

Alfred 代码模版库 目录

Ħ	录		3.2 $O(n \log n) - O(\log n)$ LCA	6
			3.3 树状数组	
1	数据结构	3	3.4 并查集	
	1.1 珂朵莉树	3	3.5 轻重权树剖分	8
	1.2 树状数组	3	3.6 强连通分量	9
			3.7 双连通分量	10
2	比赛配置 and 奇技淫巧	4	3.8 二分图匹配	12
	2.1 多组数据代码模板	4	3.9 最小费用最大流	13
	2.2 快读快写	4	3.10 AhoCorasick 自动机	15
	2.3 .clang-format	5	3.11 后缀数组	16
			3.12 LU 分解	17
3	Watashi 代码库 (备用)	5		
	3.1 $O(n \log n) - O(1)$ RMQ	5 4	对一类问题的处理方法	18

1 数据结构

1.1 珂朵莉树

支持区间推平,颜色段统计,在随机数据下期望复杂度为 $O(n \log n)$ 的暴力数据结构。

Listing 1: ChthollyTree.cpp

```
#include <set>
2
    struct ChthollyTree {
3
        typedef long long 11;
4
5
        struct Node {
6
            mutable 11 1, r, v;
7
            inline bool operator<(const Node &x) const { return 1 < x.1; }</pre>
8
9
        std::set<Node> tr;
10
        typedef std::set<Node>::iterator iterator;
11
        ChthollyTree(void) = default;
12
        ChthollyTree(int rng, int val) { init(rng, val); }
        inline void init(ll rng, ll val) noexcept {
13
            tr.insert({1, rng, val}), tr.insert({rng + 1, rng + 1, 0});
14
15
        inline iterator begin(void) const noexcept { return tr.begin(); }
16
17
        inline iterator end(void) const noexcept { return tr.end(); }
18
        inline iterator split(ll pos) {
19
            auto it = tr.lower bound({pos, 0, 0});
20
            if (it != tr.end() && it->l == pos) return it;
21
            11 1 = (--it) -> 1, r = it -> r, v = it -> v;
22
            tr.erase(it), tr.insert(\{1, pos - 1, v\});
23
            return tr.insert({pos, r, v}).first;
24
        inline void assign(ll l, ll r, ll v) {
25
            auto R = split(r + 1), L = split(l);
26
            tr.erase(L, R), tr.insert({l, r, v});
27
28
        template <class Functor> // func(iterator)
29
30
        inline void modify(ll 1, ll r, Functor func) {
31
            auto R = split(r + 1), L = split(l);
32
            for (auto it = L; it != R; it++) func(it);
33
34
        template <class Functor> // func(11 &, iterator)
        inline ll query(ll l, ll r, Functor func) {
35
            11 \text{ ans} = 0;
36
37
            auto R = split(r + 1);
38
            for (auto it = split(l); it != R; it++) func(ans, it);
39
            return ans;
40
41
    };
```

1.2 树状数组

维护满足结合律且可差分信息的,常数较小的数据结构。

Listing 2: Fenwick.cpp

```
#include <vector>
1
3
   template <class T>
4
   struct Fenwick {
5
        std::vector<T> c;
6
        inline int lowbit(int x) { return x & -x; }
7
        inline void merge(T &x, T y) { x = x + y; } // remember to modify
        inline T subtract(T x, T y) { return x - y; }
8
        inline void update(size_t pos, T x) {
9
10
            for (pos++; pos < c.size(); pos += lowbit(pos)) merge(c[pos], x);</pre>
```

```
11
12
        inline void clear(void) {
            for (auto &x : c) x = T();
13
14
        inline T query(size t pos) {
15
            T ans = T();
16
            for (pos++; pos; pos ^= lowbit(pos)) merge(ans, c[pos]);
17
18
            return ans;
19
20
        inline T query(size t l, size t r) {
21
            return subtract(query(r), query(l-1));
22
23
        Fenwick(size t len) : c(len + 2) {}
24
   };
```

2 比赛配置 and 奇技淫巧

2.1 多组数据代码模板

Listing 3: Template.cpp

```
1 #include <bits/stdc++.h>
 2 using namespace std;
 3 using i64 = long long;
   const i64 N = 1e5 + 10;
 4
   int t = 1;
 5
 6
    inline void solve(int Case) {
 7
        // your code here;
 8
 9
    inline void optimizeIO(void) {
10
        ios::sync with stdio(false);
11
        cin.tie(NULL), cout.tie(NULL);
12
   inline void init(void) {}
13
    int main(int argc, char const *argv[]) {
14
        optimizeIO(), init(), cin >> t;
15
        for (int i = 1; i <= t; i++) solve(i);</pre>
16
17
        return 0;
18
   }
```

2.2 快读快写

Listing 4: FastIO.cpp

```
1
    namespace fastIO {
2
        char c, f, e = 0;
3
        namespace usr {
4
            template <class Tp>
            inline int read(_Tp &x) {
5
                x = f = 0, c = getchar();
6
                while (!isdigit(c) && !e) f = c == '-', e \mid = c == EOF, c = getchar();
7
                while (isdigit(c) && !e) x = (x << 1) + (x << 3) + (c ^ 48), c = getchar();
8
                return (e |= c == EOF) ? 0 : ((f ? x = -x : 0), 1);
9
10
11
            template <class Tp>
            inline void write( Tp x) {
13
                if (x < 0) putchar('-'), x = -x;
                if (x > 9) write(x / 10);
14
                putchar((x % 10) ^ 48);
15
16
17
            template <typename \mathbb{T}\text{, typename...}\ \forall >
18
            inline void read(T &t, V &...v) { read(t), read(v...); }
            template <typename T, typename... V>
19
20
            inline void write(T t, V... v) {
```

2.3 .clang-format

Listing 5: .clang-format

```
1 BasedOnStyle: LLVM
2 AlignAfterOpenBracket: BlockIndent
3 # AlignConsecutiveAssignments: Consecutive
4 AlignArrayOfStructures: Right
5 UseTab: Never
6 IndentWidth: 4
7 TabWidth: 4
8 BreakBeforeBraces: Attach
9 AllowShortIfStatementsOnASingleLine: AllIfsAndElse
10 AllowShortLoopsOnASingleLine: true
11 AllowShortBlocksOnASingleLine: true
12 IndentCaseLabels: true
13 ColumnLimit: 0
14 AccessModifierOffset: −4
15
   NamespaceIndentation: All
   FixNamespaceComments: false
17
   AllowShortCaseLabelsOnASingleLine: true
18 AlwaysBreakTemplateDeclarations: MultiLine
19 BinPackParameters: true
   BraceWrapping:
20
     AfterCaseLabel: true
21
     AfterClass: true
22
23 AlignConsecutiveMacros: AcrossEmptyLinesAndComments
24 AlignTrailingComments: Always
```

3 Watashi 代码库 (备用)

3.1 $O(n \log n) - O(1)$ RMQ

Listing 6: rmq.cpp

```
#include <algorithm> // copy
1
2
    #include <climits>
                         // CHAR BIT
3
    using namespace std;
5
6
    template <typename T>
7
    struct RMQ {
        int n;
8
        vector<T> e:
9
        vector<vector<int>> rmq;
10
11
        static const int INT BIT = sizeof(4) * CHAR BIT;
12
13
        static inline int LG2(int i) { return INT_BIT - 1 - _ builtin_clz(i); }
14
        static inline int BIN(int i) { return 1 << i; }</pre>
15
16
        int cmp(int 1, int r) const {
17
            return e[1] <= e[r] ? 1 : r;
18
19
        void init(int n, const T e[]) {
20
21
            this\rightarrown = n;
22
            vector<T>(e, e + n).swap(this->e);
23
```

```
24
             int m = 1;
25
             while (BIN(m) <= n) {
26
                 ++m;
27
             }
             vector<vector<int>> (m, vector<int> (n)).swap(rmq);
28
29
30
             for (int i = 0; i < n; ++i) {</pre>
                 rmq[0][i] = i;
31
32
33
             for (int i = 0; BIN(i + 1) <= n; ++i) {</pre>
34
                 for (int j = 0; j + BIN(i + 1) <= n; ++j) {
35
                     rmq[i + 1][j] = cmp(rmq[i][j], rmq[i][j + BIN(i)]);
36
37
             }
        }
38
39
        int index(int 1, int r) const {
40
41
            int b = LG2(r - 1);
42
            return cmp(rmq[b][1], rmq[b][r - (1 << b)]);
43
44
45
        T value(int 1, int r) const {
46
            return e[index(l, r)];
47
48
    } ;
```

3.2 $O(n \log n) - O(\log n)$ **LCA**

Listing 7: lca.cpp

```
#include <algorithm>
 1
    #include <cstdio>
 2
 3
    #include <vector>
 5
    using namespace std;
 6
 7
    const int MAXM = 16;
    const int MAXN = 1 << MAXM;</pre>
 8
 9
   // LCA
10
    struct LCA {
11
        vector<int> e[MAXN];
12
        int d[MAXN], p[MAXN] [MAXM];
13
14
15
        void dfs (int v, int f) {
16
            p[v][0] = f;
            for (int i = 1; i < MAXM; ++i) {
17
18
                 p[v][i] = p[p[v][i-1]][i-1];
19
            for (int i = 0; i < (int)e[v].size(); ++i) {</pre>
20
21
                 int w = e[v][i];
22
                 if (w != f) {
23
                     d[w] = d[v] + 1;
                     dfs_(w, v);
24
25
                 }
26
            }
27
28
        int up_(int v, int m) {
29
            for (int i = 0; i < MAXM; ++i) {
30
                 if (m & (1 << i)) {
31
                     v = p[v][i];
32
33
34
35
            return v;
        }
```

```
37
        int lca(int a, int b) {
38
39
            if (d[a] > d[b]) {
40
                 swap(a, b);
41
            b = up_(b, d[b] - d[a]);
42
            if (a == b) {
43
                 return a;
44
45
            } else {
46
                 for (int i = MAXM - 1; i \ge 0; —i) {
47
                     if (p[a][i] != p[b][i]) {
48
                         a = p[a][i];
49
                         b = p[b][i];
50
51
                 return p[a][0];
52
53
            }
54
        }
55
56
        void init(int n) {
57
            for (int i = 0; i < n; ++i) {</pre>
58
                 e[i].clear();
59
        }
60
61
        void add(int a, int b) {
62
            e[a].push_back(b);
63
            e[b].push_back(a);
64
65
66
67
        void build() {
68
            d[0] = 0;
69
            dfs(0, 0);
70
71
    } lca;
```

3.3 树状数组

Listing 8: bit.cpp

```
#include <vector>
 1
 2
 3
    using namespace std;
 5
    template<typename T = int>
 6
    struct BIT {
 7
      vector<T> a;
 8
      void init(int n) {
 9
       vector<T>(n + 1).swap(a);
10
11
12
      void add(int i, T v) {
13
14
        for (int j = i + 1; j < (int)a.size(); j = (j | (j - 1)) + 1) {
15
          a[j] += v;
16
17
18
      // [0, i)
19
      T sum(int i) const {
20
        T ret = T();
21
        for (int j = i; j > 0; j = j \& (j - 1)) {
22
23
          ret += a[j];
24
25
        return ret;
26
```

```
27
28    T get(int i) const {
29        return sum(i + 1) - sum(i);
30    }
31
32    void set(int i, T v) {
33        add(i, v - get(i));
34    }
35    };
```

3.4 并查集

Listing 9: union-find.cpp

```
#include <vector>
 1
 2
 3
    using namespace std;
 4
 5
    struct DisjointSet {
 6
        vector<int> p;
 8
        void init(int n) {
 9
            p.resize(n);
            for (int i = 0; i < n; ++i) {</pre>
10
                 p[i] = i;
11
12
        }
13
14
        int getp(int i) {
15
            return i == p[i] ? i : (p[i] = getp(p[i]));
16
17
18
19
        bool setp(int i, int j) {
20
            i = getp(i);
21
             j = getp(j);
22
            p[i] = j;
23
            return i != j;
24
25
    };
```

3.5 轻重权树剖分

Listing 10: chain-decomp.cpp

```
#include <cstdio>
    #include <vector>
 2
    #include <algorithm>
 3
 4
 5
   using namespace std;
 6
 7
    const int MAXM = 16;
    const int MAXN = 1 << MAXM;</pre>
 8
10
    // Heavy-Light Decomposition
11
    struct TreeDecomposition {
12
      vector<int> e[MAXN], c[MAXN];
                     // subtree size
13
      int s[MAXN];
                      // parent id
      int p[MAXN];
14
                      // chain root id
      int r[MAXN];
15
                      // timestamp, index used in segtree
      int t[MAXN];
16
      int ts;
17
18
19
      void dfs (int v, int f) {
        p[v] = f;
20
21
        s[v] = 1;
```

```
22
        for (int i = 0; i < (int)e[v].size(); ++i) {</pre>
23
          int w = e[v][i];
          if (w != f) {
24
25
            dfs_(w, v);
26
             s[v] += s[w];
27
28
        }
29
      }
30
31
      void decomp (int v, int f, int k) {
32
        t[v] = ts++;
33
        c[k].push back(v);
34
        r[v] = k;
35
        int x = 0, y = -1;
36
37
        for (int i = 0; i < (int)e[v].size(); ++i) {</pre>
          int w = e[v][i];
38
          if (w != f) {
39
            if (s[w] > x) {
40
41
              x = s[w];
               y = w;
42
43
          }
44
45
        if (y !=-1) {
46
47
          decomp_(y, v, k);
48
49
50
        for (int i = 0; i < (int)e[v].size(); ++i) {</pre>
          int w = e[v][i];
51
52
          if (w != f && w != y) {
53
            decomp_(w, v, w);
54
55
        }
      }
56
57
      void init(int n) {
58
        for (int i = 0; i < n; ++i) {</pre>
59
          e[i].clear();
60
61
        }
62
      }
63
64
      void add(int a, int b) {
65
        e[a].push back(b);
66
        e[b].push_back(a);
67
68
      void build() { // !!
69
70
        ts = 0;
71
        dfs_(0, 0);
72
        decomp_(0, 0, 0);
73
74
    } hld;
```

3.6 强连通分量

Listing 11: scc.cpp

```
#include <algorithm>
#include <stack>
#include <vector>

using namespace std;

struct SCCTarjan {
 int n;
```

```
9
        vector<vector<int>> e;
10
11
        vector int id;
12
        vector<vector<int>> scc;
13
14
        void init(int n) {
             this\rightarrown = n;
15
16
             vector<vector<int>> (n) .swap(e);
17
             id.resize(n);
             dfn.resize(n);
18
19
             low.resize(n);
20
21
        void add(int a, int b) {
22
             e[a].push_back(b);
23
24
25
26
        vector<int> dfn, low;
27
        int timestamp;
28
        stack<int> s;
29
30
        void dfs(int v) {
31
             dfn[v] = timestamp++;
32
             low[v] = dfn[v];
             s.push(v);
33
             for (\text{vector} \times \text{int})::const_iterator w = e[v].begin(); w != e[v].end(); ++w) {
34
                 if (dfn[*w] == -1) {
35
                      dfs(*w);
36
37
                      low[v] = min(low[v], low[*w]);
38
                 } else if (dfn[*w] != -2) {
39
                      low[v] = min(low[v], dfn[*w]);
40
41
             }
42
             \textbf{if} (low[v] == dfn[v]) \{
43
                 vector<int> t;
44
                 do {
45
                     int w = s.top();
46
47
                      s.pop();
                      id[w] = (int)scc.size();
48
                      t.push back(w);
49
                      dfn[w] = -2;
50
51
                 } while (t.back() != v);
52
                 scc.push back(t);
53
             }
54
        }
55
56
        int gao() {
57
             scc.clear();
58
             stack<int>().swap(s);
             timestamp = 0;
59
60
61
             fill(dfn.begin(), dfn.end(), -1);
62
             for (int i = 0; i < n; ++i) {</pre>
63
                 if (dfn[i] == -1) {
64
                      dfs(i);
65
66
67
             return (int)scc.size();
68
69
    };
```

3.7 双连通分量

Listing 12: bcc.cpp

```
#include <algorithm>
    #include <stack>
 3
    #include <utility>
    #include <vector>
 5
    using namespace std;
 6
 7
 8
    // TODO: cannot handle duplicate edges
 9
    struct Tarjan {
10
        int n;
         vector<vector<int>> e;
11
12
13
        vector<int> cut;
14
        vector<pair<int, int>> bridge;
        vector<vector<pair<int, int>>> bcc;
15
16
17
        void init(int n) {
             this\rightarrown = n;
18
19
             e.clear();
20
             e.resize(n);
21
             dfn.resize(n);
22
             low.resize(n);
23
24
25
         void add(int a, int b) {
             // assert(find(e[a].begin(), e[a].end(), b) == e[a].end());
26
             e[a].push back(b);
27
28
             e[b].push_back(a);
29
         }
30
         vector<int> dfn, low;
31
32
         int timestamp;
33
         stack<pair<int, int>> s;
34
        \textbf{void} \ \text{dfs}(\textbf{int} \ \textbf{v, int} \ \textbf{p}) \ \ \{
35
             int part = p == -1 ? 0 : 1;
36
             dfn[v] = low[v] = timestamp++;
37
             for (\text{vector} \times \text{int})::const_iterator w = e[v].begin(); w != e[v].end(); ++w) {
38
                 pair<int, int> f = make_pair(min(v, *w), max(v, *w));
39
                 if (dfn[*w] == -1) {
40
41
                      s.push(f);
42
                      dfs(*w, v);
43
                      low[v] = min(low[v], low[*w]);
44
                      if (dfn[v] <= low[*w]) {
45
                           // articulation point
46
                          if (++part == 2) {
47
                               cut.push_back(v);
48
                           // articulation edge
49
                          if (dfn[v] < low[*w]) {
50
                               bridge.push_back(f);
51
52
53
                           // biconnected component (2-vertex-connected)
54
                          vector<pair<int, int>> t;
55
                          do {
56
                               t.push_back(s.top());
57
                               s.pop();
                           } while (t.back() != f);
58
                          bcc.push_back(t);
59
60
61
                  } else if (*w != p && dfn[*w] < dfn[v]) {</pre>
62
                      s.push(f);
                      low[v] = min(low[v], dfn[*w]);
63
64
65
             }
```

```
66
67
         void gao() {
68
             cut.clear();
69
             bridge.clear();
70
             bcc.clear();
71
 72
 73
             timestamp = 0;
 74
             stack<pair<int, int>>().swap(s);
 75
             fill(dfn.begin(), dfn.end(), -1);
 76
             for (int i = 0; i < n; ++i) {</pre>
77
                  if (dfn[i] == -1) {
 78
                      dfs(i, -1);
79
80
81
82
83
     };
84
     struct BridgeBlockTree {
85
86
         Tarjan≪MAXND bcc;
87
         DisjointSet<MAXN▷ ds;
88
         vector<int> e[MAXN];
89
         void init(int n) {
90
             bcc.init(n);
91
             ds.init(n);
92
93
         }
94
95
         void add(int a, int b) {
96
             bcc.add(a, b);
97
98
         void gao() {
99
             bcc.gao();
100
             for (const auto &i : bcc.bcc) {
101
                  if (i.size() > 1) {
102
                      for (const auto &j : i) {
103
                          ds.setp(j.first, j.second);
104
105
                  }
106
107
108
             for (const auto &i : bcc.bridge) {
109
                  int a = ds.getp(i.first);
110
                  int b = ds.getp(i.second);
111
                  e[a].push_back(b);
112
                  e[b].push_back(a);
113
              }
         }
114
115
         int id(int v) {
116
             return ds.getp(v);
117
118
119
     };
```

3.8 二分图匹配

Listing 13: bimatch.cpp

```
// maximum matchings in bipartite graphs
// maximum cardinality bipartite matching
// O(|V||E|), generally fast

finclude <algorithm
finclude <string
finclude <vector</pre>
```

```
8
 9
    using namespace std;
10
11
    struct Hungarian {
12
        int nx, ny;
        vector<int> mx, my;
13
        vector<vector<int>> e;
14
15
16
        void init(int nx, int ny) {
17
            this\rightarrownx = nx;
18
            this->ny = ny;
19
            mx.resize(nx);
20
            my.resize(ny);
21
            e.clear();
            e.resize(nx);
22
23
            mark.resize(nx);
24
25
26
        void add(int a, int b) {
27
            e[a].push back(b);
28
29
30
        // vector<bool> is evil!!!
31
        basic_string<bool> mark;
32
        bool augment(int i) {
33
            if (!mark[i]) {
34
35
                 mark[i] = true;
36
                  \textbf{if} \ (\texttt{my}[*\texttt{j}] == -1 \ || \ \texttt{augment}(\texttt{my}[*\texttt{j}])) \ \{ \\
37
38
                         mx[i] = *j;
39
                         my[*j] = i;
40
                         return true;
41
42
                 }
43
            return false;
44
45
46
47
        int gao() {
            int ret = 0;
48
            fill(mx.begin(), mx.end(), -1);
49
50
            fill(my.begin(), my.end(), -1);
51
            for (int i = 0; i < nx; ++i) {</pre>
52
                 fill(mark.begin(), mark.end(), false);
53
                 if (augment(i)) {
54
                     ++ret;
55
56
57
            return ret;
58
59
    };
```

3.9 最小费用最大流

Listing 14: flow.cpp

```
#include <algorithm>
#include <cstdio>
#include <limits>
#include <queue>
#include <vector>

#include <vector>

using namespace std;

template <int MAXN, typename T = int, typename S = T>
```

```
struct MinCostMaxFlow {
10
11
        struct NegativeCostCircuitExistsException {
12
1.3
        struct Edge {
14
            int v;
15
            T c;
16
17
             S w;
18
             int b;
             Edge(int v, T c, S w, int b) : v(v), c(c), w(w), b(b) {}
19
20
21
22
        int n, source, sink;
23
        vector Edge> e [MAXN];
24
25
        void init(int n, int source, int sink) {
26
             this\rightarrown = n;
27
             this->source = source;
28
             this->sink = sink;
             for (int i = 0; i < n; ++i) {</pre>
29
30
                 e[i].clear();
31
32
        }
33
        void addEdge(int a, int b, T c, S w) {
34
             e[a].push_back(Edge(b, c, w, e[b].size()));
35
             e[b].push_back(Edge(a, 0, -w, e[a].size() -1)); // TODO
36
37
        }
38
39
        bool mark[MAXN];
40
        T maxc[MAXN];
41
        S minw[MAXN];
42
        int dist[MAXN];
43
        Edge *prev[MAXN];
44
        bool _spfa() {
45
46
            queue<int> q;
             fill(mark, mark + n, false);
47
            fill(maxc, maxc + n, 0);
48
49
             fill(minw, minw + n, numeric_limits<S>::max());
50
             fill(dist, dist + n, 0);
             fill(prev, prev + n, (Edge *)NULL);
51
52
            mark[source] = true;
53
            maxc[source] = numeric limits<S>::max();
54
            minw[source] = 0;
55
56
             q.push (source);
57
             while (!q.empty()) {
                 int cur = q.front();
58
59
                 mark[cur] = false;
60
                 q.pop();
                 for (typename vector Edge>::iterator it = e[cur].begin(); it != e[cur].end(); ++it) {
61
62
                     T c = min(maxc[cur], it->c);
63
                     if (c == 0) {
64
                         continue;
65
66
67
                     int v = it->v;
68
                     S w = minw[cur] + it \rightarrow w;
                     if (minw[v] > w \mid \mid (minw[v] == w \&\& maxc[v] < c)) { // TODO
69
70
                         maxc[v] = c;
                         minw[v] = w;
71
72
                         dist[v] = dist[cur] + 1;
73
                         if (dist[v] >= n) {
74
                              return false;
75
                          }
```

```
76
                           prev[v] = &*it;
77
                           if (!mark[v]) {
78
                               mark[v] = true;
79
                               q.push(v);
                           }
80
81
                       }
82
                  }
83
84
             return true;
85
86
         pair<T, S> gao() {
87
             T sumc = 0;
88
             S sumw = 0;
89
             while (true) {
90
                  if (!_spfa()) {
91
                      throw NegativeCostCircuitExistsException();
92
93
                  } else if (maxc[sink] == 0) {
94
                      break;
                  } else {
95
                      T c = maxc[sink];
96
97
                      sumc += c;
98
                      sumw += c * minw[sink];
99
                      int cur = sink;
100
                      while (cur != source) {
101
                          Edge *e1 = prev[cur];
102
                           e1->c -= c;
103
104
                          Edge *e2 = \&e[e1->v][e1->b];
105
                           e2->c += c;
106
                           cur = e2 \rightarrow v;
107
                       }
108
                  }
109
110
              return make_pair(sumc, sumw);
111
112
     };
```

3.10 AhoCorasick 自动机

Listing 15: ac-automata.cpp

```
#include <algorithm>
    #include <queue>
 3
 4
   using namespace std;
 5
    struct AhoCorasick {
 6
        static const int NONE = 0;
 7
        static const int MAXN = 1024;
 8
        static const int CHARSET = 26;
 9
10
        int end;
11
        int tag[MAXN];
12
13
        int fail[MAXN];
14
        int trie[MAXN] [CHARSET];
15
        void init() {
16
            tag[0] = NONE;
17
            fill(trie[0], trie[0] + CHARSET, -1);
18
            end = 1;
19
20
21
22
        int add(int m, const int *s) {
23
            int p = 0;
            for (int i = 0; i < m; ++i) {</pre>
```

```
25
                 if (trie[p] [*s] == -1) {
26
                     tag[end] = NONE;
                     fill(trie[end], trie[end] + CHARSET, -1);
27
28
                     trie[p][*s] = end++;
29
                 }
30
                 p = trie[p][*s];
31
                 ++s;
32
33
            return p;
34
        }
35
36
        void build(void) { // !!
37
             queue int bfs;
             fail[0] = 0;
38
             for (int i = 0; i < CHARSET; ++i) {</pre>
39
                 if (trie[0][i] != −1) {
40
                     fail[trie[0][i]] = 0;
41
42
                     bfs.push(trie[0][i]);
43
                 } else {
                     trie[0][i] = 0;
44
45
46
47
            while (!bfs.empty()) {
48
                 int p = bfs.front();
                 tag[p] |= tag[fail[p]];
49
50
                 bfs.pop();
                 for (int i = 0; i < CHARSET; ++i) {</pre>
51
                     if (trie[p][i] != −1) {
52
53
                          fail[trie[p][i]] = trie[fail[p]][i];
54
                         bfs.push(trie[p][i]);
55
                     } else {
56
                         trie[p][i] = trie[fail[p]][i];
57
58
59
             }
60
61
    } ac;
```

3.11 后缀数组

Listing 16: sa.cpp

```
#include <algorithm>
    #include <utility>
    #include <vector>
 4
   using namespace std;
 5
 6
    struct SuffixArray {
 7
        vector int sa, rank, height;
 8
 9
        void init(int n, const T a[]) {
10
            sa.resize(n);
11
            rank.resize(n);
12
13
14
            vector<pair<T, int>> assoc(n);
15
            for (int i = 0; i < n; ++i) {</pre>
16
                assoc[i] = make pair(a[i], i);
17
            sort(assoc.begin(), assoc.end());
18
            for (int i = 0; i < n; ++i) {</pre>
19
                sa[i] = assoc[i].second;
20
21
                if (i == 0 \mid | assoc[i].first != assoc[i - 1].first) {
22
                    rank[sa[i]] = i;
23
                } else {
24
                    rank[sa[i]] = rank[sa[i-1]];
```

```
25
26
             }
27
28
            vector<int> tmp(n), cnt(n);
             vector<pair<int, int>> suffix(n);
29
             for (int m = 1; m < n; m <<= 1) {</pre>
30
                 // snd
31
                 for (int i = 0; i < m; ++i) {</pre>
32
33
                     tmp[i] = n - m + i;
34
35
                 for (int i = 0, j = m; i < n; ++i) {
36
                     if (sa[i] >= m) {
37
                         tmp[j++] = sa[i] - m;
38
                     }
                 }
39
                 // fst
40
                 fill(cnt.begin(), cnt.end(), 0);
41
42
                 for (int i = 0; i < n; ++i) {</pre>
43
                     ++cnt[rank[i]];
44
                 partial sum(cnt.begin(), cnt.end(), cnt.begin());
45
46
                 for (int i = n - 1; i >= 0; —i) {
47
                     sa[--cnt[rank[tmp[i]]] = tmp[i];
48
                 }
                 //
49
                 for (int i = 0; i < n; ++i) {</pre>
50
                     suffix[i] = make pair(rank[i], i + m < n ? rank[i + m] : numeric_limits<int>::min());
51
52
53
                 for (int i = 0; i < n; ++i) {</pre>
                     if (i == 0 || suffix[sa[i]] != suffix[sa[i - 1]]) {
54
55
                         rank[sa[i]] = i;
56
                     } else {
57
                         rank[sa[i]] = rank[sa[i-1]];
58
59
                 }
             }
60
61
            height.resize(n);
62
             for (int i = 0, z = 0; i < n; ++i) {
63
                 if (rank[i] == 0) {
64
                     height[0] = z = 0;
65
                 } else {
66
67
                     int x = i, y = sa[rank[i] - 1];
68
                     z = \max(0, z - 1);
69
                     while (x + z < n \&\& y + z < n \&\& a[x + z] == a[y + z])  {
70
                         ++z;
                     }
71
72
                     height[rank[i]] = z;
73
                 }
74
             }
75
        }
76
    };
```

3.12 LU 分解

Listing 17: lu.cpp

```
1  const int MAXN = 128;
2  const double EPS = 1e-10;
3
4  void LU(int n, double a[MAXN] [MAXN], int r[MAXN], int c[MAXN]) {
5   for (int i = 0; i < n; ++i) {
6     r[i] = c[i] = i;
7   }
8   for (int k = 0; k < n; ++k) {
9   int ii = k, jj = k;</pre>
```

```
10
              for (int i = k; i < n; ++i) {</pre>
                   for (int j = k; j < n; ++j) {
11
                       if (fabs(a[i][j]) > fabs(a[ii][jj])) {
12
                            ii = i;
13
                            jj = j;
14
15
                        }
                   }
16
17
18
              swap(r[k], r[ii]);
19
              swap(c[k], c[jj]);
20
              for (int i = 0; i < n; ++i) {</pre>
21
                   swap(a[i][k], a[i][jj]);
22
              for (int j = 0; j < n; ++j) {
23
                   swap(a[k][j], a[ii][j]);
24
25
26
              \textbf{if} \hspace{0.1cm} (fabs\hspace{0.1cm} (a\hspace{0.1cm} [\hspace{0.1cm} k]\hspace{0.1cm} [\hspace{0.1cm} k]\hspace{0.1cm}) \hspace{0.1cm} < \hspace{0.1cm} \texttt{EPS}) \hspace{0.1cm} \{
27
                   continue;
28
              for (int i = k + 1; i < n; ++i) {</pre>
29
                   a[i][k] = a[i][k] / a[k][k];
30
31
                   for (int j = k + 1; j < n; ++j) {
32
                       a[i][j] = a[i][k] * a[k][j];
33
34
              }
         }
35
36
    }
37
    void solve(int n, double a[MAXN] [MAXN], int r[MAXN], int c[MAXN], double b[MAXN]) {
38
39
         static double x[MAXN];
40
         for (int i = 0; i < n; ++i) {</pre>
41
              x[i] = b[r[i]];
42
43
         for (int i = 0; i < n; ++i) {</pre>
              for (int j = 0; j < i; ++j) {
44
                   x[i] = a[i][j] * x[j];
45
46
47
         for (int i = n - 1; i >= 0; —i) {
48
              for (int j = n - 1; j > i; — j) {
49
                   x[i] = a[i][j] * x[j];
50
51
52
              if (fabs(a[i][i]) >= EPS) {
53
                   x[i] /= a[i][i];
54
              55
         for (int i = 0; i < n; ++i) {</pre>
56
57
              b[c[i]] = x[i];
58
59
60
61
    // LU(n - 1, a, r, c);
    // solve(n - 1, a, r, c, b);
```

4 对一类问题的处理方法