

Alfred 代码模版库

目录

目	录			5.5 二分图最大权匹配 (MaxAssignment,基	10
1	比赛配置 and 奇技淫巧 1.1 多组数据代码模板 1.2 快读快写 1.3 关闭流与 C 风格输入输出的同步 1.4 .clang-format 1.5 debug.h 1.6 火车头 1.7 c-cpp-properties.json 1.8 launch.json 1.9 settings.json 1.10 tasks.json	3 3 3 4 4 5 6 7 7		5.8 最大流	18 20 22 23 24 25 27 28 28 28 29 29
2	数据结构 2.1 珂朵莉树 2.2 树状数组 2.3 静态可重区间信息(支持 RMQ) 2.4 PBDS 大常数平衡树 2.5 离散化容器 2.6 并查集 2.7 出现次数统计	8 8 9 9 10 10 11 11	6	5.18 取模类,按需写	30 33 36 37 37 38 38 39
3	数学(数论)算法 3.1 带模整数类	12 12 13		6.4 并查集	40 41 41 42
	字符串算法 4.1 字符串哈希	14 14		6.8 二分图匹配	43 45 46
5	jiangly 代码库 (备用,侵权请提出 issue) 5.1 int128 输出流自定义	15 15 15 15 16		6.10 AhoCorasick 自动机	48 49 50 51

1 比赛配置 and 奇技淫巧

1.1 多组数据代码模板

Listing 1: template.cpp

```
#include <bits/stdc++.h>
2
   using namespace std;
3 using i64 = long long;
   const i64 N = 1e5 + 10;
4
5
   int t = 1;
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
13
    inline void init(void) {}
    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
   }
```

1.2 快读快写

Listing 2: fast-io.cpp

```
1
    namespace fastIO {
2
        char c, f, e = 0;
3
        namespace usr {
            template <class Tp>
4
            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
            template <class Tp>
11
12
            inline void write( Tp x) {
13
                if (x < 0) putchar('-'), x = -x;
14
                if (x > 9) write(x / 10);
15
                putchar((x % 10) ^ 48);
16
17
            template <typename \mathbb{T}, typename... \mathbb{V}>
            inline void read(T &t, V &...v) { read(t), read(v...); }
18
            template <typename T, typename... V >
19
20
            inline void write(T t, V... v) {
21
                write(t), putchar('_'), write(v...);
22
23
24
   using namespace fastIO::usr;
```

1.3 关闭流与 C 风格输入输出的同步

Listing 3: io-sync-off.cpp

```
1 inline void optimizeIO(void) {
2    ios::sync_with_stdio(false);
3    cin.tie(NULL), cout.tie(NULL);
4 }
```

1.4 .clang-format

Listing 4: .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
16 FixNamespaceComments: false
17
   AllowShortCaseLabelsOnASingleLine: true
18 AlwaysBreakTemplateDeclarations: MultiLine
19 BinPackParameters: true
20 BraceWrapping:
     AfterCaseLabel: true
21
     AfterClass: true
22
23 AlignConsecutiveMacros: AcrossEmptyLinesAndComments
24 AlignTrailingComments: Always
```

1.5 debug.h

Listing 5: debug.h

```
/**
 1
     * @file
 2
                        debug.h
 3
      * @author
                        Dr.Alfred (abonlinejudge@163.com)
      * @brief
 4
                        Local Debug Printer
      * @version
                        1.0
 5
      * @date
                        2023-12-30
 6
 7
 8
      * @copyright Copyright (c) 2019—now < Rhodes Island Inc.>
 9
      */
10
11
     #include <bits/stdc++.h>
12
13
14 using std::cerr;
15 using std::pair;
16 using std::string;
17
18
    const long long dbg_inf = 9e18 + 19260817;
19
20
    void __print(int x) { cerr << x; }</pre>
21
    \begin{tabular}{ll} \textbf{void} & \_\texttt{print}(\textbf{long} \ \texttt{x}) & \{ \ \texttt{cerr} << \ \texttt{x}; \ \} \\ \end{tabular}
            print(long long x) {
22
23
         if (x != dbg_inf) {
24
              cerr << x;
25
         } else {
              cerr << "inf";</pre>
26
27
    }
28
    void print(unsigned x) { cerr << x; }</pre>
29
    void print(unsigned long x) { cerr << x; }</pre>
30
31
    void print(unsigned long long x) { cerr << x; }</pre>
    void print(float x) { cerr << x; }</pre>
    void print(double x) { cerr << x; }</pre>
```

1 比赛配置 AND 奇技淫巧 1.6 火车头

```
void print(long double x) { cerr << x; }</pre>
    void __print(char x) { cerr << '\'' << x << '\''; }</pre>
    void __print(const char *x) { cerr << '\"' << x << '\"'; }</pre>
36
    \begin{tabular}{ll} \textbf{void} \begin{tabular}{ll} \textbf{print}(\textbf{const} \ \text{string \&x}) & \{ \ \text{cerr} << \ ' \ ''' << \ x << \ ' \ '''; \ \} \\ \end{tabular}
37
    void __print(bool x) { cerr << (x ? "true" : "false"); }</pre>
38
    void __print(__int128_t x) {
39
         if (x < 0) cerr << '-', x = -x;
40
41
         if (x > 9) __print(x / 10);
         cerr << char((x % 10) ^ 48);
42
43
    }
44
    void dbgEndl(void) { cerr << '\n'; }</pre>
45
46
    template <typename T, typename V>
    void print(const pair<T, V> &x) {
47
         cerr << '{', __print(x.first), cerr << ",_", __print(x.second), cerr << '}';</pre>
48
49
    template <typename T>
50
    void print(const T &x) {
51
52
         int f = 0;
         cerr << '{';
53
         for (auto i : x) cerr << (f++ ? ", " : ""),     print(i);</pre>
54
55
         cerr << "}";
56
    }
    void print() { cerr << "]\n"; }</pre>
57
58
    template <typename T, typename... V>
    void print(T t, V... v) {
59
          _print(t);
60
         if (sizeof...(v)) cerr << ", _";
61
62
         _print(v...);
63
64
    #ifdef DEBUG
65
    // To customize a struct/class to print, just define the print function.
66
67
    #ifndef NO DBG COLOR
68
    #define dbg(x...)
         cerr << "\e[91m" << __func__ << ":" << __LINE__ << "_[" << #x << "] _=_["; \
69
70
         _print(x);
         cerr << "\e[39m";
71
72
73
    #define short dbg(x...) \
        cerr << "\e[91m[["; \
74
         print(x);
75
76
         cerr << "\e[39m";
77
78
    #define dbg(x...)
         cerr << __func__ << ":" << __LINE__ << "_[" << #x << "] _=_["; \
79
80
         print(x);
    #define short_dbg(x...) \
81
        cerr << "[";
82
         print(x);
83
84
    #endif // !NO DBG COLOR
85
86
87
    #define dbg(x...)
88
    #endif
```

1.6 火车头

Listing 6: optimize-header.h

```
1  #pragma GCC optimize(3)
2  #pragma GCC target("avx")
3  #pragma GCC optimize("Ofast")
4  #pragma GCC optimize("inline")
5  #pragma GCC optimize("-fgcse")
6  #pragma GCC optimize("-fgcse-lm")
```

```
#pragma GCC optimize("-fipa-sra")
    #pragma GCC optimize("-ftree-pre")
    #pragma GCC optimize("-ftree-vrp")
9
    #pragma GCC optimize("-fpeephole2")
10
    #pragma GCC optimize("-ffast-math")
11
    #pragma GCC optimize("-fsched-spec")
12
    #pragma GCC optimize("unroll-loops")
13
    #pragma GCC optimize("-falign-jumps")
14
    #pragma GCC optimize("-falign-loops")
15
    #pragma GCC optimize("-falign-labels")
16
    #pragma GCC optimize("-fdevirtualize")
17
    #pragma GCC optimize("-fcaller-saves")
18
    #pragma GCC optimize("-fcrossjumping")
19
    #pragma GCC optimize("-fthread-jumps")
20
    #pragma GCC optimize("-funroll-loops")
21
    #pragma GCC optimize("-fwhole-program")
22
    #pragma GCC optimize("-freorder-blocks")
23
    #pragma GCC optimize("-fschedule-insns")
24
    #pragma GCC optimize("inline-functions")
25
    #pragma GCC optimize("-ftree-tail-merge")
26
    #pragma GCC optimize("-fschedule-insns2")
27
    #pragma GCC optimize("-fstrict-aliasing")
28
    #pragma GCC optimize("-fstrict-overflow")
29
    #pragma GCC optimize("-falign-functions")
30
    #pragma GCC optimize("-fcse-skip-blocks")
31
    #pragma GCC optimize("-fcse-follow-jumps")
32
    #pragma GCC optimize("-fsched-interblock")
33
    #pragma GCC optimize("-fpartial-inlining")
34
35
    #pragma GCC optimize("no-stack-protector")
36
    #pragma GCC optimize("-freorder-functions")
37
    #pragma GCC optimize("-findirect-inlining")
38
    #pragma GCC optimize("-fhoist-adjacent-loads")
39
    #pragma GCC optimize("-frerun-cse-after-loop")
40
    #pragma GCC optimize("inline-small-functions")
    #pragma GCC optimize("-finline-small-functions")
41
    #pragma GCC optimize("-ftree-switch-conversion")
42
    #pragma GCC optimize("-foptimize-sibling-calls")
43
    #pragma GCC optimize("-fexpensive-optimizations")
44
    #pragma GCC optimize("-funsafe-loop-optimizations")
45
    #pragma GCC optimize("inline-functions-called-once")
46
    #pragma GCC optimize("-fdelete-null-pointer-checks")
```

1.7 c-cpp-properties.json

Listing 7: c-cpp-properties.json

```
1
2
        "configurations": [
3
                 "name": "macos-gcc-arm64",
4
                 "includePath": [
5
                     "${workspaceFolder}/**",
6
                     "/usr/local/include/ac-library/"
7
8
                 "compilerPath": "/usr/local/bin/g++",
9
                 "cStandard": "c17",
10
                 "cppStandard": "c++20",
11
                 "intelliSenseMode": "macos-gcc-arm64",
12
13
                 "compilerArgs": [],
                 "configurationProvider": "ms-vscode.makefile-tools"
14
15
            }
16
        ],
        "version": 4
17
18
    }
```

1.8 launch.json

Listing 8: launch.json

```
{
1
        "version": "0.2.0",
2
        "configurations": [
3
4
            {
                 "name": "(lldb) Launch",
5
                 "type": "cppdbg",
6
                "request": "launch",
7
8
                 "program": "${fileDirname}/compiled.out",
9
                 "args": [],
                "stopAtEntry": false,
10
                 "cwd": "dollar{fileDirname}",
11
                 "environment": [],
12
                 "externalConsole": true,
13
                 "internalConsoleOptions": "neverOpen",
14
                 "MIMode": "lldb",
15
                 "setupCommands": [
16
17
                     {
18
                         "description": "Enable_pretty-printing_for_lldb",
19
                         "text": "-enable-pretty-printing",
20
                         "ignoreFailures": false
21
22
                 "preLaunchTask": "Compile"
23
24
            }
25
        ],
    }
26
```

1.9 settings.json

Listing 9: settings.json

```
1
   {
2
       "files.defaultLanguage": "cpp",
3
       "editor.formatOnType": true,
       "editor.suggest.snippetsPreventQuickSuggestions": false,
4
       "editor.acceptSuggestionOnEnter": "off",
5
       "C Cpp.clang_format_sortIncludes": true,
6
       "C Cpp.errorSquiggles": "disabled",
7
       "C_Cpp.default.defines": ["LOCAL", "DEBUG"]
8
9
   }
```

1.10 tasks.json

Listing 10: tasks.json

```
// https://code.visualstudio.com/docs/editor/tasks
1
2
       "version": "2.0.0",
3
      "tasks": [
4
5
          {
              "label": "Compile", // 任务名称,与launch.json的preLaunchTask相对应
6
             "command": "g++", // 要使用的编译器, C++用g++
7
              "args": [
8
                 "${file}",
9
                 "-o", // 指定输出文件名,不加该参数则默认输出a.exe, Linux下默认a.out
10
11
                 "${fileDirname}/compiled.out",
12
                 "-g", // 生成和调试有关的信息
13
                 // "-arch aarch64",
                 // "-m64", // 不知为何有时会生成16位程序而无法运行,此条可强制生成64位的
14
                 "-Wall", // 开启额外警告
15
                 "-std=c++20", // c++14
16
```

```
"-DLOCAL",
17
18
                "-DDEBUG",
                "-03",
19
                "-ld classic", // will be deprecated
20
                "-Wno-char-subscripts",
21
22
                "-I",
                "/usr/local/include/ac-library/"
23
                                         // 手动扩大栈空间
24
                // "-stack=268435456"
             ], // 编译的命令, 其实相当于VSC帮你在终端中输了这些东西
25
             "type": "process", // process是把预定义变量和转义解析后直接全部传给command: shell相当于先打开
26
              shell再输入命令,所以args还会经过shell再解析一遍
27
             "group": {
                "kind": "build",
28
                "isDefault": true // 不为true时ctrl shift B就要手动选择了
29
30
             "presentation": {
31
                "echo": true,
32
                "reveal": "always", // 执行任务时是否跳转到终端面板,可以为always, silent, never。具体参见
33
                 VSC的文档,即使设为never,手动点进去还是可以看到
                "focus": false, // 设为true后可以使执行task时焦点聚集在终端,但对编译C/C++来说,设为true没
34
                "panel": "shared" // 不同的文件的编译信息共享一个终端面板
35
36
             "problemMatcher": "$gcc" // 捕捉编译时终端里的报错信息到问题面板中,修改代码后需要重新编译才会
37
              再次触发
             // 本来有Lint,再开problemMatcher就有双重报错,但MinGw的Lint效果实在太差了;用Clange可以注释掉
38
39
         }
40
      1
41
   }
```

2 数据结构

2.1 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 11: chtholly.cpp

```
#include <set>
1
2
3
    struct ChthollyTree {
4
        typedef long long 11;
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;
        ChthollyTree(void) = default;
11
12
        ChthollyTree(int rng, int val) { init(rng, val); }
        inline void init(ll rng, ll val) noexcept {
13
14
            tr.insert({1, rng, val}), tr.insert({rng + 1, rng + 1, 0});
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->1 == pos) return it;
            11 1 = (--it) -> 1, r = it -> r, v = it -> v;
21
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
26
            auto R = split(r + 1), L = split(l);
27
            tr.erase(L, R), tr.insert({1, r, v});
```

2 数据结构 2.2 树状数组

```
28
        template <class Functor> // func(iterator)
29
        inline void modify(ll 1, ll r, _Functor func) {
30
            auto R = split(r + 1), L = split(l);
31
            for (auto it = L; it != R; it++) func(it);
32
33
        template <class Functor> // func(ll &, iterator)
34
35
        inline 11 query(11 1, 11 r, _Functor func) {
36
            11 \text{ ans} = 0;
37
            auto R = split(r + 1);
38
            for (auto it = split(l); it != R; it++) func(ans, it);
39
            return ans;
40
41
    } ;
```

2.2 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 12: fenwick.cpp

```
template <class T>
2
    struct Fenwick {
3
        std::vector<T> c;
4
        inline int lowbit(int x) { return x & -x; }
5
        inline void merge (T &x, T &y) { x = x + y; }
6
        inline T subtract(T x, T y) { return x - y; }
7
        inline void update(size_t pos, T x) {
8
            for (pos++; pos < c.size(); pos += lowbit(pos)) merge(c[pos], x);</pre>
9
        inline void clear(void) {
10
            for (auto &x : c) x = T();
11
12
13
        inline T query(size_t pos) {
14
            T ans = T();
            for (pos++; pos; pos ^= lowbit(pos)) merge(ans, c[pos]);
15
16
            return ans;
17
18
        inline T query(size_t l, size_t r) {
            return subtract(query(r), query(l - 1));
19
20
        Fenwick(size t len) : c(len + 2) {}
21
   } ;
22
```

2.3 静态可重区间信息(支持 ${ m RMQ}$)

基于 ST 表,支持静态数组可重区间信息的数据结构。

Listing 13: sparse-table.cpp

```
1
    template <class T>
    struct MaxInfo {
2
        T val;
3
4
        MaxInfo() { val = T(); }
5
        template <class InitT>
        MaxInfo(InitT x) { val = x; }
6
7
        MaxInfo operator+ (MaxInfo &x)
8
            return {std::max(val, x.val)};
9
10
    } ;
    template <class T>
11
    struct MinInfo {
12
        T val:
13
        MinInfo() { val = T(); }
14
        template <class InitT>
15
```

```
16
        MinInfo(InitT x) { val = x; }
17
        MinInfo operator+ (MinInfo &x)
18
            return {std::min(val, x.val)};
19
20
    };
    template <class T>
21
    struct GcdInfo {
22
23
        T val;
        GcdInfo() { val = T(); }
24
        template <class InitT>
25
26
        GcdInfo(InitT x) { val = x; }
27
        GcdInfo operator+ (GcdInfo &x) {
28
            return {std::gcd(val, x.val)};
29
    };
30
31
    template <class T>
    struct SparseTable {
32
   private:
33
34
        int n;
        std::vector<std::vector<T>> ST;
35
36
37
   public:
38
        SparseTable() {}
        SparseTable(int N) : n(N), ST(N, std::vector<T>( lg(N) + 1)) {}
39
40
        template <class InitT>
        SparseTable(std::vector<InitT> init) : SparseTable(init.size()) {
41
            for (int i = 0; i < n; i++) ST[i][0] = T(init[i]);</pre>
42
            for (int i = 1; (1 << i) <= n; i++) {</pre>
43
44
                for (int j = 0; j + (1 << i) - 1 < n; j++) {
45
                     ST[j][i] = ST[j][i-1] + ST[j + (1 << (i-1))][i-1];
46
47
48
49
        inline T query(int 1, int r) { // 0 based
            int w = std::__lg(r - l + 1);
50
            return ST[1] [w] + ST[r - (1 << w) + 1] [w];
51
52
    };
53
```

2.4 PBDS 大常数平衡树

GNU PBDS 提供的大常数基于 rb-tree 的平衡树。

Listing 14: pbds-balance-tree.cpp

```
#include <bits/extc++.h>
#include <bits/stdc++.h>

using namespace std;

using namespace __gnu_pbds;

// TreeTag can also be __gnu_pbds::splay_tree_tag

template <class T, class Cmp, class TreeTag = rb_tree_tag>
using BalanceTree = tree<T, null_type, Cmp, TreeTag, tree_order_statistics_node_update>;
```

2.5 离散化容器

Listing 15: discretization.cpp

```
1 #include <vector>
2
3 template <class _Tp>
4 struct Mess {
5    std::vector<_Tp> v;
6    bool initialized = false;
```

2 数据结构 2.6 并查集

```
7
        inline Tp origin(int idx) { return v[idx - 1]; }
8
        inline void insert( Tp x) { v.push back(x); }
9
        template <typename \mathbb{T}, typename... \mathbb{V}>
10
        inline void insert(T x, V... v) { insert(x), insert(v...); }
        inline void init(void) {
11
12
            sort(v.begin(), v.end()), initialized = true;
            v.erase(unique(v.begin(), v.end()), v.end());
13
14
        inline int query(_Tp x) {
15
16
            if (!initialized) init();
17
            return lower bound(v.begin(), v.end(), x) - v.begin() + 1;
18
19
        inline bool exist( Tp x) { return origin(query(x)) == x; }
20
    } ;
```

2.6 并查集

Listing 16: dsu.cpp

```
struct DSU {
1
2
        std::vector<int> fa;
        DSU(int n) : fa(n + 1) {
3
4
            iota(fa.begin(), fa.end(), 0);
5
6
        inline int find(int x) {
7
            return fa[x] == x ? x : fa[x] = find(fa[x]);
8
        inline void merge(int x, int y) {
9
            int fx = find(x), fy = find(y);
10
            if (fx != fy) fa[fx] = fy;
11
12
13
   };
```

2.7 出现次数统计

O(nlogn) 预处理, O(logn) 查找的出现次数在线统计

Listing 17: appear-statistics.cpp

```
#include <bits/stdc++.h>
 1
 2
 3
    template <class _Tp>
 4
    struct Mess {
 5
        std::vector< Tp> v;
        bool initialized = false;
 6
 7
        inline _Tp origin(int idx) { return v[idx - 1]; }
        inline void insert(_Tp x) { v.push_back(x); }
 8
 9
        template <typename \mathbb{T}\text{, typename}\dots\ \mathbb{V}\text{>}
10
        inline void insert(T x, V... v) { insert(x), insert(v...); }
11
        inline void init(void) {
             sort(v.begin(), v.end()), initialized = true;
12
             v.erase(unique(v.begin(), v.end()), v.end());
13
14
15
        inline int query(_Tp x) {
            if (!initialized) init();
16
             return lower_bound(v.begin(), v.end(), x) - v.begin() + 1;
17
18
        inline bool exist(_Tp x) { return origin(query(x)) == x; }
19
20
    };
21
22
    template <class T>
23
   class AppearStats { // Appear Statistics.
24
   private:
25
        Mess<T> M;
26
        size_t n;
```

```
27
        std::vector<std::vector<int>> pos;
28
29
    public:
30
        AppearStats(void) : n(0) {}
        AppearStats(std::vector<T> &init) : n(init.size()) { init(init); }
31
        inline void _init(std::vector<T> &init) {
32
33
             for (auto item : init) M.insert(item);
34
             n = init.size(), M.init(), pos.resize(M.v.size());
35
             for (size t i = 0; i < n; i++) {</pre>
36
                 pos[M.query(init[i]) - 1].push_back(i);
37
38
39
        // Use \lceil base \rceil as the beginning of index, return -1 if x doesn't exist.
40
        inline int first(int 1, int r, T x, int base = 0) {
             l \rightarrow base, r \rightarrow base;
41
             if (!M.exist(x)) return -1;
42
             std::vector\leqint> &P = pos[M.query(x) - 1];
43
             auto it = std::lower_bound(P.begin(), P.end(), 1);
44
45
             return it == P.end() || *it > r ? -1 : *it + base;
46
47
        // Use \begin{bmatrix} base \end{bmatrix} as the beginning of index, return -1 if x doesn't exist.
        inline int last(int 1, int r, T x, int base = 0) {
48
49
             1 -= base, r -= base;
50
             if (!M.exist(x)) return -1;
51
             std::vector\langle int \rangle &P = pos[M.query(x) - 1];
             auto it = std::upper bound(P.begin(), P.end(), r);
52
             return it == P.begin() || *std::prev(it) < 1 ? -1 : *std::prev(it) + base;</pre>
53
54
55
        inline int count(int 1, int r, T x, int base = 0) {
56
             1 -= base, r -= base;
57
             if (!M.exist(x)) return 0;
58
             std::vector(int) &P = pos[M.query(x) - 1];
59
             auto L = std::lower bound(P.begin(), P.end(), 1);
60
             auto R = std::upper bound(P.begin(), P.end(), r);
             if (L == P.end() || R == P.begin()) return 0;
61
             if (*L > r || *std::prev(R) < 1) return 0;</pre>
62
             return R - L;
63
64
    };
65
```

3 数学(数论)算法

3.1 带模整数类

Listing 18: mod-int.cpp

```
1 template <int mod>
   inline int64_t down(int64_t x) { return x \ge mod ? x - mod : x; }
3
   template <int mod>
4
   struct ModInt {
       int64 t x;
5
       ModInt() = default;
6
7
       ModInt(int64_t x) : x((x % mod + mod) % mod) {}
8
       friend istream &operator>>(istream &in, ModInt &a) { return in >> a.x; }
       friend ostream &operator<<(ostream &out, ModInt a) { return out << a.x; }</pre>
9
       friend ModInt operator+ (ModInt a, ModInt b) { return down<mod>(a.x + b.x); }
10
       11
       friend ModInt operator*(ModInt a, ModInt b) { return (_int128)a.x * b.x % mod; }
12
       friend ModInt operator/ (ModInt a, ModInt b) { return a * ~b; }
13
       friend ModInt operator^ (ModInt a, long long b) {
14
15
           ModInt ans = 1;
16
           for (; b; b >>= 1, a *= a)
17
              if (b & 1) ans *= a;
18
           return ans;
19
       }
```

3 数学(数论)算法 3.2 计算几何

```
20
        friend ModInt operator~ (ModInt a) { return a ^ (mod - 2); }
21
        friend ModInt operator-(ModInt a) { return downmod> (mod - a.x); }
22
        friend ModInt &operator+= (ModInt &a, ModInt b) { return a = a + b; }
23
        friend ModInt & operator — (ModInt & a, ModInt b) { return a = a - b; }
        friend ModInt &operator*=(ModInt &a, ModInt b) { return a = a * b; }
24
25
        friend ModInt &operator/=(ModInt &a, ModInt b) { return a = a / b; }
        friend ModInt &operator^= (ModInt &a, long long b) { return a = a ^ b; }
26
27
        friend ModInt &operator++ (ModInt &a) { return a += 1; }
28
        friend ModInt operator++ (ModInt &a, int) {
29
            ModInt x = a;
30
            a += 1;
31
            return x;
32
        friend ModInt &operator— (ModInt &a) { return a -= 1; }
33
        friend ModInt operator— (ModInt &a, int) {
34
            ModInt x = a;
35
            a -= 1;
36
37
            return x;
38
39
        friend bool operator== (ModInt a, ModInt b) { return a.x == b.x; }
40
        friend bool operator!=(ModInt a, ModInt b) { return ! (a == b); }
41
   using mint = ModInt<>;
```

3.2 计算几何

Listing 19: computation-geometry.cpp

```
1
    template <class T>
    struct Point {
 2
 3
        Тх, у;
 4
        Point (void) = default;
        Point(T X, T Y) : x(X), y(Y) {}
 5
        inline bool operator==(const Point B) {
 6
 7
            return x == B.x && y == B.y;
 8
 9
        friend std::ostream &operator<<(std::ostream &out, Point P) {</pre>
            return out << "(" << P.x << ", _" << P.y << ")";
10
11
        friend std::istream &operator>>(std::istream &in, Point &P) {
12
13
            return in >> P.x >> P.y;
14
15
    } ;
16
    template <class T>
    struct Line {
17
18
        T A, B, C; // Ax + By + C = 0
19
        Line(void) = default;
        Line(T a, T b, T c) : A(a), B(b), C(c) {} // Ax + By + C = 0
20
21
        Line(T k, T b) : A(k), B(-1), C(b) {}
                                                  // y = kx + b
   };
22
23
    template <class T>
24
    inline int sign(T x) {
25
        return x == 0 ? 0 : (x < 0 ? -1 : 1);
26
27
    template <class T>
28
    inline bool parallel(Line<T> P, Line<T> Q) {
29
        return P.A * Q.B == P.B * Q.A;
30
    template <class T>
31
    inline Point<T> intersect(Line<T> P, Line<T> Q) {
32
        assert(!parallel(P, Q));
33
34
        return Point<T>{
             (P.C * Q.B - Q.C * P.B) / (Q.A * P.B - P.A * Q.B),
35
36
             (P.C * Q.A - Q.C * P.A) / (P.A * Q.B - Q.A * P.B)
        };
38
   }
```

```
template <class T>
40
    inline Line<T> getLine(Point<T> P, Point<T> Q) {
41
        assert(!(P == Q));
42
         if (P.x == Q.x) {
             return Line\langle T \rangle (-1, 0, P.x);
43
44
         } else if (P.y == Q.y) {
45
             return Line<T>(0, -1, P.y);
46
         } else {
47
             const T k = (P.x - Q.x) / (P.y - Q.y);
             return Line\langle T \rangle (k, P.y - k * P.x);
48
49
50
51
    template <class T>
52
    inline bool pointOnLine(Point<T> P, Line<T> L) {
         return L.A * P.x + L.B + P.y + L.C == 0;
53
54
    }
```

4 字符串算法

4.1 字符串哈希

Listing 20: hashed-string.cpp

```
1 using i64 = long long;
   using i128 = __int128;
   class HashedString {
 3
 4
   private:
        // change M and B if you want
 5
        static const i64 M = (1LL \ll 61) - 1;
 6
 7
        static const i64 B;
        // pow[i] contains B^i % M
 8
 9
        static std::vector<i64> pow;
10
        // p hash[i] is the hash of the first i characters of the given string
        std::vector<i64> r_hash, p_hash;
11
        i128 mul(i64 a, i64 b) { return (i128)a * b; }
12
        i64 mod_mul(i64 a, i64 b) { return mul(a, b) % M; }
13
14
15
   public:
        HashedString(const string &s) : r hash(s.size() + 1), p hash(s.size() + 1) {
16
17
            while (pow.size() < s.size()) { pow.push_back(mod_mul(pow.back(), B)); }</pre>
           p hash[0] = 0;
18
19
            r hash[0] = 0;
20
            for (size_t i = 0; i < s.size(); i++) {</pre>
21
                p_{ash[i + 1]} = (mul(p_{ash[i]}, B) + s[i]) % M; // 1-based
22
            i64 sz = s.size();
23
24
            for (int i = sz - 1, j = 0; i >= 0; i ---, j++) {
25
                r_hash[j + 1] = (mul(r_hash[j], B) + s[i]) % M;
26
27
        i64 getHash(int start, int end) { // 0 based
28
            i64 raw val = p hash[end + 1] - mod mul(p hash[start], pow[end - start + 1]);
29
30
            return (raw_val + M) % M;
31
        i64 getRHash(int start, int end) { // 0 based
32
            i64 raw_val = r_hash[end + 1] - mod_mul(r_hash[start], pow[end - start + 1]);
33
            return (raw_val + M) % M;
34
35
36
   };
    std::vector<i64> HashedString::pow = {1};
37
38 mt19937 rng((uint32 t)chrono::steady clock::now().time since epoch().count());
   const i64 HashedString::B = uniform int distribution<i64>(0, M − 1) (rng);
```

5 jiangly 代码库 (备用,侵权请提出 issue)

5.1 int128 输出流自定义

Listing 21: others/i128-stream.cpp

```
#include <iostream>
 2
   using i128 = int128;
 3
 4
    std::istream &operator>>(std::istream is, i128 &n) {
 5
 6
        std::string s;
 7
        is >> s;
 8
        for (auto c : s) {
 9
            n = n * 10 + (c - '0');
10
11
        return is;
12
13
    std::ostream &operator<<(std::ostream &os, i128 n) {
14
        std::string s;
15
        while (n) {
16
            s += '0' + n % 10;
17
18
            n /= 10;
19
20
        std::reverse(s.begin(), s.end());
21
        return os << s;
22
   }
```

5.2 常用数学运算库函数及 gcd 重载

Listing 22: others/clf.cpp

```
using i64 = long long;
   using i128 = int128;
   inline i64 ceilDiv(i64 n, i64 m) {
 3
        if (n >= 0) {
 4
            return (n + m - 1) / m;
 5
        } else {
 6
 7
            return n / m;
 8
 9
   }
10
   inline i64 floorDiv(i64 n, i64 m) {
11
        if (n >= 0) {
12
            return n / m;
13
        } else {
            return (n - m + 1) / m;
14
15
16
    template <class T>
17
18
    inline void chmax(T &a, T b) {
19
        if (a < b) a = b;
20
    template <class T>
21
    inline void chmin(T &a, T b) {
22
        if (!(a < b)) a = b;
23
24
   inline i128 gcd(i128 a, i128 b) {
25
        return b ? gcd(b, a % b) : a;
26
   }
27
```

5.3 强连通分量缩点 (SCC)

Listing 23: graph/scc.cpp

```
1
    #include <vector>
 2
 3
    struct SCC {
 4
        int n;
 5
        std::vector<std::vector<int>> adj;
        std::vector<int> stk;
 6
 7
        std::vector<int> dfn, low, bel;
 8
        int cur, cnt;
 9
10
        SCC() {}
11
        SCC(int n) {
12
             init(n);
13
14
        void init(int n) {
15
            this\rightarrown = n;
16
            adj.assign(n, {});
17
            dfn.assign(n, -1);
18
19
            low.resize(n);
20
            bel.assign(n, -1);
21
            stk.clear();
22
             cur = cnt = 0;
23
24
        void addEdge(int u, int v) {
25
26
            adj[u].push_back(v);
27
28
29
        void dfs(int x) {
            dfn[x] = low[x] = cur++;
30
31
            stk.push back(x);
32
33
             for (auto y : adj[x]) {
34
                 if (dfn[y] == -1) {
35
                     dfs(y);
                     low[x] = std::min(low[x], low[y]);
36
                 } else if (bel[y] == -1) {
37
                     low[x] = std::min(low[x], dfn[y]);
38
39
                 }
40
             }
41
42
             if (dfn[x] == low[x]) {
43
                 int y;
44
                 do {
45
                     y = stk.back();
46
                     bel[y] = cnt;
47
                     stk.pop_back();
                 } while (y != x);
48
                 cnt++;
49
             }
50
51
52
53
        std::vector<int> work() {
54
             for (int i = 0; i < n; i++) {</pre>
55
                 if (dfn[i] == -1) {
56
                     dfs(i);
57
                 }
58
             }
59
            return bel;
60
        }
61
   };
```

5.4 割边与割边缩点 (EBCC)

Listing 24: graph/ebcc.cpp

```
#include <set>
 2
    #include <vector>
 3
 4
    std::set<std::pair<int, int>> E;
 5
    struct EBCC {
 6
 7
        int n;
 8
        std::vector<std::vector<int>> adj;
 9
        std::vector<int> stk;
10
        std::vector<int> dfn, low, bel;
11
        int cur, cnt;
12
        EBCC() {}
13
        EBCC(int n) {
14
            init(n);
15
16
17
        void init(int n) {
18
            this\rightarrown = n;
19
            adj.assign(n, {});
20
21
            dfn.assign(n, -1);
22
            low.resize(n);
23
            bel.assign(n, -1);
24
            stk.clear();
25
            cur = cnt = 0;
26
27
28
        void addEdge(int u, int v) {
29
            adj[u].push back(v);
            adj[v].push back(u);
30
31
32
33
        void dfs(int x, int p) {
34
            dfn[x] = low[x] = cur++;
35
            stk.push_back(x);
36
            for (auto y : adj[x]) {
37
                 if (y == p) {
38
39
                     continue;
40
41
                 if (dfn[y] == -1) {
42
                     E.emplace(x, y);
43
                     dfs(y, x);
44
                     low[x] = std::min(low[x], low[y]);
                 } else if (bel[y] == -1 \&\& dfn[y] < dfn[x]) {
45
46
                     E.emplace(x, y);
47
                     low[x] = std::min(low[x], dfn[y]);
48
49
50
            if (dfn[x] == low[x]) {
51
52
                 int y;
53
                 do {
54
                     y = stk.back();
55
                     bel[y] = cnt;
56
                     stk.pop_back();
57
                 } while (y != x);
                 cnt++;
58
59
            }
60
        }
61
62
        std::vector<int> work() {
63
            dfs(0, -1);
64
            return bel;
65
```

```
66
67
        struct Graph {
68
             int n;
69
             std::vector<std::pair<int, int>> edges;
70
             std::vector<int> siz;
71
             std::vector<int> cnte;
72
        }:
73
        Graph compress() {
             Graph g;
74
75
             g.n = cnt;
76
             g.siz.resize(cnt);
77
             g.cnte.resize(cnt);
78
             for (int i = 0; i < n; i++) {</pre>
79
                 g.siz[bel[i]]++;
80
                 for (auto j : adj[i]) {
                     {f if} (bel[i] < bel[j]) {
81
                          g.edges.emplace_back(bel[i], bel[j]);
82
                      } else if (i < j) {
83
84
                          g.cnte[bel[i]]++;
85
86
                 }
87
88
             return g;
89
90
    } ;
```

5.5 二分图最大权匹配 (MaxAssignment, 基于 KM)

#include <queue>

1

Listing 25: graph/bigraph-weight-match.cpp

```
#include <vector>
2
3
4
    template <class T>
5
    struct MaxAssignment {
6
    public:
7
        T solve(int nx, int ny, std::vector<std::vector<T>> a) {
            assert(0 \le nx \& nx \le ny);
8
            assert(int(a.size()) == nx);
9
            for (int i = 0; i < nx; ++i) {</pre>
10
                assert(int(a[i].size()) == ny);
11
                for (auto x : a[i])
12
13
                     assert(x >= 0);
14
            }
15
16
            auto update = [&] (int x) {
17
                for (int y = 0; y < ny; ++y) {
18
                     if (lx[x] + ly[y] - a[x][y] < slack[y]) {
19
                         slack[y] = lx[x] + ly[y] - a[x][y];
                         slackx[y] = x;
20
21
                     }
22
                 }
23
            };
24
            costs.resize(nx + 1);
25
26
            costs[0] = 0;
27
            lx.assign(nx, std::numeric limits<T>::max());
28
            ly.assign(ny, 0);
29
            xy.assign(nx, -1);
            yx.assign(ny, -1);
30
            slackx.resize(ny);
31
            for (int cur = 0; cur < nx; ++cur) {</pre>
32
                std::queue<int> que;
33
34
                visx.assign(nx, false);
35
                visy.assign(ny, false);
36
                slack.assign(ny, std::numeric limits<T>::max());
```

```
37
                   p.assign(nx, -1);
38
39
                   for (int x = 0; x < nx; ++x) {
40
                        if (xy[x] == -1) {
41
                            que.push(x);
42
                            visx[x] = true;
                            update(x);
43
44
                        }
45
                   }
 46
 47
                   int ex, ey;
 48
                   bool found = false;
 49
                   while (!found) {
50
                        while (!que.empty() && !found) {
                            auto x = que.front();
51
52
                            que.pop();
                            for (int y = 0; y < ny; ++y) {</pre>
53
                                 \textbf{if} \ (\texttt{a}[\texttt{x}][\texttt{y}] == \texttt{lx}[\texttt{x}] \ + \texttt{ly}[\texttt{y}] \ \&\& \ !\texttt{visy}[\texttt{y}]) \ \{
54
                                      if (yx[y] == -1) {
55
56
                                          ex = x;
                                          ey = y;
57
 58
                                          found = true;
59
                                          break;
60
                                      }
61
                                      que.push(yx[y]);
62
                                      p[yx[y]] = x;
63
                                      visy[y] = visx[yx[y]] = true;
                                      update(yx[y]);
64
65
                                 }
66
                            }
67
 68
                        if (found)
69
                            break;
 70
                        T delta = std::numeric_limits<T>::max();
71
                        for (int y = 0; y < ny; ++y)
 72
                            if (!visy[y])
73
74
                                 delta = std::min(delta, slack[y]);
                        for (int x = 0; x < nx; ++x)
75
                            if (visx[x])
76
 77
                                 lx[x] = delta;
 78
                        for (int y = 0; y < ny; ++y) {</pre>
79
                            if (visy[y]) {
80
                                 ly[y] += delta;
81
                            } else {
82
                                 slack[y] -= delta;
83
                             }
84
                        for (int y = 0; y < ny; ++y) {</pre>
85
86
                            if (!visy[y] && slack[y] == 0) {
87
                                 if (yx[y] == -1) {
                                      ex = slackx[y];
88
89
                                      ey = y;
90
                                      found = true;
91
                                     break;
92
93
                                 que.push(yx[y]);
94
                                 p[yx[y]] = slackx[y];
95
                                 visy[y] = visx[yx[y]] = true;
96
                                 update(yx[y]);
97
                            }
                        }
98
99
                   }
100
101
                   costs[cur + 1] = costs[cur];
102
                   for (int x = ex, y = ey, ty; x != -1; x = p[x], y = ty) {
```

```
103
                     costs[cur + 1] += a[x][y];
104
                     if (xy[x] != -1)
                         costs[cur + 1] = a[x][xy[x]];
105
106
                     ty = xy[x];
107
                     xy[x] = y;
108
                     yx[y] = x;
109
110
111
             return costs[nx];
112
113
         std::vector<int> assignment() {
114
             return xy;
115
         std::pair<std::vector<T>, std::vector<T>> labels() {
116
117
             return std::make pair(lx, ly);
118
         std::vector<T> weights() {
119
             return costs:
120
121
122
123
    private:
         std::vector<T> lx, ly, slack, costs;
125
         std::vector<int> xy, yx, p, slackx;
126
         std::vector<bool> visx, visy;
127
    };
```

5.6 一般图最大匹配 (Graph, 带花树算法)

Listing 26: graph/general-match.cpp

```
#include <queue>
 1
    #include <vector>
 2
 3
 4
    struct Graph {
 5
        int n;
 6
        std::vector<std::vector<int>> e;
 7
        Graph(int n) : n(n), e(n) {}
 8
        void addEdge(int u, int v) {
            e[u].push_back(v);
 9
10
            e[v].push back(u);
11
        std::vector<int> findMatching() {
12
            std::vector\leqint\geq match(n, -1), vis(n), link(n), f(n), dep(n);
13
14
            // disjoint set union
15
16
            auto find = [&] (int u) {
17
                 while (f[u] != u)
18
                     u = f[u] = f[f[u]];
19
                 return u;
            };
20
21
22
            auto lca = [&] (int u, int v) {
23
                 u = find(u);
                 v = find(v);
24
25
                 while (u != v) {
26
                     if (dep[u] < dep[v])
27
                         std::swap(u, v);
28
                     u = find(link[match[u]]);
29
                 return u;
30
            } ;
31
32
            std::queue<int> que;
33
34
            auto blossom = [&] (int u, int v, int p) {
35
                 while (find(u) != p) {
36
                     link[u] = v;
```

```
37
                     v = match[u];
38
                     if (vis[v] == 0) {
39
                         vis[v] = 1;
40
                         que.push(v);
                      }
41
                     f[u] = f[v] = p;
42
                     u = link[v];
43
44
                 }
45
             };
46
47
             // find an augmenting path starting from u and augment (if exist)
48
             auto augment = [&] (int u) {
49
                 while (!que.empty())
50
                     que.pop();
51
                 std::iota(f.begin(), f.end(), 0);
52
53
                 // vis = 0 corresponds to inner vertices, vis = 1 corresponds to outer vertices
54
55
                 std::fill(vis.begin(), vis.end(), -1);
56
57
                 que.push(u);
58
                 vis[u] = 1;
59
                 dep[u] = 0;
60
61
                 while (!que.empty()) {
                     int u = que.front();
62
63
                     que.pop();
                     for (auto v : e[u]) {
64
65
                         if (vis[v] == -1) {
66
67
                              vis[v] = 0;
68
                              link[v] = u;
69
                              dep[v] = dep[u] + 1;
70
                              // found an augmenting path
71
                              if (match[v] == -1) {
72
                                  for (int x = v, y = u, temp; y != -1; x = temp, y = x == -1 ? -1 : link[x]) {
73
                                      temp = match[y];
74
                                      match[x] = y;
75
76
                                      match[y] = x;
                                  }
77
78
                                  return;
79
                              }
80
81
                              vis[match[v]] = 1;
82
                              dep[match[v]] = dep[u] + 2;
83
                              que.push(match[v]);
84
                          } else if (vis[v] == 1 && find(v) != find(u)) {
85
86
                              // found a blossom
87
                              int p = lca(u, v);
                              blossom(u, v, p);
88
89
                              blossom(v, u, p);
90
                          }
91
                     }
92
             };
93
94
             // find a maximal matching greedily (decrease constant)
95
             auto greedy = [&]() {
96
                 for (int u = 0; u < n; ++u) {
97
                     if (match[u] != -1)
98
99
                         continue;
100
                     for (auto v : e[u]) {
101
                         if (match[v] == -1) {
102
                             match[u] = v;
```

```
103
                               match[v] = u;
104
                               break;
105
                           }
106
                       }
107
                  }
108
              };
109
110
              greedy();
111
              for (int u = 0; u < n; ++u)
112
113
                  if (match[u] == -1)
114
                       augment (u);
115
116
              return match;
117
    };
118
```

5.7 2-SAT

Listing 27: graph/2-sat.cpp

```
1
    #include <vector>
 2
 3
    struct TwoSat {
 4
        int n;
 5
        std::vector<std::vector<int>> e;
 6
        std::vector<bool> ans;
 7
        TwoSat(int n) : n(n), e(2 * n), ans(n) {}
        void addClause(int u, bool f, int v, bool g) {
 8
 9
            e[2 * u + !f].push_back(2 * v + g);
            e[2 * v + !g].push back(2 * u + f);
10
11
        bool satisfiable() {
12
13
            std::vector<int> id(2 * n, -1), dfn(2 * n, -1), low(2 * n, -1);
14
            std::vector<int> stk;
15
            int now = 0, cnt = 0;
16
            std::function<void(int)> tarjan = [&] (int u) {
17
                stk.push_back(u);
                dfn[u] = low[u] = now++;
18
                for (auto v : e[u]) {
19
                     if (dfn[v] == -1) {
20
21
                         tarjan(v);
                         low[u] = std::min(low[u], low[v]);
22
23
                     } else if (id[v] == -1) {
24
                         low[u] = std::min(low[u], dfn[v]);
25
26
                if (dfn[u] == low[u]) {
27
                     int v;
28
                     do {
29
30
                         v = stk.back();
31
                         stk.pop back();
32
                         id[v] = cnt;
                     } while (v != u);
33
34
                     ++cnt;
35
                 }
36
            for (int i = 0; i < 2 * n; ++i)</pre>
37
                if (dfn[i] == -1) tarjan(i);
38
            for (int i = 0; i < n; ++i) {</pre>
39
                if (id[2 * i] == id[2 * i + 1]) return false;
40
                ans[i] = id[2 * i] > id[2 * i + 1];
41
42
43
            return true;
44
        }
45
        std::vector<bool> answer() { return ans; }
```

46 };

5.8 最大流

Listing 28: graph/max-flow.cpp

```
constexpr int inf = 1E9;
 1
    \textbf{template} < \textbf{class} \  \, \mathbb{T} >
 2
 3
    struct MaxFlow {
        struct _Edge {
 4
 5
             int to;
 6
             T cap;
 7
             _Edge(int to, T cap) : to(to), cap(cap) {}
 8
 9
10
        int n;
         std::vector<_Edge> e;
11
         std::vector<std::vector<int>> g;
12
         std::vector<int> cur, h;
13
14
        MaxFlow() {}
15
        MaxFlow(int n) {
16
17
             init(n);
18
19
20
        void init(int n) {
21
             this\rightarrown = n;
22
             e.clear();
23
             g.assign(n, {});
24
             cur.resize(n);
25
             h.resize(n);
26
         }
27
28
        bool bfs(int s, int t) {
29
             h.assign(n, -1);
30
             std::queue<int> que;
31
             h[s] = 0;
             que.push(s);
32
             while (!que.empty()) {
33
                 const int u = que.front();
34
35
                 que.pop();
36
                 for (int i : g[u]) {
37
                      auto [v, c] = e[i];
                      if (c > 0 \&\& h[v] == -1) {
38
39
                          h[v] = h[u] + 1;
40
                          if (v == t) {
41
                              return true;
42
                          }
43
                          que.push(v);
                      }
44
45
                 }
46
47
             return false;
48
         }
49
50
         T dfs(int u, int t, T f) {
51
             if (u == t) {
52
                 return f;
53
             auto r = f;
54
             for (int &i = cur[u]; i < int(g[u].size()); ++i) {</pre>
55
                 const int j = g[u][i];
56
57
                 auto [v, c] = e[j];
58
                 if (c > 0 \&\& h[v] == h[u] + 1) {
59
                      auto a = dfs(v, t, std::min(r, c));
60
                      e[j].cap -= a;
```

```
61
                      e[j ^ 1].cap += a;
62
                      r -= a;
                      if (r == 0) {
63
64
                          return f;
65
66
67
68
             return f - r;
69
 70
         void addEdge(int u, int v, T c) {
 71
              g[u].push back(e.size());
 72
              e.emplace back(v, c);
 73
              g[v].push back(e.size());
              e.emplace_back(u, 0);
74
 75
         T flow(int s, int t) {
76
77
             T ans = 0;
78
             while (bfs(s, t)) {
 79
                  cur.assign(n, 0);
                  ans += dfs(s, t, std::numeric_limits<T>::max());
80
81
82
              return ans;
83
         }
84
         std::vector<bool> minCut() {
85
              std::vector<br/>bool> c(n);
86
              for (int i = 0; i < n; i++) {</pre>
87
                  c[i] = (h[i] != -1);
88
89
90
             return c;
91
         }
 92
93
         struct Edge {
94
             int from;
95
              int to;
              T cap;
96
97
              T flow;
98
         };
         std::vector<Edge> edges() {
99
100
              std::vector<Edge> a;
              for (int i = 0; i < e.size(); i += 2) {</pre>
101
                  Edge x;
102
103
                  x.from = e[i + 1].to;
104
                  x.to = e[i].to;
105
                  x.cap = e[i].cap + e[i + 1].cap;
106
                  x.flow = e[i + 1].cap;
107
                  a.push back(x);
108
109
             return a;
110
         }
111
     };
```

5.9 最小费用可行流(或最大流)

Listing 29: graph/max-cost-flow-graph.cpp

```
1
    struct MCFGraph {
2
        struct Edge {
            int v, c, f;
3
            Edge(int v, int c, int f) : v(v), c(c), f(f) {}
4
        };
5
6
        const int n;
7
        std::vector Edge> e;
8
        std::vector<std::vector<int>> g;
9
        std::vector<i64> h, dis;
10
        std::vector<int> pre;
```

```
bool dijkstra(int s, int t) {
11
12
            dis.assign(n, std::numeric limits<i64>::max());
13
            pre.assign(n, -1);
14
            std::priority_queue<std::pair<i64, int>, std::vector<std::pair<i64, int>, std::greater<std::pair<
               i64, int>>> que;
            dis[s] = 0;
15
16
            que.emplace(0, s);
17
            while (!que.empty()) {
18
                 i64 d = que.top().first;
19
                int u = que.top().second;
20
                que.pop();
21
                if (dis[u] < d) continue;</pre>
22
                for (int i : g[u]) {
23
                     int v = e[i].v;
                     int c = e[i].c;
24
                     int f = e[i].f;
25
                     if (c > 0 \&\& dis[v] > d + h[u] - h[v] + f) {
26
                         dis[v] = d + h[u] - h[v] + f;
27
28
                         pre[v] = i;
29
                         que.emplace(dis[v], v);
30
                     }
31
                 }
32
33
            return dis[t] != std::numeric limits<i64>::max();
34
35
        MCFGraph(int n) : n(n), g(n) {}
        void addEdge(int u, int v, int c, int f) { // 可行流
36
            if (f < 0) {
37
38
                g[u].push back(e.size());
39
                e.emplace back(v, 0, f);
40
                g[v].push back(e.size());
41
                e.emplace back(u, c, -f);
42
            } else {
43
                g[u].push back(e.size());
44
                e.emplace_back(v, c, f);
                g[v].push back(e.size());
45
46
                e.emplace_back(u, 0, -f);
            }
47
48
        }
49
        // void addEdge(int u, int v, int c, int f) { // 最大流
50
        //
               g[u].push back(e.size());
51
        //
               e.emplace back(v, c, f);
52
               g[v].push back(e.size());
        //
53
        //
               e.emplace back(u, 0, -f);
        // }
54
55
        std::pair<int, i64> flow(int s, int t) {
56
            int flow = 0;
            i64 cost = 0;
57
            h.assign(n, 0);
58
59
            while (dijkstra(s, t)) {
60
                for (int i = 0; i < n; ++i) h[i] += dis[i];</pre>
61
                int aug = std::numeric limits<int>::max();
                for (int i = t; i != s; i = e[pre[i] ^ 1].v) aug = std::min(aug, e[pre[i]].c);
62
63
                for (int i = t; i != s; i = e[pre[i] ^ 1].v) {
64
                     e[pre[i]].c = aug;
65
                     e[pre[i] ^ 1].c += aug;
66
67
                flow += aug;
68
                cost += i64(aug) * h[t];
69
70
            return std::make_pair(flow, cost);
71
        }
    };
72
```

5.10 树链剖分

Listing 30: graph/hld.cpp

```
struct HLD {
 1
 2
        int n;
 3
        std::vector<int> siz, top, dep, parent, in, out, seq;
 4
        std::vector<std::vector<int>> adj;
 5
        int cur;
 6
        HLD() {}
 7
 8
        HLD(int n) {
 9
            init(n);
10
11
        void init(int n) {
12
            this\rightarrown = n;
13
            siz.resize(n);
14
            top.resize(n);
15
            dep.resize(n);
            parent.resize(n);
16
17
            in.resize(n);
18
            out.resize(n);
19
            seq.resize(n);
20
            cur = 0;
21
            adj.assign(n, {});
22
23
        void addEdge(int u, int v) {
24
            adj[u].push_back(v);
            adj[v].push_back(u);
25
26
27
        void work(int root = 0) {
28
            top[root] = root;
29
            dep[root] = 0;
30
            parent[root] = -1;
31
            dfs1(root);
32
            dfs2(root);
33
34
        void dfs1(int u) {
35
            if (parent[u] !=-1) {
                 adj[u].erase(std::find(adj[u].begin(), adj[u].end(), parent[u]));
36
37
            }
38
            siz[u] = 1;
39
40
            for (auto &v : adj[u]) {
41
                 parent[v] = u;
42
                 dep[v] = dep[u] + 1;
43
                 dfs1(v);
44
                 siz[u] += siz[v];
                 if (siz[v] > siz[adj[u][0]]) {
45
46
                     std::swap(v, adj[u][0]);
47
            }
48
49
        void dfs2(int u) {
50
51
            in[u] = cur++;
            seq[in[u]] = u;
52
53
            for (auto v : adj[u]) {
54
                 top[v] = v == adj[u][0] ? top[u] : v;
55
                 dfs2(v);
56
            }
            out[u] = cur;
57
58
        int lca(int u, int v) {
59
            while (top[u] != top[v]) {
60
61
                 if (dep[top[u]] > dep[top[v]]) {
62
                     u = parent[top[u]];
63
                 } else {
64
                     v = parent[top[v]];
65
```

```
66
67
               return dep[u] < dep[v] ? u : v;
68
69
70
          int dist(int u, int v) {
               return dep[u] + dep[v] - 2 * dep[lca(u, v)];
71
72
 73
74
          int jump(int u, int k) {
 75
               if (dep[u] < k) {
76
                   return -1;
77
 78
               int d = dep[u] - k;
79
80
               while (dep[top[u]] > d) {
81
82
                   u = parent[top[u]];
83
84
               return seq[in[u] - dep[u] + d];
85
86
87
88
          bool isAncestor(int u, int v) {
               \textbf{return} \ \text{in}[\textbf{u}] \ \Leftarrow \ \text{in}[\textbf{v}] \ \&\& \ \text{in}[\textbf{v}] \ \leqslant \ \text{out}[\textbf{u}] \textbf{;}
89
90
91
          int rootedParent(int u, int v) {
92
               std::swap(u, v);
93
94
               if (u == v) {
95
                   return u;
96
97
               if (!isAncestor(u, v)) {
98
                   return parent[u];
99
               auto it = std::upper_bound(adj[u].begin(), adj[u].end(), v, [&] (int x, int y) {
100
                   return in[x] < in[y];</pre>
101
               ) - 1;
102
               return *it;
103
104
105
          int rootedSize(int u, int v) {
106
               if (u == v) {
107
108
                   return n;
109
110
               if (!isAncestor(v, u)) {
111
                   return siz[v];
112
               }
               return n - siz[rootedParent(u, v)];
113
          }
114
115
116
          int rootedLca(int a, int b, int c) {
               return lca(a, b) ^ lca(b, c) ^ lca(c, a);
117
118
119
     };
```

5.11 快速幂

Listing 31: math/fast-pow.cpp

```
1 int power(int a, i64 b, int p) {
2    int res = 1;
3    for (; b; b /= 2, a = 1LL * a * a % p) {
4        if (b % 2) {
5          res = 1LL * res * a % p;
6        }
7    }
```

```
8     return res;
9 }
```

5.12 欧拉筛

Listing 32: math/euler-sieve.cpp

```
std::vector<int> minp, primes;
 1
 2
 3
    void sieve(int n) {
 4
        minp.assign(n + 1, 0);
 5
        primes.clear();
 6
 7
        for (int i = 2; i <= n; i++) {</pre>
 8
            if (minp[i] == 0) {
 9
                minp[i] = i;
                 primes.push_back(i);
10
11
12
            for (auto p : primes) {
13
                 if (i * p > n) {
14
                    break;
15
16
                 }
17
                minp[i * p] = p;
18
                 if (p == minp[i]) {
19
                    break;
20
21
            }
22
        }
23
    }
```

5.13 单点欧拉函数

Listing 33: math/phi.cpp

```
int phi(int n) {
 1
 2
        int res = n;
 3
        for (int i = 2; i * i <= n; i++) {</pre>
 4
            if (n % i == 0) {
 5
                 while (n % i == 0) {
 6
                     n /= i;
 7
                 res = res / i * (i - 1);
 8
            }
 9
10
        }
        if (n > 1) {
11
            res = res / n * (n-1);
12
13
14
        return res;
15
```

5.14 exgcd

Listing 34: math/exgcd.cpp

```
int exgcd(int a, int b, int &x, int &y) {
1
       if (!b) {
2
           x = 1, y = 0;
3
4
           return a;
5
6
       int g = exgcd(b, a % b, y, x);
7
       y = a / b * x;
8
       return g;
9
   }
```

5.15 组合数

Listing 35: math/comb.cpp

```
struct Comb {
 1
 2
        int n;
 3
        std::vector<Z> fac;
        std::vector<Z> invfac;
 4
        std::vector<Z> inv;
 5
 6
 7
        Comb() : n{0}, fac{1}, invfac{1}, inv{0} {}
 8
        Comb(int n) : Comb() {
 9
            init(n);
10
        }
11
        void init(int m) {
12
            m = std::min(m, Z::getMod() - 1);
13
            if (m <= n) return;</pre>
14
            _fac.resize(m + 1);
15
            _invfac.resize(m + 1);
16
            _inv.resize(m + 1);
17
18
19
             for (int i = n + 1; i <= m; i++) {</pre>
20
                 fac[i] = fac[i-1] * i;
21
             _{invfac[m]} = _{fac[m].inv();}
22
            for (int i = m; i > n; i---) {
23
                 _{invfac[i-1] = _{invfac[i]} * i;}
24
                 _{inv[i]} = _{invfac[i]} * _{fac[i-1]};
25
26
27
            n = m;
28
29
30
        Z fac(int m) {
31
            if (m > n) init(2 * m);
32
            return _fac[m];
33
        }
        Z invfac(int m) {
34
35
            if (m > n) init(2 * m);
36
            return _invfac[m];
37
        }
38
        Z inv(int m) {
39
            if (m > n) init(2 * m);
40
            return _inv[m];
41
        Z binom(int n, int m) {
42
            if (n < m | | m < 0) return 0;</pre>
43
            return fac(n) * invfac(m) * invfac(n - m);
44
45
        }
    } comb;
46
```

5.16 树状数组

Listing 36: ds/fenwick.cpp

```
template <typename T>
 2
    struct Fenwick {
 3
        int n;
        std::vector<T> a;
 4
 5
        Fenwick(int n = 0) {
 6
 7
            init(n_);
 8
10
        void init(int n_) {
11
            n = n;
```

```
12
              a.assign(n, T{});
13
         }
14
         \textbf{void} \ \text{add}(\textbf{int} \ \textbf{x, const} \ \textbf{T} \ \& \textbf{v}) \ \ \{
15
              for (int i = x + 1; i <= n; i += i & -i) {</pre>
16
                   a[i-1] = a[i-1] + v;
17
18
19
         }
20
21
         T sum(int x) {
22
              T ans{};
              for (int i = x; i > 0; i -= i & -i) {
23
24
                   ans = ans + a[i-1];
25
26
              return ans;
27
         }
28
29
         T rangeSum(int 1, int r) {
              return sum(r) - sum(l);
30
31
32
33
         int select(const T &k) {
34
              int x = 0;
              T cur{};
35
              for (int i = 1 << std::__lg(n); i; i /= 2) {</pre>
36
                   if (x + i \le n \&\& cur + a[x + i - 1] \le k) {
37
                       x += i;
38
                       cur = cur + a[x - 1];
39
40
41
42
              return x;
43
         }
44
    } ;
```

5.17 Splay

Listing 37: ds/splay.cpp

```
struct Node {
 1
        Node *1 = nullptr;
 2
        Node *r = nullptr;
 3
        int cnt = 0;
 4
 5
        i64 sum = 0;
 6
    };
 7
 8
    Node *add(Node *t, int 1, int r, int p, int v) {
 9
        Node *x = new Node;
        if (t) {
10
             *x = *t;
11
12
13
        x->cnt += 1;
14
        x->sum += v;
15
        if (r - 1 == 1) {
             return x;
16
17
18
        int m = (1 + r) / 2;
19
        if (p < m) {
20
             x\rightarrow 1 = add(x\rightarrow 1, 1, m, p, v);
21
         } else {
             x\rightarrow r = add(x\rightarrow r, m, r, p, v);
22
23
24
        return x;
25
    }
26
    int find(Node *tl, Node *tr, int l, int r, int x) {
        if (r <= x) {
```

```
29
               return -1;
30
          if (1 >= x) {
31
               int cnt = (tr ? tr\rightarrow cnt : 0) - (tl ? tl\rightarrow cnt : 0);
32
               if (cnt == 0) {
33
34
                    return -1;
35
               if (r - 1 == 1) {
36
37
                    return 1;
38
39
40
          int m = (1 + r) / 2;
41
          int res = find(tl ? tl->l : tl, tr ? tr->l : tr, l, m, x);
          if (res == -1) {
42
               res = find(tl ? tl\rightarrowr : tl, tr ? tr\rightarrowr : tr, m, r, x);
43
44
45
          return res;
46
     }
47
     std::pair<int, i64> get(Node *t, int 1, int r, int x, int y) {
48
          if (1 >= y || r <= x || !t) {</pre>
49
50
               return {0, OLL};
51
          if (1 >= x && r <= y) {</pre>
52
53
               return {t->cnt, t->sum};
54
          int m = (1 + r) / 2;
55
          auto [cl, sl] = get(t->l, l, m, x, y);
56
57
          auto [cr, sr] = get(t->r, m, r, x, y);
58
          return {cl + cr, sl + sr};
59
     }
60
61
     struct Tree {
62
          int add = 0;
          int val = 0;
63
          int id = 0;
64
          Tree *ch[2] = {};
65
          Tree *p = nullptr;
66
67
    } ;
68
    int pos(Tree *t) {
69
70
          return t \rightarrow p \rightarrow ch[1] == t;
71
72
73
     void add(Tree *t, int v) {
74
          t—>val += v;
75
          t\rightarrow add += v;
76
     }
77
78
     void push(Tree *t) {
79
          if (t->ch[0]) {
               add(t\rightarrow ch[0], t\rightarrow add);
80
81
82
          if (t->ch[1]) {
83
               add(t\rightarrow ch[1], t\rightarrow add);
84
          t\rightarrow add = 0;
85
86
     }
87
     void rotate(Tree *t) {
88
          Tree *q = t \rightarrow p;
89
          int x = !pos(t);
90
91
          q \rightarrow ch[!x] = t \rightarrow ch[x];
92
          if (t\rightarrow ch[x]) t\rightarrow ch[x]\rightarrow p = q;
93
          t\rightarrow p = q\rightarrow p;
94
          if (q\rightarrow p) q\rightarrow p\rightarrow ch[pos(q)] = t;
```

```
95
          t\rightarrow ch[x] = q;
96
          q \rightarrow p = t;
97
     }
98
99
     void splay(Tree *t) {
100
          std::vector<Tree *> s;
101
          for (Tree *i = t; i\rightarrow p; i = i\rightarrow p) s.push_back(i\rightarrow p);
102
          while (!s.empty()) {
103
               push(s.back());
104
               s.pop back();
105
106
          push(t);
107
          while (t->p) {
               108
                   \textbf{if} \ (pos(t) == pos(t \rightarrow p)) \ rotate(t \rightarrow p);
109
                   else rotate(t);
110
111
112
               rotate(t);
113
          }
     }
114
115
116
     void insert(Tree *&t, Tree *x, Tree *p = nullptr) {
117
          if (!t) {
118
               t = x;
               x \rightarrow p = p;
119
               return;
120
121
          }
122
123
          push(t);
124
          if (x->val < t->val) {
125
              insert(t\rightarrow ch[0], x, t);
126
          } else {
127
               insert(t\rightarrow ch[1], x, t);
128
129
     }
130
     void dfs(Tree *t) {
131
          if (!t) {
132
               return;
133
134
          }
          push(t);
135
          dfs(t->ch[0]);
136
137
          std::cerr << t->val << "_";
138
          dfs(t->ch[1]);
139
     }
140
     std::pair<Tree *, Tree *> split(Tree *t, int x) {
141
142
          if (!t) {
               return {t, t};
143
144
          }
          Tree *v = nullptr;
145
146
          Tree *j = t;
147
          for (Tree *i = t; i;) {
148
              push(i);
149
               j = i;
               if (i->val >= x) {
150
                   v = i;
151
                   i = i \rightarrow ch[0];
152
               } else {
153
                   i = i \rightarrow ch[1];
154
155
               }
          }
156
157
158
          splay(j);
159
          if (!v) {
160
              return {j, nullptr};
```

```
161
162
163
          splay(v);
164
          Tree *u = v \rightarrow ch[0];
165
          if (u) {
166
167
               v\rightarrow ch[0] = u\rightarrow p = nullptr;
168
          }
          // std::cerr << "split " << x << "\n";
169
170
          // dfs(u);
          // std::cerr << "\n";
171
          // dfs(v);
172
          // std::cerr << "\n";
173
          return {u, v};
174
175
176
177
     Tree *merge(Tree *1, Tree *r) {
178
          if (!1) {
179
               return r;
180
          if (!r) {
181
182
               return 1;
183
          Tree *i = 1;
184
          while (i->ch[1]) {
185
               i = i \rightarrow ch[1];
186
187
          splay(i);
188
189
          i \rightarrow ch[1] = r;
190
          r\rightarrow p = i;
191
          return i;
192
```

5.18 取模类,按需写

Listing 38: ds/mod.cpp

```
\textbf{template} < \textbf{class} \  \, \mathbb{T} \!\! >
 1
 2
    constexpr T power(T a, i64 b) {
         T res = 1;
 3
         for (; b; b /= 2, a *= a) {
 4
             if (b % 2) {
 5
 6
                  res *= a;
 7
 8
 9
         return res;
10
    }
11
    constexpr i64 mul(i64 a, i64 b, i64 p) {
12
13
        i64 \text{ res} = a * b - i64(1.L * a * b / p) * p;
         res %= p;
14
15
         if (res < 0) {
16
             res += p;
17
18
         return res;
19
20
    template <i64 P>
    struct MLong {
21
         i64 x;
22
         constexpr MLong() : x{} {}
23
         constexpr MLong(i64 x) : x{norm(x % getMod())} {}
24
25
26
         static i64 Mod;
27
         constexpr static i64 getMod() {
28
             if (P > 0) {
                  return P;
```

```
30
             } else {
31
                return Mod;
32
33
        constexpr static void setMod(i64 Mod ) {
34
35
            Mod = Mod ;
36
        constexpr i64 norm(i64 x) const {
37
38
            if (x < 0) {
39
                x += getMod();
40
41
            if (x \ge getMod()) {
42
                x = getMod();
43
44
            return x;
45
        constexpr i64 val() const {
46
47
            return x;
48
        explicit constexpr operator i64() const {
49
50
            return x;
51
52
        constexpr MLong operator-() const {
53
            MLong res;
            res.x = norm(getMod() - x);
54
            return res;
55
56
        }
        constexpr MLong inv() const {
57
58
            assert(x != 0);
59
            return power(*this, getMod() -2);
60
61
        constexpr MLong & operator* = (MLong rhs) & {
62
            x = mul(x, rhs.x, getMod());
63
            return *this;
64
        constexpr MLong & operator += (MLong rhs) & {
65
            x = norm(x + rhs.x);
66
            return *this;
67
68
69
        constexpr MLong & operator = (MLong rhs) & {
            x = norm(x - rhs.x);
70
            return *this;
71
72
73
        constexpr MLong & operator /= (MLong rhs) & {
74
            return *this *= rhs.inv();
75
76
        friend constexpr MLong operator* (MLong lhs, MLong rhs) {
77
            MLong res = lhs;
            res *= rhs;
78
79
            return res;
80
        friend constexpr MLong operator+ (MLong lhs, MLong rhs) {
81
82
            MLong res = lhs;
83
            res += rhs;
84
            return res;
85
        friend constexpr MLong operator-(MLong lhs, MLong rhs) {
86
87
            MLong res = lhs;
            res -= rhs;
88
89
            return res;
90
        friend constexpr MLong operator/ (MLong lhs, MLong rhs) {
91
92
            MLong res = lhs;
93
            res /= rhs;
94
            return res;
95
```

```
96
         friend constexpr std::istream &operator>>(std::istream &is, MLong &a) {
97
             i64 v;
98
             is >> v;
99
             a = MLong(v);
100
             return is;
101
         friend constexpr std::ostream &operator<<(std::ostream &os, const MLong &a) {
102
103
             return os << a.val();</pre>
104
         friend constexpr bool operator (MLong lhs, MLong rhs) {
105
106
             return lhs.val() == rhs.val();
107
108
         friend constexpr bool operator!=(MLong lhs, MLong rhs) {
109
             return lhs.val() != rhs.val();
110
111
     };
112
     template <>
113
     i64 MLong<0LI>::Mod = i64(1E18) + 9;
114
115
116 template <int P>
117
    struct MInt {
118
         int x;
119
         constexpr MInt() : x{} {}
         constexpr MInt(i64 x) : x{norm(x % getMod())} {}
120
121
         static int Mod;
122
         constexpr static int getMod() {
123
124
             if (P > 0) {
125
                 return P;
126
             } else {
127
                 return Mod;
128
129
130
         constexpr static void setMod(int Mod_) {
             Mod = Mod ;
131
132
         constexpr int norm(int x) const {
133
             if (x < 0) {
134
                 x += getMod();
135
136
             if (x \ge getMod()) {
137
                 x = getMod();
138
139
             }
140
             return x;
141
         }
142
         constexpr int val() const {
             return x;
143
144
         explicit constexpr operator int() const {
145
146
             return x;
147
         constexpr MInt operator-() const {
148
149
150
             res.x = norm(getMod() - x);
151
             return res;
152
         constexpr MInt inv() const {
153
             assert(x != 0);
154
             return power(*this, getMod() -2);
155
156
         constexpr MInt & operator* = (MInt rhs) & {
157
             x = 1LL * x * rhs.x % getMod();
158
159
             return *this;
160
         }
161
         constexpr MInt & operator += (MInt rhs) & {
```

```
162
             x = norm(x + rhs.x);
163
             return *this;
164
165
         constexpr MInt & operator = (MInt rhs) & {
             x = norm(x - rhs.x);
166
             return *this;
167
168
         }
169
         constexpr MInt & operator /= (MInt rhs) & {
             return *this *= rhs.inv();
170
171
172
         friend constexpr MInt operator* (MInt lhs, MInt rhs) {
173
             MInt res = lhs;
174
             res *= rhs;
175
             return res;
176
         friend constexpr MInt operator+ (MInt lhs, MInt rhs) {
177
             MInt res = lhs;
178
             res += rhs;
179
180
             return res;
181
         friend constexpr MInt operator-(MInt lhs, MInt rhs) {
182
183
             MInt res = lhs;
184
             res -= rhs;
185
             return res;
186
         friend constexpr MInt operator/ (MInt lhs, MInt rhs) {
187
188
             MInt res = lhs;
             res /= rhs;
189
190
             return res;
191
192
         friend constexpr std::istream &operator>>(std::istream &is, MInt &a) {
193
194
             is >> v;
195
             a = MInt(v);
196
             return is;
197
         friend constexpr std::ostream &operator<< (std::ostream &os, const MInt &a) {</pre>
198
             return os << a.val();</pre>
199
200
201
         friend constexpr bool operator == (MInt lhs, MInt rhs) {
             return lhs.val() == rhs.val();
202
203
204
         friend constexpr bool operator!=(MInt lhs, MInt rhs) {
205
             return lhs.val() != rhs.val();
206
207
     } ;
208
209
     template <>
     int MInt<0>::Mod = 998244353;
210
211
212
     template <int V, int P>
213
     constexpr MInt<P> CInv = MInt<P> (V) .inv();
214
215
     constexpr int P = 1000000007;
216
    using Z = MInt<P>;
```

5.19 马拉车

Listing 39: string/manacher.cpp

```
1 std::vector<int> manacher(std::vector<int> s) {
2    std::vector<int> t{0};
3    for (auto c : s) {
4         t.push_back(c);
5         t.push_back(0);
6    }
```

```
7
        int n = t.size();
 8
        std::vector<int> r(n);
 9
        for (int i = 0, j = 0; i < n; i++) {</pre>
             if (2 * j - i >= 0 \&\& j + r[j] > i) {
10
                 r[i] = std::min(r[2 * j - i], j + r[j] - i);
11
12
             while (i - r[i] \ge 0 \&\& i + r[i] < n \&\& t[i - r[i]] == t[i + r[i]]) {
13
14
                 r[i] += 1;
15
             if (i + r[i] > j + r[j]) {
16
17
18
19
20
        return r;
21
    }
```

5.20 Z 函数

Listing 40: string/z-func.cpp

```
1
    std::vector<int> zFunction(std::string s) {
2
        int n = s.size();
3
        std::vector<int> z(n + 1);
4
        z[0] = n;
        for (int i = 1, j = 1; i < n; i++) {</pre>
5
            z[i] = std: max(0, std: min(j + z[j] - i, z[i - j]));
6
            while (i + z[i] < n \&\& s[z[i]] == s[i + z[i]])  {
7
8
                 z[i]++;
9
            if (i + z[i] > j + z[j]) {
10
11
                 j = i;
12
13
14
        return z;
15
    }
```

5.21 SA 后缀数组

Listing 41: string/suffix-array.cpp

```
struct SuffixArray {
 1
        int n;
 3
        std::vector<int> sa, rk, lc;
        SuffixArray(const std::string &s) {
 5
            n = s.length();
 6
            sa.resize(n);
 7
            lc.resize(n-1);
 8
            rk.resize(n);
 9
            std::iota(sa.begin(), sa.end(), 0);
10
            std::sort(sa.begin(), sa.end(), [&](int a, int b) { return s[a] < s[b]; });
            rk[sa[0]] = 0;
11
12
            for (int i = 1; i < n; ++i)</pre>
                rk[sa[i]] = rk[sa[i-1]] + (s[sa[i]] != s[sa[i-1]]);
13
14
            int k = 1;
15
            std::vector<int> tmp, cnt(n);
16
            tmp.reserve(n);
17
            while (rk[sa[n-1]] < n-1) {
18
                tmp.clear();
                for (int i = 0; i < k; ++i)
19
                    tmp.push back(n - k + i);
20
                for (auto i : sa)
21
                    if (i >= k)
22
23
                         tmp.push back(i - k);
24
                std::fill(cnt.begin(), cnt.end(), 0);
25
                for (int i = 0; i < n; ++i)</pre>
```

```
26
                                                                                           ++cnt[rk[i]];
27
                                                                         for (int i = 1; i < n; ++i)</pre>
28
                                                                                           cnt[i] += cnt[i-1];
29
                                                                         for (int i = n - 1; i >= 0; —i)
30
                                                                                           sa[-cnt[rk[tmp[i]]]] = tmp[i];
31
                                                                         std::swap(rk, tmp);
32
                                                                         rk[sa[0]] = 0;
33
                                                                         for (int i = 1; i < n; ++i)</pre>
34
                                                                                           rk[sa[i]] = rk[sa[i-1]] + (tmp[sa[i-1]] < tmp[sa[i]] || sa[i-1] + k == n || tmp[sa[i]] + k = n || 
                                                                                                     -1] + k] < tmp[sa[i] + k]);
35
                                                                          k *= 2;
36
37
                                                       for (int i = 0, j = 0; i < n; ++i) {</pre>
38
                                                                         if (rk[i] == 0) {
39
                                                                                            j = 0;
                                                                          } else {
40
                                                                                           for (j -= j > 0; i + j < n \& sa[rk[i] - 1] + j < n \& s[i + j] == s[sa[rk[i] - 1] + j];)
41
                                                                                                             ++j;
42
43
                                                                                           lc[rk[i] - 1] = j;
44
                                                                         }
                                                       }
45
46
47
                  };
```

6 Watashi 代码库 (备用)

6.1 $O(n \log n) - O(1)$ **RMQ**

Listing 42: rmq.cpp

```
#include <algorithm> // copy
 2
    #include <climits> // CHAR_BIT
 3
 4
    using namespace std;
    \textbf{template} < \textbf{typename} \  \, \mathbb{T} >
 6
 7
    struct RMO {
 8
        int n;
 9
        vector<T> e;
10
        vector<vector<int>> rmq;
11
12
        static const int INT_BIT = sizeof(4) * CHAR_BIT;
13
         static inline int LG2(int i) { return INT_BIT - 1 - _ builtin_clz(i); }
14
         static inline int BIN(int i) { return 1 << i; }
15
        int cmp(int 1, int r) const {
16
17
             return e[1] <= e[r] ? 1 : r;
18
19
         void init(int n, const T e[]) {
20
21
             this\rightarrown = n;
             vector<T>(e, e + n).swap(this->e);
22
23
24
             int m = 1;
             while (BIN(m) \leq n) {
25
26
                 ++m;
27
             vector<vector<int>>(m, vector<int>(n)).swap(rmq);
28
29
             for (int i = 0; i < n; ++i) {</pre>
30
                 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
            int b = LG2(r - 1);
41
            return cmp(rmq[b][1], rmq[b][r - (1 \ll b)]);
42
43
44
45
        T value(int 1, int r) const {
46
            return e[index(l, r)];
47
48
    };
```

6.2 $O(n \log n) - O(\log n)$ **LCA**

Listing 43: lca.cpp

```
#include <algorithm>
    #include <cstdio>
    #include <vector>
 3
    using namespace std;
 6
 7
    const int MAXM = 16;
 8
    const int MAXN = 1 << MAXM;</pre>
 9
10
    // LCA
    struct LCA {
11
12
        vector<int> e[MAXN];
13
        int d[MAXN], p[MAXN] [MAXM];
14
15
        void dfs_(int v, int f) {
16
             p[v][0] = f;
17
             for (int i = 1; i < MAXM; ++i) {</pre>
18
                 p[v][i] = p[p[v][i-1]][i-1];
19
             for (int i = 0; i < (int)e[v].size(); ++i) {</pre>
20
                 int w = e[v][i];
21
                 if (w != f) {
22
                      d[w] = d[v] + 1;
23
24
                      dfs_(w, v);
25
                 }
26
             }
27
28
         int up (int v, int m) {
29
             for (int i = 0; i < MAXM; ++i) {</pre>
30
                 \textbf{if} \ (\texttt{m \& (1 << i))} \ \{
31
                      v = p[v][i];
32
33
34
35
             return v;
36
         }
37
38
         int lca(int a, int b) {
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
             } else {
45
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
                 }
52
                 return p[a] [0];
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
64
             e[b].push_back(a);
65
66
67
        void build() {
68
             d[0] = 0;
69
             dfs(0, 0);
70
71
    } lca;
```

6.3 树状数组

Listing 44: bit.cpp

```
#include <vector>
 1
 2
 3
    using namespace std;
 4
    template<typename T = int>
 5
    struct BIT {
 6
       vector<T> a;
 8
 9
       void init(int n) {
10
         vector<T>(n + 1).swap(a);
11
12
       void add(int i, T v) {
13
         for (int j = i + 1; j < (int)a.size(); j = (j | (j - 1)) + 1) {
14
15
           a[j] += v;
16
         }
17
18
       // [0, i)
19
       T sum(int i) const {
20
21
         T ret = T();
         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
      \textbf{void} \ \mathtt{set}(\textbf{int} \ \mathtt{i}, \ \mathtt{T} \ \mathtt{v}) \ \ \{
32
33
         add(i, v - get(i));
34
35
    };
```

6.4 并查集

Listing 45: union-find.cpp

```
#include <vector>
 1
 2
 3
    using namespace std;
 4
    struct DisjointSet {
 5
 6
        vector<int> p;
 7
 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
16
            return i == p[i] ? i : (p[i] = getp(p[i]));
17
18
19
        bool setp(int i, int j) {
20
            i = getp(i);
            j = getp(j);
21
            p[i] = j;
22
            return i != j;
23
24
25
    };
```

6.5 轻重权树剖分

Listing 46: chain-decomp.cpp

```
#include <cstdio>
    #include <vector>
    #include <algorithm>
 3
 4
   using namespace std;
 5
 6
 7
   const int MAXM = 16;
   const int MAXN = 1 << MAXM;</pre>
 8
   // Heavy-Light Decomposition
10
   struct TreeDecomposition {
11
      vector<int> e[MAXN], c[MAXN];
12
                    // subtree size
13
      int s[MAXN];
      int p[MAXN];
                    // parent id
14
                      // chain root id
      int r[MAXN];
15
      int t[MAXN];
                      // timestamp, index used in segtree
16
17
      int ts;
18
      void dfs_(int v, int f) {
19
20
        p[v] = f;
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
            dfs_(w, v);
25
            s[v] += s[w];
26
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];
42
               y = w;
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>
51
          int w = e[v][i];
          if (w != f && w != y) {
52
            decomp (w, v, w);
53
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
        ts = 0;
70
        dfs_(0, 0);
71
72
        decomp_(0, 0, 0);
73
      }
   } hld;
```

6.6 强连通分量

Listing 47: scc.cpp

```
#include <algorithm>
    #include <stack>
 2
    #include <vector>
 3
 4
 5
   using namespace std;
 6
 7
    struct SCCTarjan {
 8
        int n;
 9
        vector<vector<int>> e;
10
        vector<int> id;
11
        vector<vector<int>> scc;
12
13
        void init(int n) {
14
            this\rightarrown = n;
15
            vector<vector<int>> (n) .swap(e);
16
17
            id.resize(n);
18
            dfn.resize(n);
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];
33
            s.push(v);
            for (vector<int>::const iterator w = e[v].begin(); w != e[v].end(); ++w) {
34
35
                if (dfn[*w] == -1) {
                     dfs(*w);
36
37
                     low[v] = min(low[v], low[*w]);
                 } else if (dfn[*w] != -2) {
38
                     low[v] = min(low[v], dfn[*w]);
39
40
41
42
            if (low[v] == dfn[v]) {
43
                vector<int> t;
44
45
                do {
46
                     int w = s.top();
47
                     s.pop();
48
                     id[w] = (int)scc.size();
                     t.push back(w);
49
                     dfn[w] = -2;
50
51
                 } while (t.back() != v);
52
                scc.push back(t);
53
            }
54
        }
55
        int gao() {
56
57
            scc.clear();
            stack<int>().swap(s);
58
            timestamp = 0;
59
60
            fill(dfn.begin(), dfn.end(), -1);
61
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
    };
```

6.7 双连通分量

Listing 48: bcc.cpp

```
#include <algorithm>
    #include <stack>
    #include <utility>
 3
    #include <vector>
 4
 5
   using namespace std;
 6
 7
 8
    // TODO: cannot handle duplicate edges
    struct Tarjan {
10
        int n;
11
        vector<vector<int>> e;
```

```
12
13
        vector<int> cut;
14
        vector<pair<int, int>> bridge;
15
        vectorvector<pair<int, int>>> bcc;
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
        void add(int a, int b) {
25
             // assert(find(e[a].begin(), e[a].end(), b) == e[a].end());
26
27
             e[a].push_back(b);
             e[b].push_back(a);
28
29
        }
30
        vector<int> dfn, low;
31
32
        int timestamp;
33
        stack<pair<int, int>> s;
34
        void dfs(int v, int p) {
35
36
            int part = p == -1 ? 0 : 1;
             dfn[v] = low[v] = timestamp++;
37
38
             for (vector<int>::const_iterator w = e[v].begin(); w != e[v].end(); ++w) {
                 pair<int, int> f = make_pair(min(v, *w), max(v, *w));
39
                 if (dfn[*w] == -1) {
40
                     s.push(f);
41
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) {
                              cut.push_back(v);
47
48
                         // articulation edge
49
                         if (dfn[v] < low[*w]) {
50
51
                              bridge.push back(f);
52
                         }
53
                         // biconnected component (2-vertex-connected)
54
                         vector<pair<int, int>> t;
55
56
                              t.push_back(s.top());
57
                              s.pop();
58
                          } while (t.back() != f);
59
                         bcc.push_back(t);
                     }
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
69
             cut.clear();
70
            bridge.clear();
71
            bcc.clear();
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
```

```
78
                 if (dfn[i] == -1) {
79
                      dfs(i, -1);
80
81
82
83
     } ;
84
     struct BridgeBlockTree {
85
86
         TarjanMAXND bcc;
87
         DisjointSet<MAXN▷ ds;
88
         vector<int> e[MAXN];
89
90
         void init(int n) {
             bcc.init(n);
91
             ds.init(n);
92
93
94
95
         void add(int a, int b) {
             bcc.add(a, b);
96
97
98
99
         void gao() {
100
             bcc.gao();
             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);
                 e[b].push_back(a);
112
             }
113
         }
114
115
         int id(int v) {
116
117
             return ds.getp(v);
118
     };
119
```

6.8 二分图匹配

Listing 49: bimatch.cpp

```
// maximum matchings in bipartite graphs
    // maximum cardinality bipartite matching
    // O(|V||E|), generally fast
 3
 4
    #include <algorithm>
 5
    #include <string>
 6
    #include <vector>
 8
 9
    using namespace std;
10
    struct Hungarian {
11
12
        int nx, ny;
        vector<int> mx, my;
13
        vector<vector<int>> e;
14
15
        void init(int nx, int ny) {
16
17
            this\rightarrownx = nx;
18
            this->ny = ny;
19
            mx.resize(nx);
```

```
20
            my.resize(ny);
21
             e.clear();
22
             e.resize(nx);
23
            mark.resize(nx);
24
25
        void add(int a, int b) {
26
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;
                 for (vector<int>::const_iterator j = e[i].begin(); j != e[i].end(); ++j) {
36
37
                     if (my[*j] == -1 \mid | augment(my[*j])) {
                         mx[i] = *j;
38
                         my[*j] = i;
39
                         return true;
40
41
                     }
42
                 }
43
            return false;
44
        }
45
46
        int gao() {
47
48
             int ret = 0;
49
             fill(mx.begin(), mx.end(), -1);
             fill(my.begin(), my.end(), -1);
50
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
    };
```

6.9 最小费用最大流

Listing 50: flow.cpp

```
#include <algorithm>
    #include <cstdio>
    #include <limits>
    #include <queue>
 4
    #include <vector>
 5
 6
 7
    using namespace std;
 8
 9
    template <int MAXN, typename T = int, typename S = T>
10
    struct MinCostMaxFlow {
11
        struct NegativeCostCircuitExistsException {
12
13
        struct Edge {
14
            int v;
15
            T c;
16
            S w;
17
            int b;
18
19
            Edge(int v, T c, S w, int b) : v(v), c(c), w(w), b(b) {}
20
        };
21
```

```
22
         int n, source, sink;
23
         vector Edge> e [MAXN];
24
         void init(int n, int source, int sink) {
25
              this\rightarrown = n;
26
27
              this->source = source;
              this->sink = sink;
28
29
              for (int i = 0; i < n; ++i) {</pre>
30
                  e[i].clear();
31
32
33
34
         void addEdge(int a, int b, T c, S w) {
35
              e[a].push_back(Edge(b, c, w, e[b].size()));
              e[b].push back(Edge(a, 0, -w, e[a].size() -1)); // TODO
36
37
38
39
         bool mark[MAXN];
         T maxc[MAXN];
40
41
         S minw[MAXN];
         int dist[MAXN];
42
43
         Edge *prev[MAXN];
44
45
         bool _spfa() {
46
              queue<int> q;
              fill(mark, mark + n, false);
47
48
              fill(maxc, maxc + n, 0);
              fill(minw, minw + n, numeric_limits<S>::max());
49
50
              fill(dist, dist + n, 0);
51
              fill(prev, prev + n, (Edge *)NULL);
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
                  mark[cur] = false;
59
60
                  a.pop();
                  for (typename vector<Edge>::iterator it = e[cur].begin(); it != e[cur].end(); ++it) {
61
62
                       T c = min(maxc[cur], it \rightarrow c);
                       if (c == 0) {
63
64
                            continue;
65
                        }
66
67
                       int v = it->v;
68
                       S w = minw[cur] + it \rightarrow w;
                       \textbf{if} \ (\texttt{minw}[\texttt{v}] \ > \ \texttt{w} \ | \ | \ (\texttt{minw}[\texttt{v}] \ == \ \texttt{w} \ \&\& \ \texttt{maxc}[\texttt{v}] \ < \ \texttt{c}) \,) \ \{ \ \textit{// TODO} \ \}
69
                            maxc[v] = c;
70
                            minw[v] = w;
71
72
                            dist[v] = dist[cur] + 1;
                            if (dist[v] >= n) {
73
74
                                 return false;
75
76
                            prev[v] = &*it;
77
                            if (!mark[v]) {
                                 mark[v] = true;
78
79
                                 q.push(v);
80
                            }
                       }
81
82
                  }
83
84
              return true;
85
86
87
         pair<T, S> gao() {
```

```
88
              T sumc = 0;
89
              S sumw = 0;
              while (true) {
90
                  if (!_spfa()) {
91
                      throw NegativeCostCircuitExistsException();
92
93
                  } else if (maxc[sink] == 0) {
94
                      break;
95
                  } else {
                      T c = maxc[sink];
96
97
                      sumc += c;
98
                      sumw += c * minw[sink];
99
100
                      int cur = sink;
                      while (cur != source) {
101
                          Edge *e1 = prev[cur];
102
                          e1->c -= c;
103
                          Edge *e2 = \&e[e1->v][e1->b];
104
105
                          e2->c += c;
                          cur = e2 \rightarrow v;
106
107
                       }
                  }
108
109
110
              return make pair(sumc, sumw);
111
112
     } ;
```

6.10 AhoCorasick 自动机

Listing 51: ac-automata.cpp

```
#include <algorithm>
 1
    #include <queue>
 2
 3
 4
    using namespace std;
 5
 6
    struct AhoCorasick {
 7
        static const int NONE = 0;
        static const int MAXN = 1024;
 8
 9
        static const int CHARSET = 26;
10
        int end;
11
        int tag[MAXN];
12
        int fail[MAXN];
13
        int trie[MAXN] [CHARSET];
14
15
16
        void init() {
17
            tag[0] = NONE;
18
            fill(trie[0], trie[0] + CHARSET, -1);
            end = 1;
19
        }
20
21
        int add(int m, const int *s) {
22
23
            int p = 0;
            for (int i = 0; i < m; ++i) {</pre>
24
25
                if (trie[p][*s] == -1) {
26
                     tag[end] = NONE;
27
                     fill(trie[end], trie[end] + CHARSET, -1);
                     trie[p][*s] = end++;
28
29
                p = trie[p] [*s];
30
                ++s;
31
32
33
            return p;
34
35
        void build(void) { // !!
```

```
37
             queue int bfs;
38
             fail[0] = 0;
             for (int i = 0; i < CHARSET; ++i) {</pre>
39
40
                 if (trie[0][i] != −1) {
                     fail[trie[0][i]] = 0;
41
42
                     bfs.push(trie[0][i]);
                 } else {
43
44
                     trie[0][i] = 0;
45
46
47
            while (!bfs.empty()) {
48
                 int p = bfs.front();
49
                 tag[p] |= tag[fail[p]];
50
                 bfs.pop();
                 for (int i = 0; i < CHARSET; ++i) {</pre>
51
                     if (trie[p][i] != −1) {
52
                         fail[trie[p][i]] = trie[fail[p]][i];
53
                         bfs.push(trie[p][i]);
54
55
                      } else {
                         trie[p][i] = trie[fail[p]][i];
56
57
58
                 }
59
             }
60
61
    } ac;
```

6.11 后缀数组

Listing 52: sa.cpp

```
#include <algorithm>
 1
 2
    #include <utility>
 3
    #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
11
            sa.resize(n);
            rank.resize(n);
12
13
            vector<pair<T, int>> assoc(n);
14
            for (int i = 0; i < n; ++i) {</pre>
15
16
                 assoc[i] = make_pair(a[i], i);
17
18
            sort(assoc.begin(), assoc.end());
            for (int i = 0; i < n; ++i) {</pre>
19
20
                 sa[i] = assoc[i].second;
21
                 if (i == 0 \mid \mid assoc[i].first != assoc[i - 1].first) {
22
                     rank[sa[i]] = i;
23
                 } else {
                     rank[sa[i]] = rank[sa[i-1]];
24
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
                     tmp[i] = n - m + i;
33
34
35
                 for (int i = 0, j = m; i < n; ++i) {</pre>
36
                     if (sa[i] >= m) {
```

```
37
                          tmp[j++] = sa[i] - m;
                     }
38
39
                 }
                 // fst
40
                 fill(cnt.begin(), cnt.end(), 0);
41
                 for (int i = 0; i < n; ++i) {</pre>
42
                     ++cnt[rank[i]];
43
44
                 }
45
                 partial sum(cnt.begin(), cnt.end(), cnt.begin());
                 for (int i = n - 1; i >= 0; —i) {
46
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
                 for (int i = 0; i < n; ++i) {</pre>
53
                     if (i == 0 \mid \mid suffix[sa[i]] != suffix[sa[i-1]]) {
54
55
                          rank[sa[i]] = i;
56
                      } else {
                          rank[sa[i]] = rank[sa[i-1]];
57
58
59
                 }
60
             }
61
             height.resize(n);
62
             for (int i = 0, z = 0; i < n; ++i) {</pre>
63
                 if (rank[i] == 0) {
64
                     height[0] = z = 0;
65
66
                 } else {
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
                     height[rank[i]] = z;
72
73
                 }
74
             }
75
        }
76
    };
```

6.12 LU 分解

Listing 53: lu.cpp

```
const int MAXN = 128;
 2
    const double EPS = 1e-10;
 3
    void LU(int n, double a[MAXN] [MAXN], int r[MAXN], int c[MAXN]) {
 4
        for (int i = 0; i < n; ++i) {</pre>
 5
 6
            r[i] = c[i] = i;
 7
 8
        for (int k = 0; k < n; ++k) {
             int ii = k, jj = k;
 9
10
             for (int i = k; i < n; ++i) {</pre>
11
                 for (int j = k; j < n; ++j) {
12
                     if (fabs(a[i][j]) > fabs(a[ii][jj])) {
13
                         ii = i;
14
                         jj = j;
                     }
15
                 }
16
17
            swap(r[k], r[ii]);
18
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
24
              swap(a[k][j], a[ii][j]);
25
26
          if (fabs(a[k][k]) < EPS) {
              continue;
27
28
29
          for (int i = k + 1; i < n; ++i) {</pre>
              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
   38
39
       static double x[MAXN];
       for (int i = 0; i < n; ++i) {</pre>
40
41
          x[i] = b[r[i]];
42
43
       for (int i = 0; i < n; ++i) {</pre>
44
          for (int j = 0; j < i; ++j) {
45
              x[i] = a[i][j] * x[j];
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
          b[c[i]] = x[i];
57
58
59
   }
60
   // LU(n - 1, a, r, c);
61
   // solve(n - 1, a, r, c, b);
```

7 对一类问题的处理方法