u, ll head){
25     top[u] = head;
26     dfn[u] = ++cnt;
27     rid[cnt] = u;
28     if(!son[u]){
29         bot[u] = cnt;
30         return;
31     }
32     dfs2(son[u], head);
33     for(auto i: v[u]){
34         if(i == fa[u] || son[u] == i)
35             continue;
36         dfs2(i, i);
37     }
38     bot[u] = cnt;
39 }
40 ll sm[4 * N];
41 ll lz[4 * N];
42 void pushup(ll node){
43     sm[node] = sm[node << 1] + sm[node << 1 | 1];
44     sm[node] %= mod;
45 }
46
47 void pushdown(ll node, ll l, ll r, ll mid){
48     if(lz[node]){
49         lz[node << 1] += lz[node];
50         lz[node << 1] %= mod;
51         lz[node << 1 | 1] += lz[node];

```

```

52     lz[node << 1 | 1] %= mod;
53     sm[node << 1] += lz[node] * (mid - 1 + 1ll);
54     sm[node << 1] %= mod;
55     sm[node << 1 | 1] += lz[node] * (r - mid);
56     sm[node << 1 | 1] %= mod;
57     lz[node] = 0;
58 }
59 }
60 void build(ll node, ll l, ll r){
61     if(l == r){
62         sm[node] = vl[rid[l]];
63         sm[node] %= mod;
64         return;
65     }
66     ll mid = (l + r) >> 1;
67     build(node << 1, l, mid);
68     build(node << 1 | 1, mid + 1, r);
69     pushup(node);
70 }
71
72 void update(ll node, ll l, ll r, ll ql, ll qr, ll val){
73     if(ql <= l && r <= qr){
74         sm[node] += val * (r - l + 1ll);
75         sm[node] %= mod;
76         lz[node] += val;
77         lz[node] %= mod;
78         return;
79     }
80     ll mid = (l + r) >> 1;
81     pushdown(node, l, r, mid);
82     if(ql <= mid)
83         update(node << 1, l, mid, ql, qr, val);
84     if(qr > mid)
85         update(node << 1 | 1, mid + 1, r, ql, qr, val);
86     pushup(node);
87 }
88
89 ll query(ll node, ll l, ll r, ll ql, ll qr){
90     if(ql <= l && r <= qr){
91         return sm[node] % mod;
92     }
93     ll mid = (l + r) >> 1;
94     pushdown(node, l, r, mid);
95     ll sum = 0;
96     if(ql <= mid)
97         sum += query(node << 1, l, mid, ql, qr);
98     if(qr > mid)
99         sum += query(node << 1 | 1, mid + 1, r, ql, qr);
100     return sum % mod;
101 }
102
103 void add_path(ll u, ll v, ll val){
104     while(top[u] != top[v]){
105         if(dep[top[u]] < dep[top[v]])

```



```

106     swap(u, v);
107     update(1, 1, n, dfn[top[u]], dfn[u], val);
108     u = fa[top[u]];
109 }
110 if(dep[u] < dep[v]) swap(u, v);
111 update(1, 1, n, dfn[v], dfn[u], val);
112 }
113
114 ll get_path(ll u, ll v){
115     ll sum = 0;
116     while(top[u] != top[v]){
117         if(dep[top[u]] < dep[top[v]])
118             swap(u, v);
119         // sum += query(1, 1, n, dfn[top[u]], dfn[u]);
120         // sum %= mod;
121         u = fa[top[u]];
122     }
123     if(dep[u] < dep[v]) swap(u, v);
124     sum += query(1, 1, n, dfn[v], dfn[u]);
125     return sum % mod;
126 }
127
128 void update_root(ll x, ll val){
129     update(1, 1, n, dfn[x], bot[x], val);
130 }
131 ll get_root(ll x){
132     return query(1, 1, n, dfn[x], bot[x]);
133 }
134 int main(){
135     cin >> n >> m >> r >> mod;
136     for(ll i = 1; i <= n; ++i){
137         cin >> vl[i];
138     }
139     for(ll i = 1; i < n; ++i){
140         ll x, y;
141         cin >> x >> y;
142         v[x].push_back(y);
143         v[y].push_back(x);
144     }
145     dfs1(r, 0);
146     dfs2(r, r);
147     build(1, 1, n);
148     for(ll i = 1; i <= m; ++i){
149         ll op, x, y, z;
150         cin >> op;
151         if(op == 1){
152             cin >> x >> y >> z;
153             add_path(x, y, z);
154         }
155         else if(op == 2){
156             cin >> x >> y;
157             cout << get_path(x, y) << '\n';
158         }
159         else if(op == 3){

```

```

160     cin >> x >> z;
161     update_root(x, z);
162 }
163 else{
164     cin >> x;
165     cout << get_root(x) << '\n';
166 }
167 }
168 }

```

3.7 lcd2.cpp

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  typedef long long ll;
4  const ll N = 2e5 + 10;
5  ll fa[N], ht[N], top[N];
6  pair<ll, ll> son[N];
7  ll len[N];
8  vector<pair<ll, ll> > v[N];
9
10
11 void dfs1(ll u, ll far){
12     fa[u] = far;
13     for(auto [i, w]: v[u]){
14         if(i == far)
15             continue;
16         dfs1(i, u);
17         ht[u] = max(ht[u], ht[i] + w);
18         if(ht[son[u].first] + son[u].second < ht[i] + w)
19             son[u] = {i, w};
20     }
21 }
22
23 ll ans[N];
24 void dfs2(ll u, ll head){
25     top[u] = head;
26     if(!son[u].first){
27         return;
28     }
29     len[son[u].first] = len[u] + son[u].second;
30     dfs2(son[u].first, head);
31     for(auto [i, w]: v[u]){
32         if(i == fa[u] || son[u].first == i)
33             continue;
34         len[i] = w;
35         dfs2(i, i);
36     }
37 }
38 int main(){
39     ll n;
40     cin >> n;

```

```

41     for(ll i = 1; i < n; ++i){
42         ll x, y, z;
43         cin >> x >> y >> z;
44         v[x].emplace_back(y, z);
45         v[y].emplace_back(x, z);
46     }
47     dfs1(1, 0);
48     dfs2(1, 1);
49     for(ll i = 1; i <= n; ++i){
50         ans[top[i]] = max(ans[top[i]], len[i]);
51     }
52     ll sum = 0;
53     sort(ans + 1, ans + 1 + n, [](const auto x, const auto y){return x > y;});
54     for(ll i = 1; i <= n; ++i){
55         sum += ans[i] * 2ll;
56         cout << sum << ' ';
57     }
58
59 }

```

3.8 tarjan.cpp

```

1  #ifdef IGNORE_THIS_FILE
2      int dfn_cnt = 0, scc_cnt = 0;
3      vector<int> dfn(n + 1), low(n + 1), scc_id(n + 1);
4      stack<int> s;
5      vector<bool> in_stack(n + 1);
6      vector<vector<int>> v(n + 1), scc(n + 1);
7      // scc_id 是每个节点所属于的 scc 编号, scc 是这个编号下的所有节点
8      auto tarjan = [&](int u, auto&& self) -> void {
9          dfn[u] = low[u] = ++dfn_cnt;
10         s.push(u), in_stack[u] = true;
11         for(int i: v[u]){
12             if(!dfn[i]){
13                 self(i, self);
14                 low[u] = min(low[u], low[i]);
15             }
16             else if(in_stack[i])
17                 low[u] = min(low[u], dfn[i]);
18         }
19         if(dfn[u] == low[u]){
20             int tp;
21             ++scc_cnt;
22             do{
23                 tp = s.top();
24                 scc_id[tp] = scc_cnt;
25                 scc[scc_cnt].push_back(tp);
26                 in_stack[tp] = false;
27                 s.pop();
28             } while(tp != u);
29         }
30     };

```

```
31 #endif
```

3.9 tree divide and conquer.cpp

```
1  #include <bits/stdc++.h>
2  using namespace std;
3  typedef long long ll;
4  const int N = 1e4 + 10;
5  vector<pair<int, int>> v[N];
6  int n, m;
7  bool vis[N];
8  int siz[N], dep[N];
9  int st[1000 * N];
10 int wt[N];
11 int ans[101];
12 int get(int u, int far, int& num) { // 求子树的重心
13     siz[u] = 1;
14     wt[u] = 0;
15     ++num;
16     int root = -1;
17     for(auto& [i, w] : v[u]){
18         if(i == far || vis[i]) continue;
19         root = max(root, get(i, u, num));
20         siz[u] += siz[i];
21         wt[u] = max(wt[u], siz[i]);
22     }
23     wt[u] = max(wt[u], num - siz[u]);
24     if(wt[u] <= num / 2){
25         return u;
26     }
27     return root;
28 }
29
30 void cal(int u, int far, int w, vector<int> &lst) { // 向下暴力递归计算
31     dep[u] = dep[far] + w;
32     lst.push_back(u);
33     for (auto &[i, w] : v[u]) {
34         if (vis[i] || i == far) continue;
35         cal(i, u, w, lst);
36     }
37 }
38
39 void dfs2(int u, vector<int>& cur) {
40     int num = 0;
41     int root = get(u, 0, num);
42     vis[root] = true;
43     st[0] = 1;
44     dep[root] = 0;
45     vector<vector<int>> > lst;
46     for (auto &[i, w] : v[root]) {
47         if (vis[i]) continue;
48         vector<int> tmp;
```

```

49     cal(i, root, w, tmp);
50     lst.push_back(tmp);
51     for (auto &node : tmp) {
52         for(int k = 0; k < cur.size(); ++k){
53             if (cur[k] >= dep[node]) {
54                 ans[k] += st[cur[k] - dep[node]];
55             }
56         }
57     }
58
59     for (auto &node : tmp) {
60         if(dep[node] <= 1e7){
61             ++st[dep[node]];
62         }
63     }
64 }
65 for(auto& i: lst){
66     for(auto& j: i){
67         if(dep[j] <= 1e7)
68             st[dep[j]] = 0;
69     }
70 }
71 st[0] = 0;
72 for (auto &[i, w] : v[root]) {
73     if (vis[i]) continue;
74     dfs2(i, cur);
75 }
76 }
77
78 int main() {
79     ios::sync_with_stdio(false);
80     cin.tie(nullptr);
81     cin >> n >> m;
82     for (int i = 1; i < n; ++i) {
83         int u, v_, w;
84         cin >> u >> v_ >> w;
85         v[u].emplace_back(v_, w);
86         v[v_].emplace_back(u, w);
87     }
88     vector<int> cur;
89     for(int i = 0; i < m; ++i){
90         int x;
91         cin >> x;
92         cur.push_back(x);
93     }
94     dfs2(1, cur);
95     for(int i = 0; i < m; ++i){
96         cout << (ans[i] ? "AYE\n" : "NAY\n");
97     }
98     return 0;
99 }

```

4 heading

4.1 debug.h

```

1  #include <bits/stdc++.h>
2  #define typet typename T
3  #define typeu typename U
4  #define types typename... Ts
5  #define tempt template <typet>
6  #define tempu template <typeu>
7  #define temps template <types>
8  #define tandu template <typet, typeu>
9
10 tandu std::ostream& operator<<(std::ostream& os, const std::pair<T, U>& p) {
11     return os << '<' << p.ff << ',' << p.ss << '>';
12 }
13
14 template <
15     typet, typename = decltype(std::begin(std::declval<T>())),
16     typename = std::enable_if_t<!std::is_same_v<T, std::string>>>
17     std::ostream& operator<<(std::ostream& os, const T& c) {
18     auto it = std::begin(c);
19     if (it == std::end(c)) return os << "{}";
20     for (os << '{' << *it; ++it != std::end(c); os << ',' << *it);
21     return os << '}';
22 }
23
24 #define debug(arg...) \
25 do { \
26     std::cerr << "[" #arg "]" :"; \
27     dbg(arg); \
28 }while(false)
29
30 temps void dbg(Ts... args) {
31     (... , (std::cerr << ' ' << args));
32     std::cerr << '\n';
33 }

```

4.2 duipai.cpp

```

1  #ifdef IGNORE_THIS_FILE
2      system("g++ -std=c++2a wa.cpp -o/wa");
3      system("g++ -std=c++2a ac.cpp -o/ac");
4      system("g++ -std=c++2a gen.cpp -o/gen");
5      for(int i = 1; i <= 50; i++){
6          std::cerr << "Test" << i << " : ";
7          system("./gen > gen.in");
8          system("./ac < gen.in > ac.out");
9          system("./wa < gen.in > wa.out");
10         if (system("diff ac.out wa.out")) {
11             std::cerr << "ERR\n";
12             return 0;

```

```
13     }
14     std::cerr << "AC\n";
15 }
16 #endif
```

4.3 heading.cpp

```
1  #include <bits/stdc++.h>
2  using namespace std;
3  using ll = long long;
4  using i128 = __int128;
5  #define ff first
6  #define ss second
7  #include "debug.h"
8  constexpr int mod = 998244353;
9  constexpr ll INF = 1e18;
10 constexpr double pi = 3.141592653589793;
11 constexpr double eps = 1e-6;
12
13
```

5 math

5.1 Eratost.cpp

```

1  #ifndef IGNORE_THIS_FILE
2      vector<int> sieve(int n){
3          vector<bool> is_prime(n + 1);
4          vector<int> prime;
5          for(int i = 2; i <= n; ++i){
6              is_prime[i] = true;
7          }
8          for(int i = 2; i * i <= n; ++i){
9              if(is_prime[i]){
10                 for(int j = i * i; j <= n; j += i)
11                     is_prime[j] = false;
12             }
13         }
14         for(int i = 2; i <= n; ++i){
15             if(is_prime[i])
16                 prime.push_back(i);
17         }
18         return prime;
19     }
20 #endif

```

5.2 Euler.cpp

```

1  #ifndef IGNORE_THIS_FILE
2      // vector<int> fac(n + 1);
3      vector<int> sieve(int n){
4          vector<int> prime;
5          vector<bool> no_prime(n + 1);
6          for(int i = 2; i <= n; ++i){
7              if(!no_prime[i]){
8                  prime.push_back(i);
9                  // fac[i] = i;
10             }
11             for(int j: prime){
12                 if(j * i > n) break;
13                 no_prime[j * i] = true;
14                 // fac[j * i] = j;
15                 if(i % j == 0) break;
16             }
17         }
18         return prime;
19     }
20 #endif

```

5.3 FFT.cpp

```

1  #ifndef IGNORE_THIS_FILE
2  /*
3   * 做 FFT
4   * len 必须是 2k 形式
5   * on == 1 时是 DFT, on == -1 时是 IDFT
6   */
7  void fft(Complex y[], int len, int on) {
8      // 位逆序置换
9      change(y, len);
10     // 模拟合并过程, 一开始, 从长度为一合并到长度为二, 一直合并到长度为 len。
11     for (int h = 2; h <= len; h <= 1) {
12         // wn: 当前单位复根的间隔: w1/h
13         Complex wn(cos(2 * PI / h), sin(on * 2 * PI / h));
14         // 合并, 共 len / h 次。
15         for (int j = 0; j < len; j += h) {
16             // 计算当前单位复根, 一开始是 1 = w0/n, 之后是以 wn 为间隔递增: w1/n
17             // ...
18             Complex w(1, 0);
19             for (int k = j; k < j + h / 2; k++) {
20                 // 左侧部分和右侧是子问题的解
21                 Complex u = y[k];
22                 Complex t = w * y[k + h / 2];
23                 // 这就是把两部分分治的结果加起来
24                 y[k] = u + t;
25                 y[k + h / 2] = u - t;
26                 // 后半部 「step」 中的 一定和 「前半部」 中的成相反数
27                 // 「红圈」 上的点转一整圈「转回来」, 转半圈正好转成相反数
28                 // 一个数相反数的平方与这个数自身的平方相等
29                 w = w * wn;
30             }
31         }
32     }
33     // 如果是 IDFT, 它的逆矩阵的每一个元素不只是原元素取倒数, 还要除以长度 len。
34     if (on == -1) {
35         for (int i = 0; i < len; i++) {
36             y[i].x /= len;
37             y[i].y /= len;
38         }
39     }
40 }
41
42 #endif

```

5.4 Gause_{Xor}.cpp

```

1  #ifndef IGNORE_THIS_FILE
2
3  std::bitset<1010> matrix[2010]; // matrix[1~n]: 增广矩阵, 0 位置为常数
4  std::vector<bool> GaussElimination(int n, int m) {

```

```

5 // n 为未知数个数, m 为方程个数, 返回方程组的 (多解 / 无解返回一个空的 vector)
6 for (int i = 1; i <= n; i++) {
7     int cur = i;
8     while (cur <= m && !matrix[cur].test(i)) cur++;
9     if (cur > m) return std::vector<bool>(0);
10    if (cur != i) swap(matrix[cur], matrix[i]);
11    for (int j = 1; j <= m; j++)
12        if (i != j && matrix[j].test(i)) matrix[j] ^= matrix[i];
13 }
14 std::vector<bool> ans(n + 1);
15 for (int i = 1; i <= n; i++) ans[i] = matrix[i].test(0);
16 return ans;
17 }
18 #endif

```

5.5 NTT.cpp

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 int read() {
5     int x = 0, f = 1;
6     char ch = getchar();
7     while (ch < '0' || ch > '9') {
8         if (ch == '-') f = -1;
9         ch = getchar();
10    }
11    while (ch <= '9' && ch >= '0') {
12        x = 10 * x + ch - '0';
13        ch = getchar();
14    }
15    return x * f;
16 }
17
18 void print(int x) {
19     if (x < 0) putchar('-'), x = -x;
20     if (x >= 10) print(x / 10);
21     putchar(x % 10 + '0');
22 }
23
24 constexpr int N = 300100, P = 998244353;
25
26 int qpow(int x, int y) {
27     int res(1);
28     while (y) {
29         if (y & 1) res = 1ll * res * x % P;
30         x = 1ll * x * x % P;
31         y >>= 1;
32     }
33     return res;
34 }
35

```

```

36  int r[N];
37
38  void ntt(int *x, int lim, int opt) {
39      int i, j, k, m, gn, g, tmp;
40      for (i = 0; i < lim; ++i)
41          if (r[i] < i) swap(x[i], x[r[i]]);
42      for (m = 2; m <= lim; m <= 1) {
43          k = m >> 1;
44          gn = qpow(3, (P - 1) / m);
45          for (i = 0; i < lim; i += m) {
46              g = 1;
47              for (j = 0; j < k; ++j, g = 1ll * g * gn % P) {
48                  tmp = 1ll * x[i + j + k] * g % P;
49                  x[i + j + k] = (x[i + j] - tmp + P) % P;
50                  x[i + j] = (x[i + j] + tmp) % P;
51              }
52          }
53      }
54      if (opt == -1) {
55          reverse(x + 1, x + lim);
56          int inv = qpow(lim, P - 2);
57          for (i = 0; i < lim; ++i) x[i] = 1ll * x[i] * inv % P;
58      }
59  }
60
61  int A[N], B[N], C[N];
62
63  char a[N], b[N];
64
65  int main() {
66      int i, lim(1), n;
67      scanf("%s", a);
68      n = strlen(a);
69      for (i = 0; i < n; ++i) A[i] = a[n - i - 1] - '0';
70      while (lim < (n << 1)) lim <= 1;
71      scanf("%s", b);
72      n = strlen(b);
73      for (i = 0; i < n; ++i) B[i] = b[n - i - 1] - '0';
74      while (lim < (n << 1)) lim <= 1;
75      for (i = 0; i < lim; ++i) r[i] = (i & 1) * (lim >> 1) + (r[i >> 1] >> 1);
76      ntt(A, lim, 1);
77      ntt(B, lim, 1);
78      for (i = 0; i < lim; ++i) C[i] = 1ll * A[i] * B[i] % P;
79      ntt(C, lim, -1);
80      int len(0);
81      for (i = 0; i < lim; ++i) {
82          if (C[i] >= 10) len = i + 1, C[i + 1] += C[i] / 10, C[i] %= 10;
83          if (C[i]) len = max(len, i);
84      }
85      while (C[len] >= 10) C[len + 1] += C[len] / 10, C[len] %= 10, len++;
86      for (i = len; ~i; --i) putchar(C[i] + '0');
87      puts("");
88      return 0;

```

89 }

5.6 pollar-rho.cpp

```

1  #ifndef IGNORE_THIS_FILE
2      using ll = long long;
3      using ull = unsigned long long;
4      bool is_prime(ull n) {
5          if(n == 2) { return true; }
6          if(n % 2 == 0) { return false; }
7          auto internal_pow = [&](ull x, ull y) {
8              ull r = 1;
9              __uint128_t c = x;
10             for(; y; y >>= 1, c = c * c % n) {
11                 if(y & 1) { r = __uint128_t(r) * c % n; }
12             }
13             return r;
14         };
15         auto MillerRabin = [&](ull a) {
16             if(n <= a) { return true; }
17             int e = __builtin_ctzll(n - 1);
18             ull z = internal_pow(a, (n - 1) >> e);
19             if(z == 1 || z == n - 1) { return true; }
20             while(--e) {
21                 z = __uint128_t(z) * z % n;
22                 if(z == 1) { return false; }
23                 if(z == n - 1) { return true; }
24             }
25             return false;
26         };
27         vector<ull> cur;
28         if(n < 4759123141) cur = vector<ull>{2, 7, 61};
29         else cur = vector<ull>{2, 325, 9375, 28178, 450775, 9780504, 1795265022};
30         return all_of(cur.begin(), cur.end(), [&](auto x) { return MillerRabin(x); });
31     }
32
33     struct Montgomery {
34         ull mod, R;
35     public:
36         Montgomery(ull n): mod(n), R(n) {
37             for(int i = 0; i < 5; i++) { R *= 2 - mod * R; }
38         }
39         ull fma(ull a, ull b, ull c) const {
40             const __uint128_t d = __uint128_t(a) * b;
41             const ull e = c + mod + (d >> 64);
42             const ull f = ull(d) * R;
43             const ull g = (__uint128_t(f) * mod) >> 64;
44             return e - g;
45         }
46         ull mul(ull a, ull b) const { return fma(a, b, 0); }
47     };
48     ull PollardRho(ull n) {

```

```

49     if(n % 2 == 0) { return 2; }
50     const Montgomery m(n);
51     constexpr ull C1 = 1, C2 = 2, M = 512;
52     ull Z1 = 1, Z2 = 2;
53     retry:
54     ull z1 = Z1, z2 = Z2;
55     for(unsigned k = M;; k <= 1) {
56         const ull x1 = z1 + n, x2 = z2 + n;
57         for(unsigned j = 0; j < k; j += M) {
58             const ull y1 = z1, y2 = z2;
59             ull q1 = 1, q2 = 2;
60             z1 = m.fma(z1, z1, C1), z2 = m.fma(z2, z2, C2);
61             for(unsigned i = 0; i < M; i++) {
62                 const ull t1 = x1 - z1, t2 = x2 - z2;
63                 z1 = m.fma(z1, z1, C1), z2 = m.fma(z2, z2, C2);
64                 q1 = m.mul(q1, t1), q2 = m.mul(q2, t2);
65             }
66             q1 = m.mul(q1, x1 - z1), q2 = m.mul(q2, x2 - z2);
67             const ull q3 = m.mul(q1, q2), g3 = gcd(n, q3);
68             if(g3 == 1) { continue; }
69             if(g3 != n) { return g3; }
70             const ull g1 = gcd(n, q1), g2 = gcd(n, q2);
71             const ull C = g1 != 1 ? C1 : C2, x = g1 != 1 ? x1 : x2;
72             ull z = g1 != 1 ? y1 : y2, g = g1 != 1 ? g1 : g2;
73             if(g == n) {
74                 do {
75                     z = m.fma(z, z, C);
76                     g = gcd(n, x - z);
77                 } while(g == 1);
78             }
79             if(g != n) { return g; }
80             Z1 += 2, Z2 += 2;
81             goto retry;
82         }
83     }
84 }
85 vector<ull> PrimeFactorize(ull n) {
86     vector<ull> r;
87     auto rec = [&](auto &&rec, ull n, vector<ull> &r) -> void {
88         if(n <= 1) { return; }
89         if(is_prime(n)) {
90             r.emplace_back(n);
91             return;
92         }
93         const ull p = PollardRho(n);
94         rec(rec, p, r);
95         rec(rec, n / p, r);
96     };
97     rec(rec, n, r);
98     sort(r.begin(), r.end());
99     return r;
100 }
101 vector<pair<ll, ll>> Prime(ll n) {
102     auto ans = PrimeFactorize(n);

```

```

103     vector<pair<ll, ll>> cur;
104     for(ll i = 0; i < ans.size(); ++i){
105         ll e = 1;
106         while(i + 1 < ans.size() && ans[i + 1] == ans[i]) ++e, ++i;
107         cur.push_back({ans[i], e});
108     }
109     return cur;
110 }
111 // auto get_tot = [](auto &&get_tot, vector<pair<ll, ll>>& prime, vector<ll>& tot, int pos, ll val){
112 //     if(pos == prime.size()){
113 //         tot.push_back(val);
114 //         return;
115 //     }
116 //     for(ll j = 0, sum = 1; j <= prime[pos].second; ++j, sum *= prime[pos].first){
117 //         get_tot(get_tot, prime, tot, pos + 1, val * sum);
118 //     }
119 // };
120 #endif

```

5.7 Xor_{base}.cpp

```

1  #ifdef IGNORE_THIS_FILE
2  auto insert = [](ll x, vector<ll>& a) -> void {
3      for(int i = 63; i >= 0; --i){
4          if(!((x >> i) & 1)) continue;
5          if(a[i] x ^= a[i];
6          else{
7              for(int j = 0; j < i; ++j){
8                  if((x >> j) & 1)
9                      x ^= a[j];
10             }
11             for(int j = i + 1; j <= 63; ++j){
12                 if((a[j] >> i) & 1)
13                     a[j] ^= x;
14             }
15             a[i] = x;
16         }
17     }
18 };
19 #endif

```

6 sort

6.1 merge-sort.cpp

```

1  #ifdef IGNORE_THIS_TILE
2  void merge_sort(int l, int r, vector<int>& a) {
3      if(l == r)
4          return;
5      int mid = (l + r) >> 1;
6      merge_sort(l, mid, a);
7      merge_sort(mid + 1, r, a);
8      auto merge = [](int l, int r, int mid, vector<int>& a) -> void {
9          vector<int> b;
10         int lp = l;
11         int rp = r + 1;
12         while(lp <= mid && rp <= r){
13             if(a[lp] < a[rp])
14                 b.push_back(a[lp++]);
15             else
16                 b.push_back(a[rp++]);
17         }
18         while(lp <= mid)
19             b.push_back(a[lp++]);
20         while(rp <= r)
21             b.push_back(a[rp++]);
22         copy(b.begin(), b.end(), a.begin() + l);
23     };
24     merge(l, r, mid, a);
25 };
26 #endif

```

7 string

7.1 AC_{automation}.cpp

```

1  #ifndef IGNORE_THIS_FILE
2  struct ACAutomaton {
3      static const int N = 2e6 + 6;
4      static const int maxn = 2e5 + 10;
5
6      int tran[N][26] = {};
7      vector<int> uid[maxn];
8      int fail[N] = {};
9      int ru_degree[maxn] = {};
10     int dp[N] = {};
11     int ans[maxn] = {};
12     int tot = 0;
13
14     void clear() {
15         for(int i = 0; i < N; i++) {
16             memset(tran[i], 0, sizeof(tran[i]));
17             fail[i] = dp[i] = 0;
18         }
19         for(int i = 0; i < maxn; i++) uid[i].clear();
20         memset(ru_degree, 0, sizeof(ru_degree));
21         memset(ans, 0, sizeof(ans));
22         tot = 0;
23     }
24
25     void insert(const string& s, int id) {
26         int u = 0;
27         for(char ch : s) {
28             int c = ch - 'a';
29             if(!tran[u][c]) tran[u][c] = ++tot;
30             u = tran[u][c];
31         }
32         uid[u].push_back(id);
33     }
34
35     void build() {
36         queue<int> q;
37         for(int i = 0; i < 26; i++)
38             if(tran[0][i]) q.push(tran[0][i]);
39
40         while(!q.empty()) {
41             int u = q.front();
42             q.pop();
43             for(int i = 0; i < 26; i++) {
44                 if(tran[u][i]) {
45                     fail[tran[u][i]] = tran[fail[u]][i];
46                     q.push(tran[u][i]);
47                 } else {
48                     tran[u][i] = tran[fail[u]][i];

```



```

49         }
50     }
51 }
52 }
53
54 void query(const string& t) {
55     int u = 0;
56     for(char ch : t) {
57         u = tran[u][ch - 'a'];
58         dp[u]++;
59     }
60
61     for(int i = 1; i <= tot; i++)
62         ru_degree[fail[i]]++;
63
64     queue<int> q;
65     for(int i = 1; i <= tot; i++)
66         if(!ru_degree[i]) q.push(i);
67
68     while(!q.empty()) {
69         int u = q.front();
70         q.pop();
71         dp[fail[u]] += dp[u];
72         if(--ru_degree[fail[u]] == 0)
73             q.push(fail[u]);
74     }
75
76     for(int i = 1; i <= tot; i++)
77         for(int id : uid[i])
78             ans[id] = dp[i];
79 }
80 };
81
82 #endif

```

7.2 KMP.cpp

```

1  #ifdef IGNORE_THIS_FILE
2
3
4  vector<int> prefix_function(string s) {
5      int n = (int)s.length();
6      vector<int> pi(n);
7      for (int i = 1; i < n; i++) {
8          int j = pi[i - 1];
9          while (j > 0 && s[i] != s[j]) j = pi[j - 1];
10         if (s[i] == s[j]) j++;
11         pi[i] = j;
12     }
13     return pi;
14 }
15

```

```

16
17 void compute_automaton(string s, vector<vector<int>>& aut) {
18     s += '#';
19     int n = s.size();
20     vector<int> pi = prefix_function(s);
21     aut.assign(n, vector<int>(26));
22     for (int i = 0; i < n; i++) {
23         for (int c = 0; c < 26; c++) {
24             if (i > 0 && 'a' + c != s[i])
25                 aut[i][c] = aut[pi[i - 1]][c];
26             else
27                 aut[i][c] = i + ('a' + c == s[i]);
28         }
29     }
30 }
31
32
33 #endif

```

7.3 Manacher.cpp

```

1  #ifndef IGNORE_THIS_FILE
2
3  struct Manacher {
4      string transformed;
5      vector<int> d1;
6
7      Manacher(const string& s) {
8          // 预处理字符串, 插入 '#'
9          transformed = "#";
10         for (char c : s) {
11             transformed += c;
12             transformed += '#';
13         }
14         int n = transformed.size();
15         d1.resize(n);
16         // 计算每个中心点的最长回文半径
17         int l = 0, r = -1;
18         for (int i = 0; i < n; ++i) {
19             int k = (i > r) ? 1 : min(d1[l + r - i], r - i + 1);
20             while (i - k >= 0 && i + k < n && transformed[i - k] == transformed[i + k]) {
21                 k++;
22             }
23             d1[i] = k--;
24             if (i + k > r) {
25                 l = i - k;
26                 r = i + k;
27             }
28         }
29     }
30 };
31

```

```
32 #endif
```

7.4 PAM.cpp

```

1  #include <vector>
2  #include <string>
3  #include <array>
4  using namespace std;
5
6  struct Eertree {
7      vector<array<int, 26>> tr;    // 转移数组
8      vector<int> fail;           // 失配指针
9      vector<int> len;            // 节点表示的回文串长度
10     vector<int> ans;            // 存储每个位置的回文深度
11     vector<int> depth;          // 节点深度
12     int tot, last;             // 总节点数和当前最后节点
13
14     Eertree() {
15         tr.resize(2);
16         tr[0].fill(0), tr[1].fill(0);
17         fail.resize(2);
18         len.resize(2);
19         depth.resize(2);
20
21         fail[0] = 1; // 偶根失配指向奇根
22         fail[1] = 0; // 奇根失配指向偶根
23         len[0] = 0;  // 偶根长度 0
24         len[1] = -1; // 奇根长度 -1
25         depth[0] = 0;
26         depth[1] = 0;
27         tot = 1;    // 已创建两个节点
28         last = 1;   // 初始指向奇根
29     }
30
31     void insert(const string& s) {
32         ans.resize(s.size());
33         for (int i = 0; i < s.size(); i++) {
34             int c = s[i] - 'a';
35             int fa = get_fail(last, i, s);
36
37             if (!tr[fa][c]) {
38                 create_node(fa, c, i, s);
39             }
40             last = tr[fa][c];
41             ans[i] = depth[last];
42         }
43     }
44
45     int get_fail(int pos, int idx, const string& s) {
46         while (idx - len[pos] - 1 < 0 ||
47             s[idx] != s[idx - len[pos] - 1]) {
48             pos = fail[pos];

```

```

49     }
50     return pos;
51 }
52
53 void create_node(int fa, int c, int idx, const string& s) {
54     int cur = ++tot;
55     tr.resize(tot + 1);
56     tr.back().fill(0);
57
58     // 扩展关联数组
59     fail.resize(tot + 1);
60     len.resize(tot + 1);
61     depth.resize(tot + 1);
62
63     tr[fa][c] = cur;
64     len[cur] = len[fa] + 2;
65
66     // 设置失配指针
67     int tmp = get_fail(fail[fa], idx, s);
68     fail[cur] = (fa != 1) ? tr[tmp][c] : 0;
69
70     // 计算节点深度
71     depth[cur] = depth[fail[cur]] + 1;
72 }
73 };

```

7.5 *z_function.cpp*

```

1  #ifndef IGNORE_THIS_FILE
2  vector<int> z_function(string s) {
3      int n = (int)s.size();
4      vector<int> z(n);
5      for(int i = 1, l = 0, r = 0; i < n; ++i) {
6          if(i <= r && z[i - l] < r - i + 1) {
7              z[i] = z[i - l];
8          }
9          else {
10             z[i] = max(0, r - i + 1);
11             while(i + z[i] < n && s[z[i]] == s[i + z[i]]) ++z[i];
12         }
13         if(i + z[i] - 1 > r) l = i, r = i + z[i] - 1;
14     }
15 }
16 #endif

```
