

BEIJING NORMAL UNIVERSITY

Template

Pray Never

2024 年 11 月 21 日

目录

1	data structure	3
1.1	persistant-segement-tree.cpp	3
1.2	trie.cpp	3
2	geometry	5
2.1	convex.cpp	5
3	graph	6
3.1	LCA.cpp	6
3.2	tarjan.cpp	6
4	heading	8
4.1	debug.h	8
4.2	duipai.cpp	8
4.3	heading.cpp	9
5	math	10
5.1	Eratost.cpp	10
5.2	Euler.cpp	10
5.3	pollar-rho.cpp	11
6	sort	14
6.1	merge-sort.cpp	14

1 data structure

1.1 persistant-segement-tree.cpp

```

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

```

1.2 trie.cpp

```

1  #ifdef IGNORE_THIS_FILE
2      int cnt = 0;
3      vector<vector<int>> > node(1, vector<int>(26));
4      vector<int> isval(1);
5      auto insert = [&](string& s, int len) -> void {
6          int p = 0;
7          for(int i = 0; i < len; i++){
8              int c = s[i] - 'a';
9              if(!node[p][c]){
10                 node.push_back(vector<int>(26, 0));
11                 isval.push_back(0);
12                 node[p][c] = ++cnt;

```

```
13     }
14     p = node[p][c];
15 }
16 ++isval[p];
17 };
18 auto find = [&](string& s, int len) -> int {
19     int p = 0;
20     for(int i = 0; i < len; i++){
21         int c = s[i] - 'a';
22         if(!node[p][c]) return 0;
23         p = node[p][c];
24     }
25     return isval[p];
26 };
27 #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 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

```

3.2 tarjan.cpp

```

1  #ifndef 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
```

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 pollar-rho.cpp

```

1  #ifdef 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

```

6 sort

6.1 merge-sort.cpp

```
1  #ifndef 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
```
