

Alfred 代码模版库

目录

目	录					45
1	比赛配	置 and 奇技淫巧	4	5.22	06A - 欧拉函数(求解单个数的欧拉函数)	46
		8组数据代码模板	4	5 23	06B - 欧拉函数(求解全部数的欧拉函	10
		快读快写	4	0.20	数)	46
		长闭流与 C 风格输入输出的同步	4	5.24	07A - 组合数(小范围预处理, 逆元 +	
		clang-format	5 5		杨辉三角)	47
		ebug.n	5 6	5.25	07B - 组合数 (Comb, with. ModInt-	
		-cpp-properties.json	7		Base)	47
		aunch.json	8	5.26	08 - 素数测试与因式分解 (Miller-Rabin	
		ettings.json	8		and Pollard-Rho)	48
	1.10 ta	asks.json	8	5.27	09A - 平面几何(Point)	50
<b>2</b>	数据结	***	9	5.28	09B - 平面几何(with. std::complex) .	56
4		759 可朵莉树	9	5.29	10 - 立体几何(Point)	56
		树状数组	10	5.30	11A - 静态凸包(with. Point,旧版) .	57
		静态可重区间信息(支持 RMQ)	11	5.31	11B - 静态凸包(with. Point,新版) .	59
		BDS 大常数平衡树	12	5.32	11C - 静态凸包(with. std::complex).	59
		哥散化容器	12		12A - 多项式(Poly, with. Z)	60
		并查集	12		12B - 多项式(Poly, with. MInt)	65
		T撤销并查集	13		12C - 多项式乘法	72
		出现次数统计	14 15		13A - 生成函数 (q-int)	74
		1-1rie	16		13B - 生成函数(q-Binomial)	74
		(二维) 前缀和	17 17		13C - 生成函数 (Binomial 任意模数二	11
				0.00	项式)	75
3		数论)算法	18	5.39	14 - 自适应辛普森法 Simpson	77
		持模整数类	18		15 - 矩阵(Matrix)	78
		十算几何	19		16 - 高斯消元 (guess)【久远】	79
		且合数学 立格朗日插值	20 21		01A - 树状数组(Fenwick 旧版)	79
	3.4 11	型作的口油阻	21		01B - 树状数组(Fenwick 新版)	80
4	字符串	算法	22		02 - 并查集(DSU)	81
	4.1 学	字符串哈希	22			01
_	**1-	- 心可序(夕田 . 但切注担山:)	00	5.45	03A - 线段树(SegmentTree+Info 区间加 + 单点修改)	81
Э		y 代码库 (备用,侵权请提出 issue) 1 - int128 库函数自定义	<b>22</b> 22	5.46	03B - 线段树(SegmentTree 区间乘 +	01
		2 - 常用库函数重载	23	0.40	单点加) · · · · · · · · · · · · · · · · · · ·	83
		3 - 字符调整	$\frac{23}{24}$	5.47	03C - 线段树(SegmentTree+Info 初始	
		4A - 二分算法(整数域)	25	0.11	赋值 + 单点修改 + 查找前驱后继)	84
	5.5   0	4B - 二分算法(实数域)	25	5.48	03D - 线段树(SegmentTree+Info+Merge	
		1 - 强连通分量缩点(SCC)	26	0.10	初始赋值 + 单点修改 + 区间合并)	86
		2 - 割边与割边缩点(EBCC)	27	5.49	04 - 懒标记线段树(LazySegmentTree)	87
		3 - 二分图最大权匹配(MaxAssign-	29		05A - 取模类(Z 旧版)	90
		nent 基于 KM)	29		05B - 取模类 (MLong and MInt 新版)	92
		生)【久远】	31		05C - 动态取模类(ModIntBase)	95
		5 - TwoSat (2-Sat)	33		06 - 状压 RMQ (RMQ)	97
		6A - 最大流(Flow 旧版其一,整数应				99
		月)	33		07B - Splay	
		6B - 最大流(Flow 旧版其二,浮点数			07C - Splay	
			35			
		6C - 最大流(MaxFlow 新版) 7A - 费用流(MCFGraph 旧版)	$\frac{36}{38}$		08A - 其他平衡树	
		7A - 货用流(MCFGraph 旧版) 7B - 费用流(MinCostFlow 新版)	38 39		08B - 其他平衡树	
		7B - 気角流(MillCostriow <sub>初成</sub> ) 8 - 树链剖分(HLD)	39 41		08C - 其他平衡树	
		1 - 快速幂	43		08D - 其他平衡树	
		2 - 基姆拉尔森公式	43		09 - 分数四则运算(Frac)	
	5.19 03	3 - 欧拉筛	44		10 - 线性基(Basis)	
	$5.20 \ 0$	4 - 莫比乌斯函数筛(莫比乌斯反演) .	44	5.63	143 - 高精度(BigInt)	114

目录	目录

6	Watashi 代码库 (备用)	115		6.7	双连通分量	121
	6.1 $O(n \log n) - O(1)$ RMQ	. 115		6.8	二分图匹配	123
	6.2 $O(n \log n) - O(\log n)$ LCA	. 116			最小费用最大流	
	6.3 树状数组				AhoCorasick 自动机	
	6.4 并查集				后缀数组	
	6.5 轻重权树剖分			6.12	LU 分解	128
	6.6 强连通分量		-	7+	<b>举问题的外理方法</b>	129
		1.20	- 1	X11 —	尖叫动的外性 万法	1 /

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

# 1.2 快读快写

## Listing 2: fast-io.cpp

```
namespace fastIO {
1
        char c, f, e = 0;
2
3
        namespace usr {
            template <class _Tp>
4
            inline int read(_Tp &x) {
5
                x = f = 0, c = getchar();
6
                while (!isdigit(c) \delta\delta !e) f = c == '-', e |= c == EOF, c = getchar();
7
8
                while (isdigit(c) && !e) x = (x << 1) + (x << 3) + (c^48), c = getchar();
                return (e |= c == EOF) ? 0 : ((f ? x = -x : 0), 1);
9
10
            template <class _Tp>
11
            inline void write(_Tp x) {
12
                if (x < 0) putchar('-'), x = -x;
13
                if (x > 9) write(x / 10);
14
                putchar((x % 10) ^ 48);
15
16
            template <typename T, typename... V>
17
            inline void read(T &t, V &...v) { read(t), read(v...); }
18
            template <typename T, typename... V>
19
            inline void write(T t, V... v) {
20
                write(t), putchar('_'), write(v...);
21
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

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

## 1.4 .clang-format

### Listing 4: .clang-format

```
BasedOnStyle: LLVM
   AlignAfterOpenBracket: BlockIndent
   # AlignConsecutiveAssignments: Consecutive
3
   AlignArrayOfStructures: Right
4
5 UseTab: Never
   IndentWidth: 4
   TabWidth: 4
8 BreakBeforeBraces: Attach
9 AllowShortIfStatementsOnASingleLine: AllIfsAndElse
10 AllowShortLoopsOnASingleLine: true
11 AllowShortBlocksOnASingleLine: true
12 IndentCaseLabels: true
13 ColumnLimit: 0
14 AccessModifierOffset: -4
15 NamespaceIndentation: All
   FixNamespaceComments: false
16
17
   AllowShortCaseLabelsOnASingleLine: true
18
   AlwaysBreakTemplateDeclarations: MultiLine
19
   BinPackParameters: true
20
   BraceWrapping:
     AfterCaseLabel: true
21
     AfterClass: true
22
   AlignConsecutiveMacros: AcrossEmptyLinesAndComments
23
   AlignTrailingComments: Always
```

## 1.5 debug.h

### Listing 5: debug.h

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

1 比赛配置 AND 奇技淫巧 1.6 火车头

```
void __print(double x) { cerr << x; }</pre>
    void print(long double x) { cerr << x; }</pre>
    void __print(char x) { cerr << '\'' << x << '\''; }</pre>
35
    void \_print(const char *x) { cerr << '\"' << x << '\"'; }
36
    void __print(const string &x) { cerr << '\"' << x << '\"'; }</pre>
37
    void __print(bool x) { cerr << (x ? "true" : "false"); }</pre>
38
    void __print(__int128_t x) {
39
        if (x < 0) cerr << '-', x = -x;
40
        if (x > 9) print(x / 10);
41
        cerr << char((x % 10) ^ 48);
42
    }
43
    void dbgEndl(void) { cerr << '\n'; }</pre>
44
45
    template <typename T, typename V>
46
    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
        int f = 0;
52
        cerr << '{';
53
        for (auto i : x) cerr << (f++ ? ",_" : ""), __print(i);
54
        cerr << "}";
55
    }
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
        print(v...);
62
63
    #ifdef DEBUG
64
    // To customize a struct/class to print, just define the __print function.
65
66
    #ifndef NO DBG COLOR
67
68
    #define dbg(x...)
        cerr << "\e[91m" << __func__ << ":" << __LINE__ << "_[" << #x << "]_=_["; \
69
70
        _print(x);
        cerr << "\e[39m";
71
72
73
    #define short_dbg(x...) \setminus
        cerr << "\e[91m[["; \
74
        print(x);
75
        cerr << "\e[39m";
76
    #else
77
78
    #define dbg(x...)
        cerr << __func__ << ":" << __LINE__ << "_[" << #x << "]_=_["; \
79
         _print(x);
80
    #define short_dbg(x...) \
81
        cerr << "[";
82
83
         _print(x);
    #endif // !NO_DBG_COLOR
84
85
    #else
86
    #define dbg(x...)
87
    #endif
88
```

### 1.6 火车头

Listing 6: optimize-header.h

```
1 #pragma GCC optimize(3)
2 #pragma GCC target("avx")
```

```
#pragma GCC optimize("Ofast")
    #pragma GCC optimize("inline")
    #pragma GCC optimize("-fgcse")
5
    #pragma GCC optimize("-fgcse-lm")
6
    #pragma GCC optimize("-fipa-sra")
    #pragma GCC optimize("-ftree-pre")
8
    #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
18
    #pragma GCC optimize("-fcaller-saves")
19
    #pragma GCC optimize("—fcrossjumping")
    #pragma GCC optimize("-fthread-jumps")
20
    #pragma GCC optimize("-funroll-loops")
21
    #pragma GCC optimize("-fwhole-program")
22
23
    #pragma GCC optimize("-freorder-blocks")
    #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
    #pragma GCC optimize("no-stack-protector")
35
    #pragma GCC optimize("-freorder-functions")
36
    #pragma GCC optimize("-findirect-inlining")
37
    #pragma GCC optimize("-fhoist-adjacent-loads")
38
    #pragma GCC optimize("-frerun-cse-after-loop")
39
    #pragma GCC optimize("inline-small-functions")
40
    #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")
    #pragma GCC optimize("inline-functions-called-once")
46
    #pragma GCC optimize("-fdelete-null-pointer-checks")
47
```

## 1.7 c-cpp-properties.json

### Listing 7: c-cpp-properties.json

```
{
1
        "configurations": [
2
3
                 "name": "macos_gcc_arm64";
 4
                 "includePath": [
5
                     "${workspaceFolder}/**"
6
                     "/usr/local/include/ac—library/"
8
                 "compilerPath": "/usr/local/bin/g++",
9
                 "cStandard": "c17",
10
                 "cppStandard": "c++20",
11
```

## 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
                 "program": "${fileDirname}/compiled.out",
8
                 "args": [],
9
                 "stopAtEntry": false,
10
                 "cwd": "dollar{fileDirname}",
11
                 "environment": [],
12
                 "externalConsole": true,
13
                 "internalConsoleOptions": "neverOpen",
14
                 "MIMode": "lldb",
15
                 "setupCommands": [
16
17
                     {
                          "description": "Enable_pretty-printing_for_lldb",
18
                          "text": "-enable-pretty-printing",
19
                          "ignoreFailures": false
20
                     }
21
                 ],
22
                 "preLaunchTask": "Compile"
23
            }
24
         ],
25
    }
26
```

### 1.9 settings.json

### Listing 9: settings.json

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

### 1.10 tasks.json

## Listing 10: tasks.json

```
"args": [
8
                "${file}"
9
                "-o", // 指定输出文件名,不加该参数则默认输出a.exe, Linux下默认a.out
10
                "${fileDirname}/compiled.out",
11
                "-g", // 生成和调试有关的信息
12
                // "-arch aarch64",
13
                // "-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
                "-DDEBUG",
18
                "-03",
19
                "—ld_classic", // will be deprecated
20
                "—Wno-char-subscripts",
21
                "—I",
22
                "/usr/local/include/ac-library/"
23
                // "---stack=268435456"
                                         // 手动扩大栈空间
24
             ], // 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
             group: {
27
                "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Clangd可以注释掉
38
         }
39
      ]
40
   }
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 <bits/stdc++.h>
1
2
    struct ChthollyTree {
3
4
        using i64 = long long;
5
        struct Node {
6
            mutable i64 l, r, v;
 7
            inline bool operator (const Node δx) const { return l < x.l; }
        };
8
9
        std::set<Node> tr;
        using iterator = std::set<Node>::iterator;
10
        ChthollyTree(void) = default;
11
        ChthollyTree(int rng, int val) { init(rng, val); }
12
        inline void init(i64 rng, i64 val) noexcept {
13
            tr.insert({1, rng, val}), tr.insert({rng + 1, rng + 1, 0});
14
15
        inline iterator begin(void) const noexcept { return tr.begin(); }
16
        inline iterator end(void) const noexcept { return tr.end(); }
17
```

2 数据结构 2.2 树状数组

```
inline iterator split(i64 pos) {
18
            auto it = tr.lower bound({pos, 0, 0});
19
            if (it != tr.end() && it->l == pos) return it;
20
            i64 l = (--it)->l, r = it->r, v = it->v;
21
            tr.erase(it), tr.insert(\{l, pos - 1, v\});
22
            return tr.insert({pos, r, v}).first;
23
24
        inline void assign(i64 l, i64 r, i64 v) {
25
            auto R = split(r + 1), L = split(l);
26
            tr.erase(L, R), tr.insert({l, r, v});
27
28
        template <class Functor> // func(iterator)
29
30
        inline void modify(i64 l, i64 r, Functor func) {
            auto R = split(r + 1), L = split(l);
31
            for (auto it = L; it != R; it++) func(it);
32
33
        template <class _Functor> // func(i64 &, iterator)
34
        inline i64 query(i64 l, i64 r, _Functor func) {
35
            i64 \text{ ans} = 0;
36
            auto R = split(r + 1);
37
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

```
#include "fenwick.h"
1
    #include <bits/stdc++.h>
2
3
    template <class T>
4
    struct Fenwick {
5
        std::vector<T> c;
6
        inline int lowbit(int x) { return x \delta -x; }
7
        inline void merge(T \delta x, T \delta y) { x = x + y; }
8
        inline T subtract(T x, T y) { return x - y; }
        inline void update(size_t pos, T x) {
10
            for (pos++; pos < c.size(); pos += lowbit(pos)) merge(c[pos], x);</pre>
11
12
        inline void clear(void) {
13
            for (auto \delta x : c) x = T();
14
15
        inline T query(size t pos) {
16
            T ans = T();
17
            for (pos++; pos; pos ^= lowbit(pos)) merge(ans, c[pos]);
18
            return ans;
19
20
        inline T query(size_t l, size_t r) {
21
            return l == 0? query(r): subtract(query(r), query(l - 1));
22
23
        inline int kth(const T k) {
24
            int ans = 0;
25
            for (int i = 1 \ll std: _lg(c.size() - 1); i; i >>= 1) {
26
                 if (ans + i < (int)c.size() && c[ans + i] <= k) {
27
                     k = c[ans + i], ans += i;
28
29
30
            return ans;
31
32
```

```
33     Fenwick(size_t len) : c(len + 2) {}
34  };
```

## 2.3 静态可重区间信息(支持 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

```
#include <bits/stdc++.h>
1
2
3
    template <class T>
4
    struct MaxInfo {
5
        T val;
        MaxInfo(void) { val = std::numeric_limits<T>::min(); }
6
        template <class InitT>
 7
        MaxInfo(InitT x) { val = x; }
8
        MaxInfo operator+(MaxInfo &x) {
9
            return {std::max(val, x.val)};
10
11
    };
12
    template <class T>
13
    struct MinInfo {
14
        T val;
15
        MinInfo(void) { val = std::numeric limits<T>::max(); }
16
        template <class InitT>
17
        MinInfo(InitT x) { val = x; }
18
        MinInfo operator+(MinInfo &x) {
19
            return {std::min(val, x.val)};
20
21
22
    };
    template <class T>
23
    struct GcdInfo {
24
        T val;
25
        GcdInfo(void) { val = T(); }
26
27
        template <class InitT>
        GcdInfo(InitT x) { val = x; }
28
        GcdInfo operator+(GcdInfo &x) {
29
    #if __cplusplus >= 201703L
30
            return {std::gcd(x.val, val)};
31
    #else
32
            return { gcd(x.val, val)};
33
    #endif
34
35
36
    template <class T>
37
    class SparseTable {
38
    private:
39
40
        int n;
        std::vector<std::vector<T>> ST;
41
42
    public:
43
        SparseTable(void) {}
44
        SparseTable(int N) : n(N), ST(N, std::vector<T>(std::_lg(N) + 1)) {}
45
        template <class InitT>
46
        SparseTable(std::vector<InitT> &init) : SparseTable(init.size()) {
47
            for (int i = 0; i < n; i++) ST[i][0] = T(init[i]);
48
            for (int i = 1; (1 << i) <= n; i++) {
49
                for (int j = 0; j + (1 << i) - 1 < n; j++) {
50
                    ST[j][i] = ST[j][i-1] + ST[j + (1 << (i-1))][i-1];
51
52
            }
53
54
        inline T query(int l, int r) \{ // 0 \text{ based} \}
55
```

```
if (l > r) return T();
56
            int w = std:: \lg(r - l + 1);
57
            return ST[l][w] + ST[r - (1 << w) + 1][w];
58
59
        inline T disjoint_query(int l, int r) {
60
            T ans = T();
61
            for (int i = l; i \le r; i + (1 << std:__lg(r - i + 1))) {
62
                ans = ans + ST[i][std::_lg(r - i + 1)];
63
64
            return ans;
65
66
   };
67
```

### 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 = treeXT, null_type, Cmp, TreeTag, tree_order_statistics_node_updates;</pre>
```

### 2.5 离散化容器

### Listing 15: discretization.cpp

```
#include <bits/stdc++.h>
1
2
   template <class _Tp>
3
    struct Mess {
4
        std::vector< Tp> v;
5
        bool initialized = false;
6
        inline _Tp origin(int idx) { return v[idx - 1]; }
7
        inline void insert(_Tp x) { v.push_back(x); }
8
        template <typename T, typename... V>
9
        inline void insert(T x, V... v) { insert(x), insert(v...); }
10
        inline void init(void) {
11
            sort(v.begin(), v.end()), initialized = true;
12
            v.erase(unique(v.begin(), v.end()), v.end());
13
14
        inline void clear(void) { v.clear(), initialized = false; }
15
        inline int query(_Tp x) {
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
   };
21
```

### 2.6 并查集

Listing 16: dsu.cpp

```
1 #include <bits/stdc++.h>
2
3 struct DSU {
4 std::vector<int> fa, siz;
```

2 数据结构 2.7 可撤销并查集

```
DSU(int n) : fa(n + 1), siz(n + 1, 1) {
5
            std::iota(fa.begin(), fa.end(), 0);
6
7
        inline int find(int x) {
8
            return fa[x] == x ? x : fa[x] = find(fa[x]);
9
10
        inline bool same(int x, int y) {
11
12
            return find(x) == find(y);
13
        // true if x and y were not in the same set, false otherwise.
14
15
        inline bool merge(int x, int y) {
            int fx = find(x), fy = find(y);
16
            if (fx == fy) return false;
17
            if (siz[fx] < siz[fy]) swap(fx, fy);
18
            fa[fy] = fx, siz[fx] += siz[fy], siz[fy] = 0;
19
            return true;
20
21
        // x \rightarrow y, a.k.a let x be son of y (disable merge by rank).
22
        inline bool directed merge(int x, int y) {
23
            int fx = find(x), fy = find(y);
24
            if (fx == fy) return false;
25
            fa[fx] = fy, siz[fy] += siz[fx], siz[fx] = 0;
26
            return true;
27
28
    };
29
```

## 2.7 可撤销并查集

### Listing 17: cancel-dsu.cpp

```
#include <bits/stdc++.h>
1
    struct CancelDSU {
3
        std::stack<int> S;
4
        std::vector<int> fa, siz;
5
        CancelDSU(int n): fa(n + 1), siz(n + 1, 1) {
6
 7
            std::iota(fa.begin(), fa.end(), 0);
8
        inline int find(int x) {
9
            return fa[x] == x ? x : find(fa[x]);
10
11
        inline bool same(int x, int y) {
12
            return find(x) == find(y);
13
14
        inline void merge(int u, int v) {
15
            int fu = find(u), fv = find(v);
16
            if (fu == fv) return S.push(-1);
17
            if (siz[fu] < siz[fv]) swap(fu, fv);</pre>
18
            siz[fu] += siz[fv], fa[fv] = fu, S.push(fv);
19
20
        inline void _cancel(void) {
21
            if (S.empty()) return;
22
23
            if (S.top() == -1) return S.pop();
            siz[fa[S.top()]] -= siz[S.top()];
24
            fa[S.top()] = S.top(), S.pop();
25
26
        inline void cancel(int t = 1) {
27
            while (t—) _cancel();
28
29
    };
30
```

2 数据结构 2.8 出现次数统计

## 2.8 出现次数统计

#include <bits/stdc++.h>

1

### O(nlogn) 预处理, O(logn) 查找的出现次数在线统计

Listing 18: appear-statistics.cpp

```
2
    template <class _Tp>
3
    struct Mess {
4
        std::vector< Tp> v;
5
6
        bool initialized = false;
7
        inline _Tp origin(int idx) { return v[idx - 1]; }
        inline void insert(_Tp x) { v.push_back(x); }
8
        template <typename T, typename... V>
9
        inline void insert(T x, V... v) { insert(x), insert(v...); }
10
        inline void init(void) {
11
            sort(v.begin(), v.end()), initialized = true;
12
            v.erase(unique(v.begin(), v.end()), v.end());
13
14
        inline int query(_Tp x) {
15
            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
    template <class T>
22
    class AppearStats { // Appear Statistics.
23
    private:
24
        Mess<T> M;
25
26
        size_t n;
        std::vector<std::vector<int>> pos;
27
28
    public:
29
        AppearStats(void) : n(0) {}
30
        AppearStats(std::vector<T> &init) : n(init.size()) { _init(init); }
31
        inline void _init(std::vector<T> &init) {
32
            for (auto item : init) M.insert(item);
33
            n = init.size(), M.init(), pos.resize(M.v.size());
34
            for (size t i = 0; i < n; i++) {
35
                pos[M.query(init[i]) - 1].push_back(i);
36
37
        }
38
        // Use [base] as the beginning of index, return -1 if x doesn't exist.
39
        inline int first(int l, int r, T x, int base = 0) {
40
            l = base, r = base;
41
            if (!M.exist(x)) return -1;
42
            std::vector<int> \&P = pos[M.query(x) - 1];
43
            auto it = std::lower_bound(P.begin(), P.end(), l);
            return it == P.end() \mid \mid *it > r ? -1 : *it + base;
45
46
        // Use [base] as the beginning of index, return —1 if x doesn't exist.
47
        inline int last(int l, int r, T x, int base = 0) {
48
            l = base, r = base;
49
            if (!M.exist(x)) return -1;
50
            std::vector<int> \delta P = pos[M.query(x) - 1];
51
            auto it = std::upper bound(P.begin(), P.end(), r);
52
            return it == P.begin() || *std::prev(it) < l ? -1 : *std::prev(it) + base;
53
54
        inline int count(int l, int r, T x, int base = 0) {
55
            l = base, r = base;
56
            if (!M.exist(x)) return 0;
57
            std::vectorkint> \delta P = pos[M.query(x) - 1];
58
```

2 数据结构 2.9 01-Trie

```
auto L = std::lower_bound(P.begin(), P.end(), l);
auto R = std::upper_bound(P.begin(), P.end(), r);
if (L == P.end() || R == P.begin()) return 0;
if (*L > r || *std::prev(R) < l) return 0;
return R - L;
}</pre>
```

#### 2.9 01-Trie

### Listing 19: binary-trie.cpp

```
// Thanks neal for this template.
2
    #include <bits/stdc++.h>
3
    const int BITS = 30;
    const int INF = 1e9 + 7;
    struct BinaryTrie { // 01—Trie
7
        static const int ALPHABET = 2;
        struct Node {
8
             const int parent;
9
             int words here = 0;
                                     // How many words EXACTLY here.
10
             int starting_with = 0; // How many words have the PREFIX of this node.
11
             int min_index = INF; // The minimum index of words which have PREFIX of this node. int max_index = —INF; // The maximum index of words which have PREFIX of this node.
12
13
             std::array<int, ALPHABET> child;
14
             Node(int p = -1): parent(p) { child.fill(-1); }
15
        };
16
        static const int ROOT = 0;
17
        std::vector<Node> tr = {Node()};
18
        BinaryTrie(int total_length = -1) { // Sum of |s|, leave -1 if don't know.
19
             if (total_length >= 0) tr.reserve(total_length + 1);
20
21
        // Returns the Node reference of word.
22
        // NOTICE: this function creates a new Node if word isn't in the trie.
23
24
        Node & Soperator[](uint64_t word) {
             return tr[build(word, 0)];
25
26
        // Get or create c—th (c = 0, 1) child of node
27
        // Returns BinaryTrie node.
28
        int get_or_create_child(int node, int c) {
29
             if (tr[node].child[c] == -1) {
30
                 tr[node].child[c] = (int)tr.size();
31
                 tr.push back(Node(node));
32
33
             return tr[node].child[c];
34
35
        // Build rootpath of word, insert delta (个) words
36
37
        // Returns BinaryTrie node.
        int build(uint64_t word, int delta) {
38
             int node = ROOT;
39
40
             for (int i = BITS - 1; i >= 0; i—) {
                 tr[node].starting_with += delta;
41
42
                 node = get_or_create_child(node, word >> i & 1);
43
44
             tr[node].starting_with += delta;
             return node;
45
46
        // Insert a word with the index of index, INF if index is unknown.
47
        // Returns BinaryTrie node.
48
        int insert(uint64_t word, int index = INF) {
49
             int node = build(word, 1);
50
             tr[node].words_here += 1;
51
52
             for (int x = node; x != -1; x = tr[x].parent) {
```

2 数据结构 2.10 滑动窗口

```
if (index != INF) {
53
                      tr[x].min_index = std::min(tr[x].min_index, index);
54
                      tr[x].max_index = std::max(tr[x].max_index, index);
55
56
57
             return node;
58
59
         // Find such an x inserted in the trie that word ^ x is minimized.
60
         // Returns such x (x is certain).
61
62
         uint64_t query_min(uint64_t word) {
63
             int node = ROOT;
64
             uint64_t val = 0;
65
             for (int i = BITS - 1; i >= 0; i ---) {
66
                 int go_bit = word >> i & 1;
67
                 if (tr[node].child[go_bit] == -1) {
                     go_bit ^= 1;
68
69
70
                 val |= 1ull << go_bit;</pre>
71
                 node = tr[node].child[go bit];
 72
73
             return val;
         }
74
         // Find such an x inserted in the trie that word ^ x is maximized.
75
         // Returns such x (x is certain).
76
         uint64 t query max(uint64 t word) {
77
             int node = ROOT;
78
             uint64_t val = 0;
79
             for (int i = BITS - 1; i \ge 0; i \longrightarrow ) {
80
                 int go_bit = (word >> i & 1) ^ 1;
81
                 if (tr[node].child[go bit] == -1) {
82
                     go_bit ^= 1;
83
                 }
84
85
                 val |= 1ull << go_bit;
86
                 node = tr[node].child[go_bit];
87
88
             return val;
         }
89
         // CF1983F: Find such an x inserted in the trie that word ^x < \text{upper bound}
90
         // Returns a pair {min_index, max_index} of x.
91
         std::pair<int, int> query_ub(uint64_t word, uint64_t upper_bound) {
92
             int mn = INF, mx = -INF, node = ROOT;
93
             for (int i = BITS - 1; i >= 0; i—) {
94
                 int word_bit = word >> i & 1;
                                                      // digit i of word
95
                 int ub bit = upper bound >> i & 1; // digit i of ub
96
                 if (ub\_bit == 1 \& tr[node].child[word\_bit] != -1) {
97
                      // if digit i of ub is 1, then we can choose either
98
                      // the subtree of word_bit or word_bit ^ 1.
99
                     mn = std::min(mn, tr[tr[node].child[word bit]].min index);
100
                     mx = std::max(mx, tr[tr[node].child[word_bit]].max_index);
101
102
                 // else if digit i of ub is 0, then we can only choose
103
                 // the subtree of word_bit. (otherwise, we will violate the range)
104
                 node = tr[node].child[word_bit ^ ub_bit];
105
106
                 if (node == -1) break;
107
108
             return {mn, mx};
109
    };
110
```

### 2.10 滑动窗口

Listing 20: sliding-window.cpp

1 #include <bits/stdc++.h>

2 数据结构 2.11 (二维)前缀和

```
2
    template <class T> // default max.
    std::vector<T> sliding window(std::vector<T> A, size t k) {
5
        std::vector<T> res;
6
        std::dequ≪size_t> Q;
        for (size t i = 0; i < A.size(); i++) {
7
            if (!Q.empty() && Q[0] + k == i) {
8
9
                Q.pop_front();
10
            while (!Q.empty() && A[Q.back()] \le A[i]) {
11
12
                Q.pop_back();
13
            Q.push back(i);
14
            if (i \ge k - 1) { // warning: assert k \ge 1
15
                res.push back(A[Q[0]]);
16
17
18
        return res;
19
    }
20
    template <class T>
21
    std::vector<std::vector<T>>> grid_sliding_window(
22
        std::vector<std::vector<T>> &A, size_t x, size_t y
23
    ) {
24
25
        const size_t n = A.size(), m = A[0].size();
        std::vector<std::vector<T>> cols(m - y + 1);
26
        std::vector<std::vector<T>> ans(n - x + 1, std::vector<math><T>(m - y + 1));
27
        for (size_t i = 0; i < n; i++) {
28
            std::vector<T> res = sliding_window(A[i], y);
29
            for (size_t j = 0; j <= m - y; j++) {
30
                cols[j].push_back(res[j]);
31
32
33
        for (size_t j = 0; j <= m - y; j++) {
34
35
            std::vector<T> res = sliding_window(cols[j], x);
            for (size t i = 0; i <= n - x; i++) {
36
                ans[i][j] = res[i];
37
38
39
40
        return ans;
41
    }
```

## 2.11 (二维)前缀和

Listing 21: **prefix-sum.cpp** 

```
#include <bits/stdc++.h>
1
2
    template <class T>
3
    class Sum {
4
    private:
5
6
        size t n;
7
        std::vector<T> sum;
8
    public:
9
        Sum(void) : n(0) \{\}
10
        template <class InitT>
11
        Sum(std::vector<InitT> &init) { _init(init); }
12
        template <class InitT>
13
        inline void _init(std::vector<InitT> &init) {
14
            if (init.empty()) return;
15
            sum.resize(n = init.size()), sum[0] = init[0];
16
            for (size_t i = 1; i < n; i++) {
17
18
                 sum[i] = sum[i - 1] + init[i];
```

```
19
20
        inline T query(int l, int r) {
21
            if (l > r) return T();
22
            return l == 0? sum[r] : sum[r] - sum[l - 1];
23
        }
24
   };
25
    template <class T>
    class GridSum {
27
28
   private:
29
        size_t n, m;
        std::vector<std::vector<T>> sum;
30
31
    public:
32
        GridSum(void) : n(0), m(0) \{ \}
33
        template <class InitT>
34
        GridSum(std::vector<InitT>> &init) { _init(init); }
35
        template <class InitT>
36
37
        inline void _init(std::vector<std::vector<InitT>> &init) {
            if (init.empty()) return;
38
            n = init.size(), m = init[0].size();
39
            sum.assign(n, std::vectorT>(m)), sum[0][0] = init[0][0];
40
            for (size t i = 1; i < n; i++) {
41
                sum[i][0] = sum[i - 1][0] + init[i][0];
42
43
            for (size t i = 1; i < m; i++) {
44
45
                sum[0][i] = sum[0][i - 1] + init[0][i];
46
            for (size_t i = 1; i < n; i++) {
47
                for (size_t j = 1; j < m; j++) {
48
                    sum[i][j] = sum[i - 1][j] + sum[i][j - 1] - sum[i - 1][j - 1] + init[i][j];
49
50
            }
51
52
        inline T query(int x1, int y1, int x2, int y2) {
53
            T s1 = x1 == 0 ? 0 : sum[x1 - 1][y2];
54
            T s2 = y1 == 0 ? 0 : sum[x2][y1 - 1];
55
56
            T s3 = x1 == 0 \mid \mid y1 == 0 ? 0 : sum[x1 - 1][y1 - 1];
57
            return sum[x2][y2] - s1 - s2 + s3;
        }
58
   };
59
```

# 3 数学(数论)算法

### 3.1 带模整数类

Listing 22: mod-int.cpp

```
#include <bits/stdc++.h>
1
2
   template <int mod>
3
   inline int down(int x) { return x >= mod ? x - mod : x; }
   template <int mod>
5
6
    struct ModInt {
7
        int x;
8
        ModInt(void) = default;
        ModInt(int x) : x(x) {}
9
        friend std::istream δoperator>>(std::istream δin, ModInt δa) { return in >> a.x; }
10
        friend std::ostream &operator<<(std::ostream &out, ModInt a) { return out << a.x; }</pre>
11
        friend ModInt operator+(ModInt a, ModInt b) { return down<mod>(a.x + b.x); }
12
        friend ModInt operator—(ModInt a, ModInt b) { return down<mod>(a.x - b.x + mod); }
13
        friend ModInt operator*(ModInt a, ModInt b) { return (long long)a.x * b.x % mod; }
14
        friend ModInt operator/(ModInt a, ModInt b) { return a * ~b; }
15
```

3 数学(数论)算法 3.2 计算几何

```
friend ModInt operator^(ModInt a, long long b) {
16
            ModInt ans = 1;
17
            for (; b; b >>= 1, a \star= a)
18
                if (b & 1) ans *= a;
19
            return ans;
20
        }
21
        friend ModInt operator~(ModInt a) { return a ^{(mod - 2)}; }
22
        friend ModInt operator—(ModInt a) { return down<mod>(mod - a.x); }
23
24
        friend ModInt & Soperator+=(ModInt & a, ModInt b) { return a = a + b; }
        friend ModInt Soperator—=(ModInt Sa, ModInt b) { return a = a - b; }
25
        friend ModInt & Operator*=(ModInt & a, ModInt b) { return a = a * b; }
26
        friend ModInt & Soperator/=(ModInt & a, ModInt b) { return a = a / b; }
27
        friend ModInt & Operator = (ModInt & , long long b) { return a = a ^ b; }
28
        friend ModInt & Operator++(ModInt & a) { return a += 1; }
29
        friend ModInt operator++(ModInt &a, int) {
30
            ModInt x = a;
31
            a += 1;
32
33
            return x;
34
        friend ModInt & Operator—(ModInt & a) { return a -= 1; }
35
        friend ModInt operator—(ModInt &a, int) {
36
            ModInt x = a;
37
            a = 1;
38
            return x;
39
40
        friend bool operator==(ModInt a, ModInt b) { return a.x == b.x; }
41
        friend bool operator!=(ModInt a, ModInt b) { return !(a == b); }
42
43
    inline void __print(mint x) { std::cerr << x; }</pre>
```

## 3.2 计算几何

### Listing 23: computation-geometry.cpp

```
#include <bits/stdc++.h>
1
2
    // Caution: This computation geometry template is pure shit
3
                because of the terrible math level of the author.
   //
4
   //
                It will be rewritten some time.
5
    template <class T>
6
    struct Point {
7
8
        T x, y;
        Point(void) = default;
9
        Point(T X, T Y) : x(X), y(Y) {}
10
        inline bool operator==(const Point B) {
11
12
            return x == B.x \delta\delta y == B.y;
13
        friend std::ostream & Soperator<<(std::ostream & Sout, Point P) {</pre>
14
            return out << "(" << P.x << ",_" << P.y << ")";
15
16
        friend std::istream & Soperator >> (std::istream & in, Point & P) {
17
            return in >> P.x >> P.y;
18
19
    };
20
    template <class T>
21
    struct Line {
22
        T A, B, C; // Ax + By + C = 0
23
        Line(void) = default;
24
        Line(T a, T b, T c) : A(a), B(b), C(c) \{\} // Ax + By + C = 0
25
        Line(T k, T b) : A(k), B(-1), C(b) {}
                                                   // y = kx + b
26
    };
27
    template <class T>
28
    inline int sign(T x) {
29
```

3 数学(数论)算法 3.3 组合数学

```
return x == 0 ? 0 : (x < 0 ? -1 : 1);
30
   }
31
   template <class T>
32
    inline bool parallel(LineT> P, LineT> Q) {
33
        return P.A * Q.B == P.B * Q.A;
34
35
   template <class T>
36
    inline Point<T> intersect(Line<T> P, Line<T> Q) {
37
        assert(!parallel(P, Q));
38
        return Point<T>{
39
            (P.C * Q.B - Q.C * P.B) / (Q.A * P.B - P.A * Q.B),
40
            (P.C * Q.A - Q.C * P.A) / (P.A * Q.B - Q.A * P.B)
41
        };
42
   }
43
    template <class T>
44
    inline Line<T> get_line(Point<T> P, Point<T> Q) {
45
        assert(!(P == Q));
46
47
        if (P.x == Q.x) {
            return LineT>(-1, 0, P.x);
48
        } else if (P.y == Q.y) {
49
            return LineT>(0, -1, P.y);
50
        } else {
51
            return Line≺T>(
52
                Q.y - P.y, P.x - Q.x, P.y * Q.x - P.x * Q.y
53
54
            );
55
    }
56
    template <class T>
57
    inline bool point on line(Point<T> P, Line<T> L) {
58
        return L.A * P.x + L.B * P.y + L.C == 0;
59
   }
60
   template <class T>
61
   inline T dis_square(Point<T> P, Point<T> Q) {
62
        return (P.x - Q.x) * (P.x - Q.x) + (P.y - Q.y) * (P.y - Q.y);
63
   }
64
```

## 3.3 组合数学

Listing 24: comb.cpp

```
#include <bits/stdc++.h>
1
2
3
    template <class mint>
4
    struct Comb {
5
        int n;
        std::vector<mint> _fac, _invfac, _inv;
6
        Comb(void) : n{0}, _fac{1}, _invfac{1}, _inv{0} {}
7
        Comb(int n) : Comb() { init(n); }
8
        inline void init(int m) {
9
             _fac.resize(m + 1), _inv.resize(m + 1), _invfac.resize(m + 1);
10
            for (int i = n + 1; i \le m; i++) {
11
                 _{fac[i]} = _{fac[i-1]} * i;
12
13
            _{invfac[m]} = _{fac[m]};
14
            for (int i = m; i > n; i—) {
15
                _{invfac[i-1] = _{invfac[i]} * i;}
                _{inv[i]} = _{invfac[i]} * _{fac[i-1]};
17
18
19
            n = m;
20
        inline mint fac(int m) {
21
            if (m > n) init(m);
22
            return _fac[m];
23
```

3 数学(数论)算法 3.4 拉格朗日插值

```
}
24
        inline mint invfac(int m) {
25
            if (m > n) init(m);
26
            return _invfac[m];
27
28
        inline mint inv(int m) {
29
            if (m > n) init(m);
30
            return _inv[m];
31
32
        inline mint binom(int n, int m) {
33
34
            if (n < m \mid | m < 0) return 0;
            return fac(n) * invfac(m) * invfac(n - m);
35
36
37
    };
38
    Comb<mint> comb;
```

## 3.4 拉格朗日插值

Listing 25: lagrange.cpp

```
#include "comb.h"
2
    #include "mod—int.h"
    #include <bits/stdc++.h>
3
4
5
    inline mint lagrange(std::vector<mint> &x, std::vector<mint> &y, mint k) {
6
        mint ans = 0, cur;
7
        const int n = x.size();
        for (int i = 0; i < n; i++) {
8
            cur = y[i];
9
            for (int j = 0; j < n; j++) {
10
                if (j == i) continue;
11
                cur *= (k - x[j]) / (x[i] - x[j]);
12
            }
13
14
            ans += cur;
        }
15
        return ans;
16
   }
17
   // y[0] is placeholder.
18
    // If for all integer x_i in [1, n], we have f(x_i) = y_i (mod p), find f(k) mod p.
19
    inline mint cont_lagrange(std::vector∢mint> &y, mint k) {
20
21
        mint ans = 0;
22
        const int n = y.size() - 1;
23
        std::vectormint> pre(n + 1, 1), suf(n + 2, 1);
        for (int i = 1; i \le n; i++) pre[i] = pre[i-1] * (k-i);
24
        for (int i = n; i >= 1; i—) suf[i] = suf[i + 1] * (k - i);
25
        for (int i = 1; i <= n; i++) {
26
            mint A = pre[i-1] * suf[i+1];
27
            mint B = comb.fac(i - 1) * comb.fac(n - i);
28
            ans += ((n-i) \& 1 ? -1 : 1) * y[i] * A / B;
29
        }
30
        return ans;
31
   }
32
    // find 1^k + 2^k + ... + n^k. in O(k) of time complexity.
33
    inline mint sum_of_kth_powers(mint n, int k) {
34
        mint sum = 0;
35
        std::vector<mint> Y{0};
36
        for (int i = 1; i \le k + 2; i++) {
37
            Y.push back(sum += (mint)i ^ k);
38
39
40
        return cont_lagrange(Y, n);
41
   }
```

# 4 字符串算法

## 4.1 字符串哈希

Listing 26: hashed-string.cpp

```
#include <bits/stdc++.h>
1
2
    template <int mod, int seed>
3
    struct SingleHash {
4
        int n;
5
        std::vector<int> pow, h;
6
        SingleHash(void) = default;
7
        SingleHash(std::string &s) { init(s); }
8
        inline void init(std::string &s) {
9
            n = s.size(), h.assign(n + 2, 0), pow.assign(n + 2, 1);
10
            for (int i = 1; i \le n; i++) {
11
                pow[i] = 1ll * pow[i - 1] * seed % mod;
12
                h[i] = (111 * h[i-1] * seed + s[i-1]) % mod;
13
            }
14
        }
15
        inline int get_hash(int l, int r) {
16
17
            return (h[r + 1] - 1ll * h[l] * pow[r - l + 1] % mod + mod) % mod;
18
19
        inline bool check same(int l1, int r1, int l2, int r2) {
            return get_hash(l1, r1) == get_hash(l2, r2);
20
21
   };
22
23
    struct HashedString {
24
        SingleHask 998244353, 477> H1;
25
        SingleHask1000000007, 233> H2;
26
        HashedString(void) = default;
        HashedString(std::string s) : H1(s), H2(s) {}
27
        inline void init(std::string s) {
28
29
            H1.init(s), H2.init(s);
30
        std::pair<int, int> get_hash(int l, int r) { // not recommended.
31
            return {H1.get_hash(l, r), H2.get_hash(l, r)};
32
33
        // caution: index begins with zero.
34
        // If index beginning with one is wanted, use s = ' ' + s
35
        inline bool check_same(int l1, int r1, int l2, int r2) {
36
            return H1.check same(l1, r1, l2, r2) && H2.check same(l1, r1, l2, r2);
37
38
        inline bool check_period(int l, int r, int p) {
39
            return check same(l, r - p, l + p, r);
40
41
    };
42
```

# 5 jiangly 代码库 (备用,侵权请提出 issue)

### 5.1 01 - int128 库函数自定义

Listing 27: others/01-i128-Func.cpp

```
1
   /**
         int128 输出流自定义
         2023-03-20: https://codeforces.com/contest/1806/submission/198413531
2
    *
    **/
3
   using i128 = __int128;
4
5
   std::ostream &operator<<(std::ostream &os, i128 n) {</pre>
6
7
       std::string s;
8
       while (n) {
```

```
s += '0' + n % 10;
 9
            n /= 10;
10
11
        std::reverse(s.begin(), s.end());
12
        return os << s;
13
    }
14
15
    std::istream & operator >> (std::istream & is, i128 & n) {
16
17
        std::string s;
        is >> s;
18
        for (auto &c:s) {
19
            n = n * 10 + c - '0';
20
21
        return is;
22
    }
23
24
    i128 toi128(const std::string &s) {
25
        i128 n = 0;
26
        for (auto c : s) {
27
            n = n * 10 + (c - '0');
28
29
        return n;
30
31
    }
32
33
    i128 sqrti128(i128 n) {
34
        i128 lo = 0, hi = 1E16;
        while (lo < hi) {
35
            i128 x = (lo + hi + 1) / 2;
36
            if (x * x <= n) {
37
                lo = x;
38
            } else {
39
                hi = x - 1;
40
41
        }
42
        return lo;
43
    }
44
```

## 5.2 02 - 常用库函数重载

## Listing 28: others/02-Math-Func.cpp

```
using i64 = long long;
    using i128 = __int128;
 2
 3
    /**
          上取整下取整
 4
    *
          2023-10-15: https://codeforces.com/contest/293/submission/228297248
 5
    **/
 6
    i64 ceilDiv(i64 n, i64 m) {
 7
        if (n >= 0) {
 8
            return (n + m - 1) / m;
 9
        } else {
10
            return n / m;
11
12
    }
13
14
    i64 floorDiv(i64 n, i64 m) {
15
        if (n >= 0) {
16
            return n / m;
17
        } else {
18
            return (n - m + 1) / m;
19
20
    }
21
22
    /**
          最大值赋值
```

```
2023-09-30: https://codeforces.com/contest/1874/submission/226069129
24
    **/
25
    template≺class T>
26
    void chmax(T &a, T b) {
27
        if (a < b) {
28
            a = b;
29
30
    }
31
32
    /**
          最大公约数
33
    *
34
          -: -
    **/
35
    i128 gcd(i128 a, i128 b) {
36
        return b ? gcd(b, a % b) : a;
37
    }
38
39
    /**
          精确开平方
40
          2024-03-02: https://qoj.ac/submission/343317
41
    *
    **/
42
43
    i64 sqrt(i64 n) {
        i64 s = std::sqrt(n);
44
        while (s * s > n) {
45
46
            S--;
        }
47
        while ((s + 1) * (s + 1) <= n) {
48
            S++;
49
50
        return s;
51
    }
52
53
54
    /**
          精确开平方
55
     *
          2023-09-19: https://qoj.ac/submission/183430
    **/
56
57
    i64 get(i64 n) {
        i64 u = std::sqrt(2.0L * n);
58
        while (u * (u + 1) / 2 < n) {
59
            u++;
60
        }
61
        while (u * (u - 1) / 2 + 1 > n) {
62
63
            u--;
        }
64
        return u;
65
    }
66
```

## 5.3 03 - 字符调整

Listing 29: others/03-Char.cpp

```
/**
          大小写转换、获取字母序
1
    *
          2024-03-16: https://qoj.ac/submission/355156
2
    **/
3
    void rev(std::string &s) {
4
        int l = s.size();
5
        for (int i = 1; i < l; i += 2) {
6
            if (std::isupper(s[i])) {
7
8
                s[i] = std::tolower(s[i]);
9
            } else {
10
                s[i] = std::toupper(s[i]);
11
        }
12
    }
13
14
15
    int get(char c) {
16
        int x;
```

## 5.4 04A - 二分算法(整数域)

### Listing 30: others/04A-Binary-Search.cpp

```
/**
          二分算法(整数域):
 1
          2023-09-18: https://qoj.ac/submission/182628
 2
     *
    **/
 3
    int lo = 1, hi = 1E9;
 4
    while (lo < hi) {
 5
        int m = (lo + hi + 1) / 2;
 6
        if (check(m)) {
 7
 8
            lo = m;
        } else {
 9
            hi = m - 1;
10
11
12
    }
    std::cout << lo << "\n";
13
14
          二分算法(整数域):后继
    /**
15
          2023-09-18: https://qoj.ac/submission/182752
     *
16
    **/
17
    int lo = 1, hi = n;
18
    while (lo < hi) {
19
20
        int m = (lo + hi) / 2;
        if (check(m)) {
21
            hi = m;
22
23
        } else {
24
            lo = m + 1;
25
26
   std::cout << lo << "\n";
```

## 5.5 04B - 二分算法(实数域)

Listing 31: others/04B-Binary-Search.cpp

```
/**
          二分算法 (实数域)
 1
          2023—10—21: https://qoj.ac/submission/222042
 2
    **/
 3
    auto check = [&](double t) {
 4
        // write
 5
    };
 6
 7
    double lo = 0;
 8
 9
    double hi = 1E12;
10
    while (hi - lo > std: max(1.0, lo) * eps) {
        double x = (lo + hi) / 2;
11
        if (check(x)) {
12
            hi = x;
13
        } else {
14
            lo = x;
15
16
17
    }
18
19
    std::cout << lo << "\n";
20
```

```
/**
          二分算法 (实数域)
21
     *
          2023-09-15: https://qoj.ac/submission/179994
22
    **/
23
    using i64 = long long;
24
    using real = long double;
25
26
    constexpr real eps = 1E-7;
27
28
    auto get = [&](const auto &f) {
29
        real lo = -1E4, hi = 1E4;
30
        while (hi - lo > 3 * eps) {
31
            real x1 = (lo + hi - eps) / 2;
32
            real x2 = (lo + hi + eps) / 2;
33
            if (f(x1) > f(x2)) {
34
                 lo = x1;
35
            } else {
36
37
                 hi = x2;
38
        }
39
        return f((lo + hi) / 2);
40
    };
41
42
    std::cout << get([&](real px) {</pre>
43
        return get([&](real py) {
44
            // write
45
        });
46
    }) << "\n";</pre>
```

## 5.6 01 - 强连通分量缩点(SCC)

### Listing 32: graph/01-SCC.cpp

```
/**
          强连通分量缩点 (SCC)
1
          2023-06-18: https://codeforces.com/contest/1835/submission/210147209
2
     *
    **/
3
    struct SCC {
4
        int n;
5
6
        std::vector<std::vector<int>> adj;
        std::vector<int> stk;
7
8
        std::vectorint> dfn, low, bel;
9
        int cur, cnt;
10
        SCC() {}
11
        SCC(int n) {
12
            init(n);
13
14
15
        void init(int n) {
16
17
            this\rightarrown = n;
            adj.assign(n, {});
18
            dfn.assign(n, -1);
19
            low.resize(n);
20
            bel.assign(n, -1);
21
22
            stk.clear();
23
            cur = cnt = 0;
        }
24
25
        void addEdge(int u, int v) {
26
            adj[u].push back(v);
27
28
29
        void dfs(int x) {
30
            dfn[x] = low[x] = cur++;
31
```

```
stk.push_back(x);
32
33
             for (auto y : adj[x]) {
34
                 if (dfn[y] == -1) {
35
                     dfs(y);
36
37
                     low[x] = std::min(low[x], low[y]);
                 } else if (bel[y] == -1) {
38
                     low[x] = std::min(low[x], dfn[y]);
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();
48
                 } while (y != x);
49
50
                 cnt++;
             }
51
        }
52
53
        std::vector<int> work() {
54
             for (int i = 0; i < n; i++) {
55
                 if (dfn[i] == -1) {
56
                     dfs(i);
57
58
59
             return bel;
60
        }
61
    };
```

# 5.7 02 - 割边与割边缩点(EBCC)

### Listing 33: graph/02-EBCC.cpp

```
割边与割边缩点(EBCC)
    /**
 1
          2023-05-11: https://codeforces.com/contest/118/submission/205426518
 2
     *
    **/
 3
    std::set<std::pair<int, int>> E;
 4
 5
    struct EBCC {
 6
 7
        int n;
        std::vector<std::vector<int>> adj;
 8
        std::vector<int> stk;
 9
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
            dfn.assign(n, -1);
21
            low.resize(n);
22
            bel.assign(n, -1);
23
            stk.clear();
24
            cur = cnt = 0;
25
        }
26
27
```

```
void addEdge(int u, int v) {
28
             adj[u].push_back(v);
29
             adj[v].push_back(u);
30
        }
31
32
        void dfs(int x, int p) {
33
             dfn[x] = low[x] = cur++;
34
             stk.push_back(x);
35
36
             for (auto y : adj[x]) {
37
38
                 if (y == p) {
39
                     continue;
40
                 if (dfn[y] == -1) {
41
                     E.emplace(x, y);
42
                     dfs(y, x);
43
                     low[x] = std::min(low[x], low[y]);
44
                 \} else if (bel[y] == -1 && dfn[y] < dfn[x]) {
45
                     E.emplace(x, y);
46
                     low[x] = std::min(low[x], dfn[y]);
47
                 }
48
             }
49
50
             if (dfn[x] == low[x]) {
51
                 int y;
52
                 do {
53
                     y = stk.back();
54
55
                     bel[y] = cnt;
56
                     stk.pop_back();
                 } while (y != x);
57
                 cnt++;
58
59
             }
        }
60
61
62
        std::vector<int> work() {
             dfs(0, -1);
63
             return bel;
64
65
66
67
        struct Graph {
68
             int n;
             std::vector<std::pair<int, int>> edges;
69
70
             std::vector<int> siz;
             std::vector<int> cnte;
71
        };
72
        Graph compress() {
73
            Graph g;
74
             g.n = cnt;
75
             g.siz.resize(cnt);
76
             g.cnte.resize(cnt);
77
             for (int i = 0; i < n; i++) {
78
                 g.siz[bel[i]]++;
79
                 for (auto j : adj[i]) {
80
                     if (bel[i] < bel[j]) {</pre>
81
82
                         g.edges.emplace_back(bel[i], bel[j]);
                     } else if (i < j) {
83
                         g.cnte[bel[i]]++;
84
                     }
85
86
                 }
87
88
             return g;
89
    };
90
```

## 5.8 03 - 二分图最大权匹配(MaxAssignment 基于 KM)

### Listing 34: graph/03-Max-Assignment.cpp

```
二分图最大权匹配 (MaxAssignment 基于KM)
    /**
1
          2022-04-10: https://atcoder.jp/contests/abc247/submissions/30867023
     *
2
3
     *
          2023-09-21: https://qoj.ac/submission/184824
4
    **/
    constexpr int inf = 1E7;
5
    template<class T>
6
    struct MaxAssignment {
7
        public:
8
            T solve(int nx, int ny, std::vector<std::vector<T>> a) {
9
                assert(0 \le nx \& nx \le ny);
10
                assert(int(a.size()) == nx);
11
                for (int i = 0; i < nx; ++i) {
12
                    assert(int(a[i].size()) == ny);
13
                    for (auto x : a[i])
14
                         assert(x >= 0);
15
                }
16
17
                auto update = [&](int x) {
18
19
                    for (int y = 0; y < ny; ++y) {
20
                         if (lx[x] + ly[y] - a[x][y] < slack[y]) {
21
                             slack[y] = lx[x] + ly[y] - a[x][y];
22
                             slackx[y] = x;
                         }
23
                     }
24
                };
25
26
                costs.resize(nx + 1);
27
                costs[0] = 0;
28
                lx.assign(nx, std::numeric limits<T>::max());
29
                ly.assign(ny, 0);
30
                xy.assign(nx, -1);
31
                yx.assign(ny, -1);
32
                slackx.resize(ny);
33
34
                for (int cur = 0; cur < nx; ++cur) {
35
                    std::queue<int> que;
                    visx.assign(nx, false);
36
                    visy.assign(ny, false);
37
                    slack.assign(ny, std::numeric_limits<T>::max());
38
                    p.assign(nx, -1);
39
40
                    for (int x = 0; x < nx; ++x) {
41
                        if (xy[x] == -1) {
42
                             que.push(x);
43
                             visx[x] = true;
44
                             update(x);
45
                         }
46
                    }
47
48
                    int ex, ey;
49
                    bool found = false;
50
                    while (!found) {
51
                        while (!que.empty() && !found) {
52
                             auto x = que.front();
53
                             que.pop();
54
                             for (int y = 0; y < ny; ++y) {
55
                                 if (a[x][y] == lx[x] + ly[y] && !visy[y]) {
56
                                     if (yx[y] == -1) {
57
58
                                         ex = x;
59
                                         ey = y;
60
                                         found = true;
```

```
61
                                           break;
 62
                                       que.push(yx[y]);
63
                                       p[yx[y]] = x;
 64
                                       visy[y] = visx[yx[y]] = true;
 65
                                       update(yx[y]);
66
                                   }
67
                              }
 68
                          }
 69
                          if (found)
 70
                              break;
 71
 72
                          T delta = std::numeric limits<T>::max();
 73
                          for (int y = 0; y < ny; ++y)
 74
                              if (!visy[y])
 75
                                  delta = std::min(delta, slack[y]);
 76
                          for (int x = 0; x < nx; ++x)
 77
                              if (visx[x])
 78
                                  lx[x] = delta;
 79
                          for (int y = 0; y < ny; ++y) {
80
                              if (visy[y]) {
 81
                                  ly[y] += delta;
 82
                              } else {
83
                                  slack[y] -= delta;
84
85
                          }
86
                          for (int y = 0; y < ny; ++y) {
87
                              if (!visy[y] \& slack[y] == 0) {
 88
                                  if (yx[y] == -1) {
89
                                       ex = slackx[y];
90
                                       ey = y;
 91
                                       found = true;
 92
 93
                                       break;
                                   }
94
                                  que.push(yx[y]);
 95
 96
                                  p[yx[y]] = slackx[y];
 97
                                  visy[y] = visx[yx[y]] = true;
                                  update(yx[y]);
98
                              }
99
                          }
100
                      }
101
102
                      costs[cur + 1] = costs[cur];
103
                      for (int x = ex, y = ey, ty; x != -1; x = p[x], y = ty) {
104
                          costs[cur + 1] += a[x][y];
105
                          if (xy[x] != -1)
106
                              costs[cur + 1] = a[x][xy[x]];
107
108
                          ty = xy[x];
                          xy[x] = y;
109
                          yx[y] = x;
110
                      }
111
                  }
112
                 return costs[nx];
113
             }
114
             std::vector<int> assignment() {
115
                 return xy;
116
117
             std::pair<std::vector<T>, std::vector<T>> labels() {
118
                 return std::make_pair(lx, ly);
119
120
             std::vector<T> weights() {
121
                 return costs;
123
124
         private:
```

```
std::vector<T> lx, ly, slack, costs;
std::vector<int> xy, yx, p, slackx;
std::vector<bool> visx, visy;
std::vector<bool>
```

## 5.9 04 - 一般图最大匹配(Graph 带花树算法)【久远】

## Listing 35: graph/04-Graph-Match.cpp

```
/**
          一般图最大匹配(Graph 带花树算法)【久远】
 1
          2021-12-24: https://codeforces.com/contest/1615/submission/140509278
 2
    **/
 3
    struct Graph {
 4
        int n;
 5
        std::vector<std::vector<int>> e;
 6
        Graph(int n) : n(n), e(n) {}
 7
        void addEdge(int u, int v) {
 8
            e[u].push_back(v);
 9
            e[v].push_back(u);
10
        }
11
        std::vector<int> findMatching() {
12
            std::vectorkint> match(n, -1), vis(n), link(n), f(n), dep(n);
13
14
            // disjoint set union
15
            auto find = [&](int u) {
16
                while (f[u] != u)
17
                    u = f[u] = f[f[u]];
18
                return u;
19
            };
20
21
            auto lca = [\delta](int u, int v) {
22
23
                u = find(u);
                v = find(v);
24
                while (u != v) {
25
                     if (dep[u] < dep[v])
26
27
                         std::swap(u, v);
                    u = find(link[match[u]]);
28
29
30
                return u;
            };
31
32
            std::queu≪int> que;
33
34
            auto blossom = [\delta](int u, int v, int p) {
                while (find(u) != p) {
35
36
                    link[u] = v;
                     v = match[u];
37
                     if (vis[v] == 0) {
38
                         vis[v] = 1;
39
                         que.push(v);
40
41
                    f[u] = f[v] = p;
42
                    u = link[v];
43
                 }
44
            };
45
46
            // find an augmenting path starting from u and augment (if exist)
47
            auto augment = [&](int u) {
48
49
                while (!que.empty())
50
                    que.pop();
51
52
                std::iota(f.begin(), f.end(), 0);
53
54
```

```
// vis = 0 corresponds to inner vertices, vis = 1 corresponds to outer vertices
 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()){
62
                      int u = que.front();
63
 64
                      que.pop();
 65
                      for (auto v : e[u]) {
                          if (vis[v] == -1) {
66
67
                              vis[v] = 0;
68
 69
                              link[v] = u;
 70
                              dep[v] = dep[u] + 1;
 71
                              // found an augmenting path
 72
 73
                              if (match[v] == -1) {
                                   for (int x = v, y = u, temp; y != -1; x = temp, y = x == -1 ? -1 : link[x]) {
 74
                                       temp = match[y];
 75
                                      match[x] = y;
 76
 77
                                       match[y] = x;
                                   }
 78
 79
                                  return;
                              }
 80
81
                              vis[match[v]] = 1;
82
                              dep[match[v]] = dep[u] + 2;
83
                              que.push(match[v]);
84
85
                          } else if (vis[v] == 1 \& find(v) != find(u)) {
 86
                              // found a blossom
 87
88
                              int p = lca(u, v);
                              blossom(u, v, p);
89
                              blossom(v, u, p);
90
                          }
91
                      }
92
                  }
93
94
 95
             };
 96
97
             // find a maximal matching greedily (decrease constant)
             auto greedy = [\delta]() {
98
99
                 for (int u = 0; u < n; ++u) {
100
                      if (match[u] != -1)
101
102
                          continue;
                      for (auto v : e[u]) {
103
                          if (match[v] == -1) {
104
                              match[u] = v;
105
                              match[v] = u;
106
                              break;
107
                          }
108
                      }
109
                  }
110
             };
111
112
             greedy();
113
114
             for (int u = 0; u < n; ++u)
115
                 if (match[u] == -1)
116
                      augment(u);
117
118
```

```
119          return match;
120      }
121      };
```

### 5.10 05 - TwoSat (2-Sat)

### Listing 36: graph/05-Two-Sat.cpp

```
/**
          TwoSat (2—Sat)
1
          2023-09-29: https://atcoder.jp/contests/arc161/submissions/46031530
2
    **/
3
4
    struct TwoSat {
5
        int n;
        std::vector<std::vector<int>> e;
6
        std::vector<bool> ans;
7
8
        TwoSat(int n) : n(n), e(2 * n), ans(n) {}
        void addClause(int u, bool f, int v, bool g) {
9
            e[2 * u + !f].push_back(2 * v + g);
10
            e[2 * v + !g].push_back(2 * u + f);
11
        }
12
13
        bool satisfiable() {
            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::functionvoid(int)> tarjan = [&](int u) {
17
                stk.push_back(u);
18
                dfn[u] = low[u] = now++;
19
                for (auto v : e[u]) {
20
21
                     if (dfn[v] == -1) {
22
                        tarjan(v);
                        low[u] = std::min(low[u], low[v]);
23
                    } else if (id[v] == -1) {
24
25
                        low[u] = std::min(low[u], dfn[v]);
26
                }
27
28
                if (dfn[u] == low[u]) {
                    int v;
29
                    do {
30
31
                         v = stk.back();
                        stk.pop_back();
32
                         id[v] = cnt;
33
                    } while (v != u);
34
                    ++cnt;
35
                }
36
37
            };
            for (int i = 0; i < 2 * n; ++i) if (dfn[i] == -1) tarjan(i);
38
            for (int i = 0; i < n; ++i) {
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
        std::vector<bool> answer() { return ans; }
45
   };
46
```

### 5.11 06A - 最大流(Flow 旧版其一,整数应用)

### Listing 37: graph/06A-Max-Flow.cpp

```
1 /** 最大流(Flow 旧版其一,整数应用)
2 * 2022-09-03: https://codeforces.com/contest/1717/submission/170688062
3 **/
4 template<class T>
```

```
struct Flow {
 5
        const int n;
 6
 7
        struct Edge {
            int to;
 8
            T cap;
 9
            Edge(int to, T cap) : to(to), cap(cap) {}
10
        };
11
        std::vector<Edge> e;
12
        std::vector<std::vector<int>> g;
13
        std::vector<int> cur, h;
14
        Flow(int n): n(n), g(n) {}
15
16
        bool bfs(int s, int t) {
17
            h.assign(n, -1);
18
            std::queu≪int> que;
19
20
            h[s] = 0;
            que.push(s);
21
            while (!que.empty()) {
22
                const int u = que.front();
23
24
                que.pop();
                 for (int i : g[u]) {
25
26
                     auto [v, c] = e[i];
                     if (c > 0 \& h[v] == -1) {
27
                         h[v] = h[u] + 1;
28
                         if(v == t) {
29
                             return true;
30
31
                         que.push(v);
32
                     }
33
                 }
34
35
36
            return false;
37
38
        T dfs(int u, int t, T f) {
39
            if (u == t) {
40
                return f;
41
42
            auto r = f;
43
            for (int &i = cur[u]; i < int(g[u].size()); ++i) {
44
                const int j = g[u][i];
45
                auto [v, c] = e[j];
46
                 if (c > 0 \& h[v] == h[u] + 1) {
47
                     auto a = dfs(v, t, std::min(r, c));
48
                     e[j].cap = a;
49
                     e[j ^1].cap += a;
50
                     r -= a;
51
                     if (r == 0) {
52
                         return f;
53
                     }
54
                 }
55
56
            return f - r;
57
58
        void addEdge(int u, int v, T c) {
59
            g[u].push_back(e.size());
60
            e.emplace_back(v, c);
61
            g[v].push_back(e.size());
62
            e.emplace_back(u, 0);
63
64
        T maxFlow(int s, int t) {
65
            T ans = 0;
66
            while (bfs(s, t)) {
67
                cur.assign(n, 0);
68
```

## 5.12 06B - 最大流 (Flow 旧版其二, 浮点数应用)

### Listing 38: graph/06B-Max-Flow.cpp

```
/**
          最大流(Flow 旧版其二,浮点数应用)
 1
          2022-04-09: https://cf.dianhsu.com/gym/104288/submission/201412765
 2
    **/
 3
    template<class T>
 4
    struct Flow {
 5
        const int n;
 6
        struct Edge {
 7
            int to;
 8
 9
            T cap;
            Edge(int to, T cap) : to(to), cap(cap) {}
10
        };
11
        std::vector<Edge> e;
12
        std::vector<std::vector<int>> g;
13
        std::vector<int> cur, h;
14
        Flow(int n): n(n), g(n) {}
15
16
        bool bfs(int s, int t) {
17
            h.assign(n, -1);
18
            std::queue<int> que;
19
            h[s] = 0;
20
            que.push(s);
21
            while (!que.empty()) {
22
                const int u = que.front();
23
24
                que.pop();
                for (int i : g[u]) {
25
                    auto [v, c] = e[i];
26
                    if (c > 0 \& h[v] == -1) {
27
                         h[v] = h[u] + 1;
28
                         if (v == t) {
29
                             return true;
30
                         }
31
                         que.push(v);
32
                     }
33
                }
34
35
            return false;
36
        }
37
38
        T dfs(int u, int t, T f) {
39
            if (u == t) {
40
                return f;
41
42
            auto r = f;
43
            double res = 0;
44
            for (int &i = cur[u]; i < int(g[u].size()); ++i) {</pre>
45
                const int j = g[u][i];
46
                auto [v, c] = e[j];
47
                if (c > 0 \& h[v] == h[u] + 1) {
48
                    auto a = dfs(v, t, std::min(r, c));
49
                    res += a;
50
                    e[j].cap = a;
51
                    e[j ^ 1].cap += a;
52
53
                    r -= a;
54
                    if (r == 0) {
```

```
55
                         return f;
                     }
56
                 }
57
             }
58
             return res;
59
60
        void addEdge(int u, int v, T c) {
61
62
             g[u].push_back(e.size());
             e.emplace_back(v, c);
63
             g[v].push_back(e.size());
64
             e.emplace_back(u, 0);
65
        }
66
        T maxFlow(int s, int t) {
67
68
             T ans = 0;
69
            while (bfs(s, t)) {
                 cur.assign(n, 0);
70
                 ans += dfs(s, t, 1E100);
71
72
             return ans;
73
74
        }
75
    };
```

# 5.13 06C - 最大流 (MaxFlow 新版)

### Listing 39: graph/06C-Max-Flow.cpp

```
/**
          最大流 (MaxFlow 新版)
 1
 2
     *
          2023-07-21: https://ac.nowcoder.com/acm/contest/view-submission?submissionId=62915815
 3
    **/
 4
    constexpr int inf = 1E9;
    template<class T>
 5
    struct MaxFlow {
 6
        struct _Edge {
 7
            int to;
 8
 9
            T cap;
             _Edge(int to, T cap) : to(to), cap(cap) {}
10
        };
11
12
        int n;
13
        std::vector< Edge> e;
14
15
        std::vector<std::vector<int>> g;
16
        std::vector<int> cur, h;
17
        MaxFlow() {}
18
        MaxFlow(int n) {
19
            init(n);
20
21
22
        void init(int n) {
23
24
            this\rightarrown = n;
25
            e.clear();
            g.assign(n, {});
26
            cur.resize(n);
27
            h.resize(n);
28
        }
29
30
        bool bfs(int s, int t) {
31
32
            h.assign(n, -1);
33
            std::queu≪int> que;
34
            h[s] = 0;
            que.push(s);
35
            while (!que.empty()) {
36
                const int u = que.front();
37
```

```
38
                 que.pop();
                 for (int i : g[u]) {
 39
                      auto [v, c] = e[i];
 40
                      if (c > 0 \& h[v] == -1) {
 41
                          h[v] = h[u] + 1;
42
                          if (v == t) {
 43
                              return true;
 44
                          }
 45
                          que.push(v);
 46
                      }
 47
                  }
 48
             }
 49
             return false;
50
         }
51
52
         T dfs(int u, int t, T f) {
53
             if (u == t) {
54
                 return f;
55
56
             auto r = f;
57
             for (int &i = cur[u]; i < int(g[u].size()); ++i) {
58
                 const int j = g[u][i];
59
                 auto [v, c] = e[j];
60
                 if (c > 0 \& h[v] == h[u] + 1) {
61
                      auto a = dfs(v, t, std::min(r, c));
 62
                      e[j].cap = a;
 63
                      e[j ^ 1].cap += a;
 64
                      r -= a;
 65
66
                      if (r == 0) {
67
                          return f;
 68
                  }
 69
             }
 70
             return f - r;
 71
         }
 72
         void addEdge(int u, int v, T c) {
 73
 74
             g[u].push_back(e.size());
             e.emplace_back(v, c);
 75
             g[v].push_back(e.size());
76
 77
             e.emplace_back(u, 0);
         }
78
 79
         T flow(int s, int t) {
             T ans = 0;
 80
             while (bfs(s, t)) {
81
                 cur.assign(n, 0);
82
                 ans += dfs(s, t, std::numeric_limits<T>::max());
83
84
             return ans;
85
         }
86
87
         std::vector<bool> minCut() {
88
             std::vector<bool> c(n);
 89
             for (int i = 0; i < n; i++) {
90
                  c[i] = (h[i] != -1);
91
92
             return c;
93
94
95
         struct Edge {
96
97
             int from;
             int to;
98
             T cap;
99
             T flow;
100
         };
101
```

```
std::vector<Edge> edges() {
102
103
             std::vector<Edge> a;
             for (int i = 0; i < e.size(); i += 2) {
104
                 Edge x;
105
                 x.from = e[i + 1].to;
106
                 x.to = e[i].to;
107
                 x.cap = e[i].cap + e[i + 1].cap;
108
                 x.flow = e[i + 1].cap;
109
110
                 a.push_back(x);
111
             return a;
112
113
    };
114
```

### 5.14 07A - 费用流(MCFGraph 旧版)

### Listing 40: graph/07A-Min-Cost-Flow.cpp

```
费用流(MCFGraph 旧版)
1
2
          2022-12-12: https://codeforces.com/contest/1766/submission/184974697
3
    *
          下方为最小费用**最大流**模板,如需求解最小费用**可行流**,需要去除建边限制
4
    *
    **/
5
   struct MCFGraph {
6
        struct Edge {
7
8
            int v, c, f;
            Edge(int v, int c, int f) : v(v), c(c), f(f) {}
9
        };
10
        const int n;
11
        std::vector<Edge> e;
12
        std::vector<std::vector<int>> g;
13
        std::vector<i64> h, dis;
14
15
        std::vector<int> pre;
        bool dijkstra(int s, int t) {
16
            dis.assign(n, std::numeric_limit≤i64>::max());
17
18
            pre.assign(n, -1);
            std::priority_queuestd::paixi64, int>, std::vectorstd::paixi64, int>>, std::greaterstd::paix
19
              i64, int>>> que;
            dis[s] = 0;
20
21
            que.emplace(0, s);
            while (!que.empty()) {
22
                i64 d = que.top().first;
23
                int u = que.top().second;
24
                que.pop();
25
                if (dis[u] < d) continue;</pre>
26
                for (int i : g[u]) {
27
                    int v = e[i].v;
28
                    int c = e[i].c:
29
                    int f = e[i].f;
30
                    if (c > 0 \& dis[v] > d + h[u] - h[v] + f) {
31
                        dis[v] = d + h[u] - h[v] + f;
32
                        pre[v] = i;
33
                        que.emplace(dis[v], v);
34
                    }
35
                }
36
37
            return dis[t] != std::numeric_limits<i64>::max();
38
39
       MCFGraph(int n) : n(n), g(n) {}
40
        void addEdge(int u, int v, int c, int f) {
41
            // if (f < 0) {
42
                g[u].push_back(e.size());
43
                e.emplace_back(v, 0, f);
44
45
                g[v].push_back(e.size());
```

```
e.emplace_back(u, c, -f);
46
            // } else {
47
            //
                    g[u].push back(e.size());
48
            //
                    e.emplace_back(v, c, f);
49
            //
                    g[v].push_back(e.size());
50
            //
                    e.emplace_back(u, 0, -f);
51
            // }
52
        }
53
54
        std::pair<int, i64> flow(int s, int t) {
55
            int flow = 0;
56
            i64 cost = 0;
57
            h.assign(n, 0);
            while (dijkstra(s, t)) {
58
                 for (int i = 0; i < n; ++i) h[i] += dis[i];
59
                 int aug = std::numeric_limits<int>::max();
60
                 for (int i = t; i != s; i = e[pre[i] ^ 1].v) aug = std::min(aug, e[pre[i]].c);
61
                 for (int i = t; i != s; i = e[pre[i] ^ 1].v) {
62
                     e[pre[i]].c = aug;
63
                     e[pre[i] ^ 1].c += aug;
64
65
                flow += aug;
66
                cost += i64(aug) * h[t];
67
68
            return std::make_pair(flow, cost);
69
        }
70
    };
71
```

### 5.15 07B - 费用流(MinCostFlow 新版)

#### Listing 41: graph/07B-Min-Cost-Flow.cpp

```
/**
          MinCostFlow 新版
1
2
          2023-11-09: https://goj.ac/submission/244680
3
    **/
4
    template<class T>
5
    struct MinCostFlow {
        struct _Edge {
6
            int to;
7
            T cap;
8
            T cost;
9
            _Edge(int to_, T cap_, T cost_) : to(to_), cap(cap_), cost(cost_) {}
10
        };
11
        int n;
12
        std::vector<_Edge> e;
13
        std::vector<std::vector<int>> g;
14
        std::vector<T> h, dis;
15
        std::vector<int> pre;
16
        bool dijkstra(int s, int t) {
17
            dis.assign(n, std::numeric_limits<T>::max());
18
            pre.assign(n, -1);
19
            std::priority_queue<std::pair<T, int>, std::vector<std::pair<T, int>>, std::greater<std::pair<T,
20
              int>>> que;
            dis[s] = 0;
21
            que.emplace(0, s);
22
            while (!que.empty()) {
23
                T d = que.top().first;
24
                int u = que.top().second;
25
                que.pop();
26
                if (dis[u] != d) {
27
                    continue;
28
29
                for (int i : g[u]) {
30
                    int v = e[i].to;
31
```

```
T cap = e[i].cap;
32
                     T cost = e[i].cost;
33
                     if (cap > 0 \& dis[v] > d + h[u] - h[v] + cost) {
34
35
                         dis[v] = d + h[u] - h[v] + cost;
36
                         pre[v] = i;
                         que.emplace(dis[v], v);
37
                     }
38
                 }
39
            }
40
            return dis[t] != std::numeric limits<T>::max();
41
42
        MinCostFlow() {}
43
        MinCostFlow(int n_) {
44
            init(n_);
45
46
47
        void init(int n_) {
48
            n = n_{;}
49
            e.clear();
            g.assign(n, {});
50
51
        void addEdge(int u, int v, T cap, T cost) {
52
            g[u].push_back(e.size());
53
            e.emplace back(v, cap, cost);
54
            g[v].push back(e.size());
55
            e.emplace_back(u, 0, -cost);
56
        }
57
        std::pair<T, T> flow(int s, int t) {
58
59
            T flow = 0;
            T cost = 0;
60
            h.assign(n, 0);
61
            while (dijkstra(s, t)) {
62
                 for (int i = 0; i < n; ++i) {
63
                     h[i] += dis[i];
64
65
                T aug = std::numeric_limits<int>::max();
66
                for (int i = t; i != s; i = e[pre[i] ^ 1].to) {
67
                     aug = std::min(aug, e[pre[i]].cap);
68
69
                for (int i = t; i != s; i = e[pre[i] ^ 1].to) {
70
                     e[pre[i]].cap = aug;
71
                     e[pre[i] ^ 1].cap += aug;
72
73
                flow += aug;
74
                cost += aug * h[t];
75
76
            return std::make_pair(flow, cost);
77
        }
78
        struct Edge {
79
            int from;
80
81
            int to;
82
            T cap;
83
            T cost;
            T flow;
84
        };
85
        std::vector<Edge> edges() {
86
87
            std::vector<Edge> a;
88
            for (int i = 0; i < e.size(); i += 2) {
89
                Edge x;
                x.from = e[i + 1].to;
90
                x.to = e[i].to;
91
                x.cap = e[i].cap + e[i + 1].cap;
92
                x.cost = e[i].cost;
93
                x.flow = e[i + 1].cap;
94
                a.push_back(x);
95
```

```
96
97
             return a;
98
    };
99
```

#### 08 - 树链剖分(HLD) 5.16

```
Listing 42: graph/08-HLD.cpp
    /**
          树链剖分 (HLD)
 1
     *
          2023-08-31: https://codeforces.com/contest/1863/submission/221214363
 2
    **/
 3
    struct HLD {
 4
        int n;
 5
        std::vector<int> siz, top, dep, parent, in, out, seq;
 6
        std::vector<std::vector<int>> adj;
 7
 8
        int cur;
 9
        HLD() \{ \}
10
        HLD(int n) {
11
            init(n);
12
        }
13
        void init(int n) {
14
            this\rightarrown = n;
15
            siz.resize(n);
16
17
            top.resize(n);
            dep.resize(n);
18
            parent.resize(n);
19
20
            in.resize(n);
            out.resize(n);
21
            seq.resize(n);
22
            cur = 0;
23
            adj.assign(n, {});
24
        }
25
        void addEdge(int u, int v) {
26
27
            adj[u].push_back(v);
            adj[v].push_back(u);
28
29
        void work(int root = 0) {
30
            top[root] = root;
31
            dep[root] = 0;
32
33
            parent[root] = -1;
            dfs1(root);
34
            dfs2(root);
35
36
        void dfs1(int u) {
37
38
            if (parent[u] != -1) {
                adj[u].erase(std::find(adj[u].begin(), adj[u].end(), parent[u]));
39
40
41
            siz[u] = 1;
42
            for (auto &v : adj[u]) {
43
                parent[v] = u;
44
                dep[v] = dep[u] + 1;
45
                dfs1(v);
46
                siz[u] += siz[v];
47
                if (siz[v] > siz[adj[u][0]]) {
48
                     std::swap(v, adj[u][0]);
49
                 }
50
            }
51
        }
52
        void dfs2(int u) {
53
            in[u] = cur++;
54
```

```
55
             seq[in[u]] = u;
             for (auto v : adj[u]) {
 56
                 top[v] = v == adj[u][0] ? top[u] : v;
57
                 dfs2(v);
58
 59
             out[u] = cur;
60
         }
61
         int lca(int u, int v) {
62
             while (top[u] != top[v]) {
63
                  if (dep[top[u]] > dep[top[v]]) {
 64
 65
                      u = parent[top[u]];
                  } else {
 66
                      v = parent[top[v]];
67
68
69
             return dep[u] < dep[v] ? u : v;
70
         }
 71
 72
73
         int dist(int u, int v) {
             return dep[u] + dep[v] - 2 * dep[lca(u, v)];
74
 75
 76
         int jump(int u, int k) {
 77
             if (dep[u] < k) {
 78
 79
                 return -1;
 80
81
             int d = dep[u] - k;
82
83
             while (dep[top[u]] > d) {
84
                  u = parent[top[u]];
85
86
87
             return seq[in[u] - dep[u] + d];
88
         }
89
90
         bool isAncester(int u, int v) {
91
             return in[u] <= in[v] && in[v] < out[u];</pre>
92
93
 94
         int rootedParent(int u, int v) {
95
             std::swap(u, v);
96
             if (u == v) {
 97
                 return u;
 98
 99
100
             if (!isAncester(u, v)) {
                 return parent[u];
101
102
             auto it = std::upper_bound(adj[u].begin(), adj[u].end(), v, [\delta](int x, int y) {
103
104
                 return in[x] < in[y];
             }) - 1;
105
             return *it;
106
         }
107
108
         int rootedSize(int u, int v) {
109
             if (u == v) {
110
                 return n;
111
112
             if (!isAncester(v, u)) {
113
                 return siz[v];
114
115
             return n - siz[rootedParent(u, v)];
116
         }
117
118
```

```
int rootedLca(int a, int b, int c) {
    return lca(a, b) ^ lca(b, c) ^ lca(c, a);
}

120
    return lca(a, b) ^ lca(b, c) ^ lca(c, a);
}
```

### 5.17 01 - 快速幂

### Listing 43: math/01-Power.cpp

```
/**
          快速幂 - 普通版
 1
     *
          2023—10—09: https://atcoder.jp/contests/tenka1—2017/submissions/46411797
 2
    **/
 3
    int power(int a, i64 b, int p) {
 4
 5
        int res = 1;
        for (; b; b /= 2, a = 1LL * a * a % p) {
 6
            if (b % 2) {
 7
                res = 1LL * res * a % p;
 8
 9
10
11
        return res;
    }
12
13
    /**
          快速幂 - 手写乘法
14
     *
          2023-09-27: https://qoj.ac/submission/189343
15
    **/
16
17
    using i64 = long long;
18
    i64 mul(i64 a, i64 b, i64 p) {
19
        i64 c = a * b - i64(1.0L * a * b / p) * p;
20
21
        c %= p;
        if (c < 0) {
22
23
            c += p;
        }
24
25
        return c;
    }
26
27
    i64 power(i64 a, i64 b, i64 p) {
28
        i64 \text{ res} = 1;
29
        for (; b; b /= 2, a = mul(a, a, p)) {
30
            if (b % 2) {
31
                res = mul(res, a, p);
32
33
34
35
        return res;
36
```

# 5.18 02 - 基姆拉尔森公式

#### Listing 44: math/02-Kim-Larsen.cpp

```
/**
          基姆拉尔森公式
 1
          2023-09-05: https://qoj.ac/submission/164735
     *
 2
    **/
 3
    const int d[] = {31, 28, 31, 30, 31, 30, 31, 30, 31, 30, 31};
 4
 5
    bool isLeap(int y) {
 6
        return y % 400 == 0 || (y % 4 == 0 && y % 100 != 0);
 7
    }
 8
 9
    int daysInMonth(int y, int m) {
10
        return d[m-1] + (isLeap(y) \&\& m == 2);
11
    }
12
13
```

```
int getDay(int y, int m, int d) {
        int ans = 0;
15
        for (int i = 1970; i < y; i++) {
16
            ans += 365 + isLeap(i);
17
18
        for (int i = 1; i < m; i++) {
19
            ans += daysInMonth(y, i);
20
21
        ans += d;
22
        return (ans + 2) \% 7 + 1;
23
    }
24
```

### 5.19 03 - 欧拉筛

### Listing 45: math/03-Euler-Sieve.cpp

```
/**
          欧拉筛
 1
          2023—11—14: https://qoj.ac/submission/251234
     *
 2
    **/
 3
    std::vector<int> minp, primes;
 4
 5
    void sieve(int n) {
 6
        minp.assign(n + 1, 0);
 7
 8
        primes.clear();
 9
        for (int i = 2; i \le n; i++) {
10
            if (minp[i] == 0) {
11
                minp[i] = i;
12
                primes.push_back(i);
13
            }
14
15
            for (auto p : primes) {
16
                 if (i * p > n) {
17
18
                     break;
                 }
19
                minp[i * p] = p;
20
                if (p == minp[i]) {
21
                     break;
22
23
            }
24
        }
25
    }
26
27
    bool isprime(int n) {
28
        return minp[n] == n;
29
    }
30
```

### 5.20 04 - 莫比乌斯函数筛(莫比乌斯反演)

#### Listing 46: math/04-Mu-Sieve.cpp

```
/**
         莫比乌斯函数筛(莫比乌斯函数/反演)
1
    *
         2023-03-04: https://atcoder.jp/contests/tupc2022/submissions/39391116
2
   **/
3
   std::unordered_mapxint, Z> fMu;
4
5
   constexpr int N = 1E7;
6
   std::vectorint> minp, primes;
7
   std::vector<Z> mu;
8
9
   void sieve(int n) {
10
       minp.assign(n + 1, 0);
11
12
       mu.resize(n);
```

```
primes.clear();
13
14
        mu[1] = 1;
15
        for (int i = 2; i \le n; i++) {
16
            if (minp[i] == 0) {
17
                 mu[i] = -1;
18
                 minp[i] = i;
19
                 primes.push_back(i);
20
            }
21
22
            for (auto p : primes) {
23
                 if (i * p > n) {
24
                     break;
25
                 }
26
                 minp[i * p] = p;
27
                 if (p == minp[i]) {
28
                     break;
29
30
                 mu[i * p] = -mu[i];
31
            }
32
        }
33
34
        for (int i = 1; i <= n; i++) {
35
            mu[i] += mu[i-1];
36
37
38
    }
39
    Z sumMu(int n) {
40
        if (n <= N) {
41
            return mu[n];
42
43
        if (fMu.count(n)) {
44
            return fMu[n];
45
46
        if (n == 0) {
47
            return 0;
48
49
        Z ans = 1;
50
        for (int l = 2, r; l <= n; l = r + 1) {
51
            r = n / (n / l);
52
53
            ans = (r - l + 1) * sumMu(n / l);
54
55
        return ans;
    }
56
```

# 5.21 05 - 扩展欧几里得(exgcd)

### Listing 47: math/05-Exgcd.cpp

```
/**
          扩展欧几里得 (exgcd)
 1
          2023-10-09: https://atcoder.jp/contests/tenka1-2017/submissions/46411797
    *
 2
    **/
 3
    int exgcd(int a, int b, int &x, int &y) {
 4
 5
        if (!b) {
 6
            x = 1, y = 0;
 7
            return a;
 8
        int g = exgcd(b, a \% b, y, x);
 9
        y = a / b * x;
10
        return g;
11
12
13
    /**
          扩展欧几里得 (exgcd)
14
          2023-09-05: https://qoj.ac/submission/165983
15
```

```
16 **/
17 std::array<i64, 3> exgcd(i64 a, i64 b) {
18     if (!b) {
19         return {a, 1, 0};
20     }
21     auto [g, x, y] = exgcd(b, a % b);
22     return {g, y, x - a / b * y};
23 }
```

# 5.22 06A - 欧拉函数 (求解单个数的欧拉函数)

#### Listing 48: math/06A-Phi.cpp

```
/**
          欧拉函数 (求解单个数的欧拉函数)
1
          2023-10-09: https://atcoder.jp/contests/tenka1-2017/submissions/46411797
2
    *
    **/
3
    int phi(int n) {
4
        int res = n;
5
        for (int i = 2; i * i <= n; i++) {
6
7
            if (n \% i == 0) {
8
                while (n \% i == 0) {
9
                    n = i;
                }
10
                res = res / i * (i - 1);
11
            }
12
        }
13
        if (n > 1) {
14
            res = res / n * (n - 1);
15
16
        return res;
17
    }
18
```

### 5.23 06B - 欧拉函数(求解全部数的欧拉函数)

#### Listing 49: math/06B-Phi-Sieve.cpp

```
/**
          欧拉函数 (求解全部数的欧拉函数)
 1
     *
          2023-09-24: https://qoj.ac/submission/187055
 2
     **/
 3
    constexpr int N = 1E7;
 4
    constexpr int P = 1000003;
 5
 6
    bool isprime[N + 1];
 7
    int phi[N + 1];
 8
    std::vector<int> primes;
 9
10
    void sieve(void) {
11
        std::fill(isprime + 2, isprime + N + 1, true);
12
        phi[1] = 1;
1.3
        for (int i = 2; i \le N; i++) {
14
            if (isprime[i]) {
15
                primes.push_back(i);
16
                phi[i] = i - 1;
17
18
            for (auto p : primes) {
19
                if (i * p > N) {
20
                    break;
21
22
                isprime[i * p] = false;
23
                if (i \% p == 0) {
24
                    phi[i * p] = phi[i] * p;
25
                    break;
26
                }
27
```

### 5.24 07A - 组合数(小范围预处理, 逆元 + 杨辉三角)

### Listing 50: math/07A-Comb.cpp

```
/**
          组合数(小范围预处理,逆元+杨辉三角)
 1
          2024-03-14: https://qoj.ac/submission/353877
 2
          2023—10—06: https://qoj.ac/submission/203196
     *
 3
    **/
 4
    constexpr int P = 1000000007;
 5
    constexpr int L = 10000;
 6
 7
    int fac[L + 1], invfac[L + 1];
 8
 9
    int sumbinom[L + 1][7];
10
    int binom(int n, int m) {
11
        if (n < m \mid | m < 0) {
12
            return 0;
13
14
        return 1LL * fac[n] * invfac[m] % P * invfac[n - m] % P;
15
16
    }
17
    int power(int a, int b) {
18
        int res = 1;
19
        for (; b; b /= 2, a = 1LL * a * a % P) {
20
            if (b % 2) {
21
                res = 1LL * res * a % P;
22
23
24
        return res;
25
    }
26
27
    int main() {
28
        fac[0] = 1;
29
        for (int i = 1; i <= L; i++) {
30
            fac[i] = 1LL * fac[i - 1] * i % P;
31
32
        invfac[L] = power(fac[L], P - 2);
33
        for (int i = L; i; i—) {
34
            invfac[i-1] = 1LL * invfac[i] * i % P;
35
36
37
        sumbinom[0][0] = 1;
38
        for (int i = 1; i \le L; i++) {
39
            for (int j = 0; j < 7; j++) {
40
                sumbinom[i][j] = (sumbinom[i-1][j] + sumbinom[i-1][(j+6) % 7]) % P;
41
42
        }
43
    }
44
```

### 5.25 07B - 组合数 (Comb, with. ModIntBase)

#### Listing 51: math/07B-Comb.cpp

```
1 /** 组合数 (Comb, with. MInt & MLong)
2 * 2023—08—26: https://codeforces.com/contest/1864/submission/220584872
3 **/
4 struct Comb {
5 int n;
6 std::vector<Z> _fac;
```

```
7
        std::vector<Z> _invfac;
 8
        std::vector<Z> _inv;
 9
        Comb(): n{0}, _fac{1}, _invfac{1}, _inv{0} {}
10
        Comb(int n) : Comb() {
11
             init(n);
12
13
14
        void init(int m) {
15
             m = std::min(m, Z::getMod() - 1);
16
17
             if (m <= n) return;
            _fac.resize(m + 1);
18
             _invfac.resize(m + 1);
19
             _{inv.resize(m + 1);}
20
21
             for (int i = n + 1; i \le m; i++) {
22
23
                 _{fac[i]} = _{fac[i-1]} * i;
24
             _invfac[m] = _fac[m].inv();
25
             for (int i = m; i > n; i—) {
26
                 _{invfac[i-1] = _{invfac[i] * i;}
27
                 _{inv[i]} = _{invfac[i]} * _{fac[i-1]};
28
29
             n = m;
30
        }
31
32
        Z fac(int m) {
33
             if (m > n) init(2 * m);
34
             return _fac[m];
35
36
37
        Z invfac(int m) {
             if (m > n) init(2 * m);
38
             return _invfac[m];
39
40
        Z inv(int m) {
41
             if (m > n) init(2 * m);
42
             return inv[m];
43
44
        Z binom(int n, int m) {
45
             if (n < m \mid | m < 0) return 0;
46
             return fac(n) * invfac(m) * invfac(n - m);
47
48
    } comb;
49
```

### 5.26 08 - 素数测试与因式分解(Miller-Rabin and Pollard-Rho)

#### Listing 52: math/08-Prime.cpp

```
i64 mul(i64 a, i64 b, i64 m) {
1
        return static_cast<__int128(a) * b % m;</pre>
2
    }
3
    i64 power(i64 a, i64 b, i64 m) {
4
        i64 res = 1 % m;
5
        for (; b; b >>= 1, a = mul(a, a, m))
6
            if (b & 1)
7
                res = mul(res, a, m);
8
        return res;
9
    }
10
    bool isprime(i64 n) {
11
        if (n < 2)
12
            return false;
13
        static constexpr int A[] = {2, 3, 5, 7, 11, 13, 17, 19, 23};
14
        int s = \_builtin\_ctzll(n - 1);
15
```

```
i64 d = (n-1) >> s;
16
        for (auto a : A) {
17
             if (a == n)
18
                 return true;
19
             i64 \times = power(a, d, n);
20
             if (x == 1 || x == n - 1)
21
22
                 continue;
             bool ok = false;
23
24
             for (int i = 0; i < s - 1; ++i) {
                 x = mul(x, x, n);
25
                 if (x == n - 1) {
26
                     ok = true;
27
28
                     break;
                 }
29
30
             if (!ok)
31
                 return false;
32
33
34
        return true;
35
    std::vector<i64> factorize(i64 n) {
36
37
        std::vector<i64> p;
        std::function< void(i64)> f = [8](i64 n) {
38
             if (n <= 10000) {
39
                 for (int i = 2; i * i <= n; ++i)
40
                     for (; n \% i == 0; n /= i)
41
                         p.push_back(i);
42
                 if (n > 1)
43
44
                     p.push_back(n);
45
                 return;
             }
46
             if (isprime(n)) {
47
                 p.push_back(n);
48
                 return;
49
50
             auto g = [8](i64 x) {
51
                 return (mul(x, x, n) + 1) % n;
52
53
             i64 \times 0 = 2;
54
            while (true) {
55
                 i64 x = x0;
56
                 i64 y = x0;
57
                 i64 d = 1;
58
                 i64 power = 1, lam = 0;
59
                 i64 v = 1;
60
                 while (d == 1) {
61
62
                     y = g(y);
63
                     ++lam;
64
                     v = mul(v, std::abs(x - y), n);
                     if (lam % 127 == 0) {
65
                         d = std::gcd(v, n);
66
                          v = 1;
67
                     }
68
                     if (power == lam) {
69
                         x = y;
70
                         power \star= 2;
71
                         lam = 0;
72
                         d = std::gcd(v, n);
73
                          v = 1;
74
                     }
75
76
                 }
                 if (d != n) {
77
                     f(d);
78
```

```
f(n / d);
79
80
                      return;
                  }
81
                 ++x0;
82
83
         };
84
         f(n);
85
         std::sort(p.begin(), p.end());
86
87
         return p;
    }
88
```

### 5.27 09A - 平面几何(Point)

#### Listing 53: math/09A-Flat-Geometry.cpp

```
/**
           平面几何 (Point)
 1
 2
     *
           2023-09-22: https://qoj.ac/submission/185408
    **/
 3
 4
    template<class T>
    struct Point {
 5
        Tx;
 6
 7
        Ту;
        Point(const T \delta x_{-} = 0, const T \delta y_{-} = 0) : x(x_{-}), y(y_{-}) {}
 8
 9
        template<class U>
10
        operator Point<U>() {
11
             return Point<U>(U(x), U(y));
12
13
14
        Point & Operator += (const Point & p) & {
15
             x += p.x;
             y += p.y;
16
             return *this;
17
        }
18
        Point & Operator—=(const Point & p) & {
19
             x = p.x;
20
             y = p.y;
21
             return *this;
22
23
        Point & Operator*=(const T & v) & {
24
             x *= V;
25
             y *= v;
26
             return *this;
27
28
        Point & Operator /= (const T & v) & {
29
             x /= v;
30
             v /= v;
31
             return *this;
32
33
        Point operator—() const {
34
             return Point(-x, -y);
35
36
        friend Point operator+(Point a, const Point &b) {
37
             return a += b;
38
39
        friend Point operator—(Point a, const Point &b) {
40
             return a -= b;
41
42
        friend Point operator*(Point a, const T &b) {
43
44
             return a *= b;
45
        friend Point operator/(Point a, const T &b) {
46
47
             return a /= b;
48
49
        friend Point operator*(const T &a, Point b) {
```

```
50
             return b *= a;
51
         friend bool operator==(const Point &a, const Point &b) {
52
             return a.x == b.x & a.y == b.y;
53
54
         friend std::istream & Soperator >> (std::istream & Sis, Point & p) {
55
56
             return is >> p.x >> p.y;
57
         friend std::ostream &operator<<(std::ostream &os, const Point &p) {
58
             return os << "(" << p.x << ",_" << p.y << ")";
59
60
    };
61
62
     template<class T>
63
     struct Line {
64
         Point<T> a;
65
66
         Point<T> b;
         Line(const Point<T> &a_ = Point<T>(), const Point<T> &b_ = Point<T>()) : a(a_), b(b_) {}
67
68
    };
69
70
    template<class T>
     T dot(const Point<T> &a, const Point<T> &b) {
71
         return a.x * b.x + a.y * b.y;
72
73
74
75
    template<class T>
     T cross(const Point<T> &a, const Point<T> &b) {
76
77
         return a.x \star b.y – a.y \star b.x;
78
79
    template<class T>
80
81
     T square(const Point<T> &p) {
         return dot(p, p);
82
83
84
    template<class T>
85
     double length(const Point<T> &p) {
86
         return std::sqrt(square(p));
87
     }
88
89
     template<class T>
90
    double length(const Line≺T> &l) {
91
         return length(l.a - l.b);
92
     }
93
94
     template<class T>
95
     Point<T> normalize(const Point<T> &p) {
96
97
         return p / length(p);
     }
98
99
     template<class T>
100
     bool parallel(const Lin≪T> &l1, const Lin≪T> &l2) {
101
         return cross(l1.b - l1.a, l2.b - l2.a) == 0;
102
103
104
     template<class T>
105
     double distance(const Point<T> &a, const Point<T> &b) {
106
107
         return length(a - b);
108
     }
109
     template<class T>
110
     double distancePL(const Point<T> &p, const Line<T> &l) {
111
         return std::abs(cross(l.a - l.b, l.a - p)) / length(l);
112
113
    }
```

```
114
     template<class T>
115
     double distancePS(const Point<T> &p, const Line<T> &l) {
116
         if (dot(p - l.a, l.b - l.a) < 0) {
117
             return distance(p, l.a);
118
         }
119
         if (dot(p - l.b, l.a - l.b) < 0) {
120
121
             return distance(p, l.b);
122
         return distancePL(p, 1);
123
124
     }
125
126
     template<class T>
     Point<T> rotate(const Point<T> &a) {
127
         return Point(-a.y, a.x);
128
129
     }
130
131
     template<class T>
     int sgn(const Point<T> &a) {
132
         return a.y > 0 || (a.y == 0 \& a.x > 0) ? 1 : -1;
133
     }
134
135
     template<class T>
136
     bool pointOnLineLeft(const Point<T> &p, const Line<T> &l) {
137
         return cross(l.b - l.a, p - l.a) > 0;
138
     }
139
140
     template<class T>
141
     Point<T> lineIntersection(const Line<T> 8l1, const Line<T> 8l2) {
142
         return l1.a + (l1.b - l1.a) * (cross(l2.b - l2.a, l1.a - l2.a) / cross(l2.b - l2.a, l1.a - l1.b));
143
     }
144
145
     template<class T>
146
     bool pointOnSegment(const Point<T> &p, const Line<T> &l) {
147
         return cross(p - l.a, l.b - l.a) == 0 \delta\delta std::min(l.a.x, l.b.x) <= p.x <math>\delta\delta p.x <= std::max(l.a.x, l.b.x)
148
             && std::min(l.a.y, l.b.y) <= p.y && p.y <= std::max(l.a.y, l.b.y);
149
     }
150
151
     template<class T>
152
     bool pointInPolygon(const Point<T> &a, const std::vector<Point<T>> &p) {
153
154
         int n = p.size();
         for (int i = 0; i < n; i++) {
155
              if (pointOnSegment(a, Line(p[i], p[(i + 1) % n]))) \{
156
                  return true;
157
158
         }
159
160
         int t = 0;
161
         for (int i = 0; i < n; i++) {
162
             auto u = p[i];
163
             auto v = p[(i + 1) \% n];
164
             if (u.x < a.x \& v.x >= a.x \& pointOnLineLeft(a, Line(v, u))) {
165
                  t ^= 1;
166
167
168
             if (u.x >= a.x \& v.x < a.x \& pointOnLineLeft(a, Line(u, v))) {
                  t ^= 1;
169
170
         }
171
172
173
         return t == 1;
174
     }
175
    // 0 : not intersect
176
     // 1 : strictly intersect
```

```
// 2 : overlap
178
    // 3 : intersect at endpoint
    template≺class T>
180
     std::tupl≪int, Point<T>, Point<T>> segmentIntersection(const Lin≪T> 8l1, const Lin≪T> 8l2) {
181
         if (std::max(l1.a.x, l1.b.x) < std::min(l2.a.x, l2.b.x)) {
182
             return {0, Point<T>(), Point<T>()};
183
184
         if (std::min(l1.a.x, l1.b.x) > std::max(l2.a.x, l2.b.x)) {
185
186
             return {0, Point<T>(), Point<T>()};
187
188
         if (std::max(l1.a.y, l1.b.y) < std::min(l2.a.y, l2.b.y)) {
             return {0, Point<T>(), Point<T>()};
189
190
         if (std::min(l1.a.y, l1.b.y) > std::max(l2.a.y, l2.b.y)) {
191
             return {0, Point<T>(), Point<T>()};
192
193
         if (cross(l1.b - l1.a, l2.b - l2.a) == 0) {
194
             if (cross(l1.b - l1.a, l2.a - l1.a) != 0) {
195
                 return {0, Point<T>(), Point<T>()};
196
             } else {
197
                 auto maxx1 = std::max(l1.a.x, l1.b.x);
198
                 auto minx1 = std::min(l1.a.x, l1.b.x);
199
                 auto maxy1 = std::max(l1.a.y, l1.b.y);
200
                 auto miny1 = std::min(l1.a.y, l1.b.y);
201
202
                 auto maxx2 = std::max(l2.a.x, l2.b.x);
                 auto minx2 = std::min(l2.a.x, l2.b.x);
203
                 auto maxy2 = std::max(l2.a.v, l2.b.v);
204
                 auto miny2 = std::min(l2.a.y, l2.b.y);
205
                 Point<T> p1(std::max(minx1, minx2), std::max(miny1, miny2));
206
207
                 Point<T> p2(std::min(maxx1, maxx2), std::min(maxy1, maxy2));
                 if (!pointOnSegment(p1, l1)) {
208
                     std::swap(p1.y, p2.y);
209
                 }
210
                 if (p1 == p2) {
211
                     return {3, p1, p2};
212
                 } else {
213
                     return {2, p1, p2};
214
                 }
215
             }
216
217
         auto cp1 = cross(l2.a - l1.a, l2.b - l1.a);
218
         auto cp2 = cross(l2.a - l1.b, l2.b - l1.b);
219
         auto cp3 = cross(l1.a - l2.a, l1.b - l2.a);
220
         auto cp4 = cross(l1.a - l2.b, l1.b - l2.b);
221
222
         if ((cp1 > 0 \& cp2 > 0) \mid | (cp1 < 0 \& cp2 < 0) \mid | (cp3 > 0 \& cp4 > 0) \mid | (cp3 < 0 \& cp4 < 0)) 
223
             return {0, Point<T>(), Point<T>()};
224
         }
225
226
         Point p = lineIntersection(l1, l2);
227
         if (cp1 != 0 && cp2 != 0 && cp3 != 0 && cp4 != 0) {
228
229
             return {1, p, p};
         } else {
230
             return {3, p, p};
231
         }
232
     }
233
234
     template<class T>
235
     double distanceSS(const Lin≪T> &l1, const Lin≪T> &l2) {
236
         if (std::get<0>(segmentIntersection(l1, l2)) != 0) {
237
238
             return 0.0;
         }
239
```

```
return std::min({distancePS(l1.a, l2), distancePS(l1.b, l2), distancePS(l2.a, l1), distancePS(l2.b, l1)
240
     }
241
242
     template<class T>
243
     bool segmentInPolygon(const Lin≪T> &l, const std::vector<Point<T>> &p) {
244
         int n = p.size();
245
         if (!pointInPolygon(l.a, p)) {
246
             return false;
247
248
         if (!pointInPolygon(l.b, p)) {
249
             return false;
250
251
         for (int i = 0; i < n; i++) {
252
             auto u = p[i];
253
254
             auto v = p[(i + 1) \% n];
             auto w = p[(i + 2) \% n];
255
             auto [t, p1, p2] = segmentIntersection(l, Line(u, v));
256
257
             if (t == 1) {
258
                 return false;
259
260
             if (t == 0) {
261
                 continue;
262
263
             if (t == 2) {
264
                 if (pointOnSegment(v, l) && v != l.a && v != l.b) {
265
                      if (cross(v - u, w - v) > 0) {
266
                          return false;
267
268
                  }
269
             } else {
270
                 if (p1 != u && p1 != v) {
271
                      if (pointOnLineLeft(l.a, Line(v, u))
272
                          || pointOnLineLeft(l.b, Line(v, u))) {
273
                          return false;
274
                      }
275
                 } else if (p1 == v) {
276
                      if (l.a == v) {
277
                          if (pointOnLineLeft(u, l)) {
278
                              if (pointOnLineLeft(w, l)
279
280
                                  && pointOnLineLeft(w, Line(u, v))) {
281
                                  return false;
282
                          } else {
283
284
                              if (pointOnLineLeft(w, l)
                                   || pointOnLineLeft(w, Line(u, v))) {
285
                                  return false;
286
                              }
287
288
                          }
                     } else if (l.b == v) {
289
                          if (pointOnLineLeft(u, Line(l.b, l.a))) {
290
                              if (pointOnLineLeft(w, Line(l.b, l.a))
291
                                  && pointOnLineLeft(w, Line(u, v))) {
292
293
                                  return false;
                              }
294
                          } else {
295
                              if (pointOnLineLeft(w, Line(l.b, l.a))
296
                                   || pointOnLineLeft(w, Line(u, v))) {
297
                                  return false;
298
                              }
299
                          }
300
                      } else {
301
```

```
if (pointOnLineLeft(u, l)) {
302
                              if (pointOnLineLeft(w, Line(l.b, l.a))
303
                                   || pointOnLineLeft(w, Line(u, v))) {
304
                                  return false;
305
306
                          } else {
307
                              if (pointOnLineLeft(w, l)
308
                                   || pointOnLineLeft(w, Line(u, v))) {
309
                                  return false;
310
311
                          }
312
                      }
313
                 }
314
             }
315
316
317
         return true;
     }
318
319
     template<class T>
320
321
     std::vector<Point<T>> hp(std::vector<Line<T>> lines) {
         std::sort(lines.begin(), lines.end(), [&](auto l1, auto l2) {
322
             auto d1 = l1.b - l1.a;
323
324
             auto d2 = 12.b - 12.a;
325
             if (sgn(d1) != sgn(d2)) {
326
                 return sgn(d1) == 1;
327
328
329
             return cross(d1, d2) > 0;
330
         });
331
332
         std::deque<Line<T>> ls;
333
         std::deque<Point<T>> ps;
334
         for (auto l : lines) {
335
             if (ls.empty()) {
                 ls.push back(l);
337
338
                 continue;
             }
339
340
             while (!ps.empty() && !pointOnLineLeft(ps.back(), l)) {
341
                 ps.pop_back();
342
                 ls.pop_back();
343
344
345
             while (!ps.empty() && !pointOnLineLeft(ps[0], l)) {
346
                 ps.pop front();
347
                 ls.pop_front();
348
349
350
             if (cross(l.b - l.a, ls.back().b - ls.back().a) == 0) {
351
                  if (dot(l.b - l.a, ls.back().b - ls.back().a) > 0) {
352
353
                      if (!pointOnLineLeft(ls.back().a, l)) {
354
                          assert(ls.size() == 1);
355
                          ls[0] = l;
356
                      }
357
358
                      continue;
                  }
359
                 return {};
360
             }
361
362
             ps.push_back(lineIntersection(ls.back(), l));
363
364
             ls.push_back(l);
         }
365
```

```
366
        while (!ps.empty() && !pointOnLineLeft(ps.back(), ls[0])) {
367
            ps.pop back();
368
            ls.pop_back();
369
        }
370
        if (ls.size() <= 2) {
371
372
            return {};
373
374
        ps.push_back(lineIntersection(ls[0], ls.back()));
375
376
        return std::vector(ps.begin(), ps.end());
377
378
    using real = long double;
379
    using P = Point<real>;
380
381
    constexpr real eps = 0;
382
            09B - 平面几何(with. std::complex)
     5.28
```

```
Listing 54: math/09B-Flat-Geometry.cpp
```

```
/**
          平面几何 (with. std::complex)
 1
          2023-09-04: https://qoj.ac/submission/164445
 2
    **/
 3
 4
    using Point = std::complexlong double>;
 5
 6
    #define x real
    #define y imag
 7
 8
    long double dot(const Point &a, const Point &b) {
 9
        return (std::conj(a) * b).x();
10
11
12
    long double cross(const Point &a, const Point &b) {
13
        return (std::conj(a) * b).y();
14
    }
15
16
    long double length(const Point &a) {
17
        return std::sqrt(dot(a, a));
18
19
    }
20
    long double dist(const Point &a, const Point &b) {
21
        return length(a - b);
22
23
    long double get(const Point &a, const Point &b, const Point &c, const Point &d) {
25
        auto e = a + (b - a) * cross(c - a, d - a) / cross(b - a, d - c);
26
        return dist(d, e);
27
    }
28
```

#### 10 - 立体几何(Point) 5.29

#### Listing 55: math/10-Solid-Geometry.cpp

```
/**
         立体几何
1
         2023-09-25 (i64): https://qoj.ac/submission/188519
2
    *
3
         2023-09-28 (double): https://qoj.ac/submission/190463
4
   **/
  using i64 = long long;
5
  using real = double;
6
7
  struct Point {
```

```
9
        real x = 0;
10
        real y = 0;
        real z = 0;
11
    };
12
13
    Point operator+(const Point &a, const Point &b) {
14
        return \{a.x + b.x, a.y + b.y, a.z + b.z\};
15
16
17
    Point operator—(const Point &a, const Point &b) {
18
        return \{a.x - b.x, a.y - b.y, a.z - b.z\};
19
    }
20
21
    Point operator*(const Point &a, real b) {
22
        return \{a.x * b, a.y * b, a.z * b\};
23
    }
24
25
    Point operator/(const Point &a, real b) {
26
        return \{a.x / b, a.y / b, a.z / b\};
27
    }
28
29
30
    real length(const Point &a) {
        return std::hypot(a.x, a.y, a.z);
31
32
    }
33
    Point normalize(const Point &a) {
34
        real l = length(a);
35
        return \{a.x / l, a.y / l, a.z / l\};
36
    }
37
38
    real getAng(real a, real b, real c) {
39
        return std::acos((a * a + b * b - c * c) / 2 / a / b);
40
    }
41
42
    std::ostream &operator<<(std::ostream &os, const Point &a) {</pre>
43
        return os << "(" << a.x << ",_" << a.y << ",_" << a.z << ")";
44
45
46
47
    real dot(const Point &a, const Point &b) {
48
        return a.x * b.x + a.y * b.y + a.z * b.z;
49
50
    Point cross(const Point &a, const Point &b) {
51
52
        return {
53
            a.y * b.z - a.z * b.y
            a.z * b.x - a.x * b.z,
54
            a.x * b.y - a.y * b.x
55
        };
56
    }
57
```

### 5.30 11A - 静态凸包(with. Point, 旧版)

#### Listing 56: math/11A-Convex-Hull.cpp

```
/**
         静态凸包(with. Point, 旧版)
1
2
   *
         2023-04-09: https://cf.dianhsu.com/gym/104288/submission/201412835
3
   **/
   struct Point {
4
       i64 x;
5
       i64 y;
6
       Point(i64 x = 0, i64 y = 0) : x(x), y(y) {}
7
   };
8
9
   bool operator==(const Point &a, const Point &b) {
```

```
return a.x == b.x && a.y == b.y;
    }
12
13
    Point operator+(const Point &a, const Point &b) {
        return Point(a.x + b.x, a.y + b.y);
15
    }
16
17
    Point operator—(const Point &a, const Point &b) {
18
        return Point(a.x - b.x, a.y - b.y);
19
    }
20
21
    i64 dot(const Point &a, const Point &b) {
22
        return a.x * b.x + a.y * b.y;
23
    }
24
25
    i64 cross(const Point &a, const Point &b) {
26
        return a.x * b.y - a.y * b.x;
27
    }
28
29
    void norm(std::vector<Point> &h) {
30
        int i = 0;
31
        for (int j = 0; j < int(h.size()); j++) {
32
            if (h[j].y < h[i].y || (h[j].y == h[i].y && h[j].x < h[i].x)) {
33
34
35
        }
36
        std::rotate(h.begin(), h.begin() + i, h.end());
37
    }
38
39
40
    int sgn(const Point &a) {
        return a.y > 0 || (a.y == 0 \& a.x > 0) ? 0 : 1;
41
42
43
    std::vector<Point> getHull(std::vector<Point> p) {
44
        std::vector<Point> h, l;
45
        std::sort(p.begin(), p.end(), [&](auto a, auto b) {
46
47
            if (a.x != b.x) {
48
                return a.x < b.x;
            } else {
49
                return a.y < b.y;
50
            }
51
        });
52
        p.erase(std::unique(p.begin(), p.end()), p.end());
53
        if (p.size() <= 1) {
54
            return p;
55
56
57
        for (auto a : p) {
58
            while (h.size() > 1 \& cross(a - h.back(), a - h[h.size() - 2]) <= 0) {
59
                h.pop back();
60
61
            while (l.size() > 1 \& cross(a - l.back(), a - l[l.size() - 2]) >= 0) 
62
                l.pop_back();
63
64
            l.push_back(a);
65
            h.push_back(a);
66
        }
67
68
        l.pop_back();
69
        std::reverse(h.begin(), h.end());
70
71
        h.pop_back();
        l.insert(l.end(), h.begin(), h.end());
72
        return l;
73
```

74 }

### 5.31 11B - 静态凸包 (with. Point, 新版)

### Listing 57: math/11B-Convex-Hull.cpp

```
/**
          静态凸包(with. Point,新版)
 1
          2024-04-06: https://qoj.ac/submission/379920)
 2
    **/
 3
    struct Point {
 4
        i64 x;
 5
        i64 y;
 6
        Point(): x{0}, y{0} {}
 7
        Point(i64 x_, i64 y_) : x\{x_{}, y\{y_{}\} \{\}
 8
    };
 9
10
    i64 dot(Point a, Point b) {
11
        return a.x * b.x + a.y * b.y;
12
13
14
    i64 cross(Point a, Point b) {
15
        return a.x \star b.y – a.y \star b.x;
16
17
18
    Point operator+(Point a, Point b) {
19
        return Point(a.x + b.x, a.y + b.y);
20
    }
21
22
    Point operator—(Point a, Point b) {
23
24
        return Point(a.x - b.x, a.y - b.y);
25
26
    auto getHull(std::vector<Point> p) {
27
        std::sort(p.begin(), p.end(),
28
            [&](auto a, auto b) {
29
                return a.x < b.x || (a.x == b.x \&\& a.y < b.y);
30
            });
31
32
        std::vector<Point> hi, lo;
33
34
        for (auto p : p) {
            while (hi.size() > 1 \& cross(hi.back() - hi[hi.size() - 2], p - hi.back()) >= 0) {
35
                hi.pop_back();
36
37
            while (!hi.empty() && hi.back().x == p.x) {
38
39
                hi.pop_back();
40
            hi.push back(p);
41
            while (lo.size() > 1 \& cross(lo.back() - lo[lo.size() - 2], p - lo.back()) <= 0)
42
                lo.pop_back();
43
44
            if (lo.empty() || lo.back().x < p.x) {
45
                lo.push_back(p);
46
47
48
        return std::make_pair(hi, lo);
49
50
    const double inf = INFINITY;
52
```

### 5.32 11C - 静态凸包(with. std::complex)

#### Listing 58: math/11C-Convex-Hull.cpp

```
/**
          静态凸包(with.std::complex)
 1
 2
     *
          2022-02-04: https://loj.ac/s/1370861
    **/
 3
    using Point = std::complexi64>;
 4
 5
    #define x real
 6
    #define y imag
 7
 8
    auto dot(const Point &a, const Point &b) {
 9
10
        return (std::conj(a) \star b).x();
11
12
    auto cross(const Point &a, const Point &b) {
13
14
        return (std::conj(a) * b).y();
15
16
17
    auto rot(const Point &p) {
        return Point(-p.y(), p.x());
18
    }
19
20
    auto complexHull(std::vector<Point> a) {
21
        std::sort(a.begin(), a.end(), [&](auto a, auto b) {
22
23
            if (a.x() != b.x()) {
                 return a.x() < b.x();
24
25
                 return a.y() < b.y();
26
27
        });
28
29
30
        std::vector<Point> l, h;
31
        for (auto p : a) {
32
            \label{eq:while (l.size() > 1 & cross(l.back() - l[l.size() - 2], p - l.back()) <= 0) {} \\
33
                 l.pop_back();
34
35
36
            while (h.size() > 1 \& cross(h.back() - h[h.size() - 2], p - h.back()) >= 0) {
37
                 h.pop_back();
38
39
40
            l.push_back(p);
41
42
            h.push_back(p);
43
44
45
        std::reverse(h.begin(), h.end());
46
47
        h.insert(h.end(), l.begin() + 1, l.end() - 1);
48
49
        return h;
50
    }
51
    int sgn(Point p) {
52
53
        if (p.y() > 0 \mid | (p.y() == 0 \& p.x() < 0)) {
54
            return 0;
55
        } else {
56
            return 1;
57
    }
58
```

### 5.33 12A - 多项式 (Poly, with. Z)

#### Listing 59: math/12A-Poly.cpp

```
/**
          多项式相关(Poly, with. Z)
1
2
     *
          2023-02-06: https://atcoder.jp/contests/arc155/submissions/38664055
3
    **/
    std::vector<int> rev;
    std::vector<Z> roots{0, 1};
5
6
    void dft(std::vector<Z> &a) {
        int n = a.size();
8
9
        if (int(rev.size()) != n) {
            int k = \underline{\text{builtin\_ctz}(n)} - 1;
10
            rev.resize(n);
11
            for (int i = 0; i < n; i++) {
12
                rev[i] = rev[i >> 1] >> 1 | (i & 1) << k;
13
14
        }
15
16
        for (int i = 0; i < n; i++) {
17
            if (rev[i] < i) {
18
                std::swap(a[i], a[rev[i]]);
19
20
21
        if (int(roots.size()) < n) {</pre>
22
            int k = __builtin_ctz(roots.size());
23
            roots.resize(n);
24
            while ((1 << k) < n) {
25
                 Z = power(Z(3), (P-1) >> (k+1));
26
                 for (int i = 1 << (k - 1); i < (1 << k); i++) {
27
                     roots[2 * i] = roots[i];
28
                     roots[2 * i + 1] = roots[i] * e;
29
30
31
                 k++;
            }
32
33
34
        for (int k = 1; k < n; k *= 2) {
            for (int i = 0; i < n; i += 2 * k) {
35
                 for (int j = 0; j < k; j++) {
36
                     Zu = a[i + j];
37
                     Z v = a[i + j + k] * roots[k + j];
38
                     a[i + j] = u + v;
39
                     a[i + j + k] = u - v;
40
41
                 }
            }
42
43
44
    }
45
    void idft(std::vector<Z> &a) {
        int n = a.size();
46
        std::reverse(a.begin() + 1, a.end());
47
        dft(a);
48
        Z \text{ inv} = (1 - P) / n;
49
        for (int i = 0; i < n; i++) {
50
            a[i] *= inv;
51
52
53
    struct Poly {
54
55
        std::vector<Z> a;
        Poly() {}
56
        explicit Poly(int size, std::function<Z(int)> f = [](int) { return 0; }) : a(size) {
57
            for (int i = 0; i < size; i++) {
58
                a[i] = f(i);
59
60
61
        Poly(const std::vector<Z> &a) : a(a) {}
62
```

```
Poly(const std::initializer_list<Z> &a) : a(a) {}
63
         int size() const {
64
             return a.size();
65
66
         void resize(int n) {
67
             a.resize(n);
68
69
70
         Z operator[](int idx) const {
             if (idx < size()) {
71
                 return a[idx];
72
73
             } else {
                 return 0;
75
         }
76
77
         Z & operator[](int idx) {
78
             return a[idx];
79
80
         Poly mulxk(int k) const {
81
             auto b = a;
             b.insert(b.begin(), k, 0);
82
             return Poly(b);
83
84
         Poly modxk(int k) const {
85
             k = std::min(k, size());
86
             return Poly(std::vector<Z>(a.begin(), a.begin() + k));
87
88
         Poly divxk(int k) const {
89
             if (size() <= k) {
90
                 return Poly();
91
92
             return Poly(std::vector<Z>(a.begin() + k, a.end()));
93
94
         friend Poly operator+(const Poly &a, const Poly &b) {
95
             std::vector<Z> res(std::max(a.size(), b.size()));
96
97
             for (int i = 0; i < int(res.size()); i++) {
98
                 res[i] = a[i] + b[i];
99
100
             return Poly(res);
101
         friend Poly operator—(const Poly &a, const Poly &b) {
102
             std::vector<Z> res(std::max(a.size(), b.size()));
103
             for (int i = 0; i < int(res.size()); i++) {
104
                 res[i] = a[i] - b[i];
105
106
             return Poly(res);
107
108
109
         friend Poly operator—(const Poly &a) {
             std::vector<Z> res(a.size());
110
             for (int i = 0; i < int(res.size()); i++) {</pre>
111
                 res[i] = -a[i];
112
113
             return Poly(res);
114
115
         friend Poly operator*(Poly a, Poly b) {
116
             if (a.size() == 0 || b.size() == 0) {
117
                 return Poly();
118
119
             if (a.size() < b.size()) {
120
                 std::swap(a, b);
121
122
             if (b.size() < 128) {
123
124
                 Poly c(a.size() + b.size() - 1);
                 for (int i = 0; i < a.size(); i++) {
125
```

```
for (int j = 0; j < b.size(); j++) {
126
                          c[i + j] += a[i] * b[j];
127
128
                  }
129
                  return c;
130
1.31
              int sz = 1, tot = a.size() + b.size() - 1;
132
              while (sz < tot) {
133
134
                  sz *= 2;
135
              a.a.resize(sz);
136
              b.a.resize(sz);
137
             dft(a.a);
138
              dft(b.a);
139
140
              for (int i = 0; i < sz; ++i) {
141
                  a.a[i] = a[i] * b[i];
142
143
              idft(a.a);
              a.resize(tot);
144
              return a;
145
146
         friend Poly operator*(Z a, Poly b) {
147
              for (int i = 0; i < int(b.size()); i++) {
148
                  b[i] *= a;
149
150
             return b;
151
152
         friend Poly operator*(Poly a, Z b) {
153
              for (int i = 0; i < int(a.size()); i++) {
154
155
                  a[i] *= b;
156
              return a;
157
158
         Poly & Operator += (Poly b) {
159
160
              return (*this) = (*this) + b;
161
         Poly & Operator—=(Poly b) {
162
              return (*this) = (*this) - b;
163
164
         Poly & Operator*=(Poly b) {
165
              return (*this) = (*this) * b;
166
167
         Poly Soperator*=(Z b) {
168
              return (*this) = (*this) * b;
169
170
         Poly deriv() const {
171
              if (a.empty()) {
172
                  return Poly();
173
174
              std::vector\langle Z \rangle res(size() - 1);
175
              for (int i = 0; i < size() - 1; ++i) {
176
                  res[i] = (i + 1) * a[i + 1];
177
178
              return Poly(res);
179
180
         Poly integr() const {
181
              std::vector<Z> res(size() + 1);
182
              for (int i = 0; i < size(); ++i) {
183
                  res[i + 1] = a[i] / (i + 1);
184
185
              return Poly(res);
186
187
         Poly inv(int m) const {
188
```

```
Poly x\{a[0].inv()\};
189
190
             int k = 1;
             while (k < m) {
191
                 k *= 2;
192
                 x = (x * (Poly{2} - modxk(k) * x)).modxk(k);
193
194
             return x.modxk(m);
195
         }
196
         Poly log(int m) const {
197
             return (deriv() * inv(m)).integr().modxk(m);
198
199
200
         Poly exp(int m) const {
201
             Poly x\{1\};
             int k = 1;
202
203
             while (k < m) {
204
                 k *= 2;
                 x = (x * (Poly{1} - x.log(k) + modxk(k))).modxk(k);
205
206
             return x.modxk(m);
207
208
         Poly pow(int k, int m) const {
209
             int i = 0;
210
             while (i < size() && a[i].val() == 0) {
211
212
213
             if (i == size() || 1LL * i * k >= m) {
214
                 return Poly(std::vector<Z>(m));
215
216
             Z v = a[i];
217
             auto f = divxk(i) * v.inv();
218
             return (f.\log(m-i*k)*k).\exp(m-i*k).mulxk(i*k)*power(v, k);
219
220
         Poly sqrt(int m) const {
221
             Poly x\{1\};
222
             int k = 1;
223
             while (k < m) {
224
                 k *= 2;
225
                 x = (x + (modxk(k) * x.inv(k)).modxk(k)) * ((P + 1) / 2);
226
227
             return x.modxk(m);
228
229
         Poly mulT(Poly b) const {
230
             if (b.size() == 0) {
231
232
                 return Poly();
233
234
             int n = b.size();
             std::reverse(b.a.begin(), b.a.end());
235
             return ((*this) * b).divxk(n - 1);
236
237
         std::vector<Z> eval(std::vector<Z> x) const {
238
             if (size() == 0) {
239
                 return std::vector<Z>(x.size(), 0);
240
241
             const int n = std::max(int(x.size()), size());
242
             std::vector<Poly> q(4 * n);
243
             std::vector<Z> ans(x.size());
244
             x.resize(n);
245
             std::function<br/><br/>void(int, int, int)> build = [\delta](int p, int l, int r) {
246
                 if (r - l == 1) {
247
                      q[p] = Poly{1, -x[l]};
248
                  } else {
249
                      int m = (l + r) / 2;
250
                     build(2 * p, l, m);
251
```

```
build(2 * p + 1, m, r);
252
                      q[p] = q[2 * p] * q[2 * p + 1];
253
                  }
254
             };
255
             build(1, 0, n);
256
             std::function<roid(int, int, int, const Poly &)> work = [&](int p, int l, int r, const Poly &num) {
257
                 if (r - l == 1) {
258
                      if (l < int(ans.size())) {</pre>
259
                          ans[l] = num[0];
260
                      }
261
                 } else {
262
                      int m = (l + r) / 2;
263
                      work(2 * p, l, m, num.mulT(q[2 * p + 1]).modxk(m - l));
264
                      work(2 * p + 1, m, r, num.mulT(q[2 * p]).modxk(r - m));
265
266
             };
267
             work(1, 0, n, mulT(q[1].inv(n)));
268
269
             return ans;
270
         }
    };
271
```

### 5.34 12B - 多项式 (Poly, with. MInt)

### Listing 60: math/12B-Poly.cpp

```
多项式相关(Poly, with. MInt & MLong)
    /**
 1
     *
          2023-09-20: https://atcoder.jp/contests/arc163/submissions/45737810
 2
    **/
 3
 4
    std::vector<int> rev;
    template<int P>
 5
    std::vector<MInt<P>> roots{0, 1};
 6
 7
 8
    template<int P>
    constexpr MInt<P> findPrimitiveRoot() {
 9
10
        MInt<P>i = 2;
        int k = \_builtin\_ctz(P - 1);
11
        while (true) {
12
            if (power(i, (P-1)/2)!=1) {
13
14
                break;
15
            i += 1;
16
        }
17
        return power(i, (P-1) \gg k);
18
    }
19
20
    template<int P>
21
    constexpr MInt<P> primitiveRoot = findPrimitiveRoot<P>();
22
23
    template<>
24
    constexpr MInt<998244353> primitiveRoot<998244353> {31};
25
26
    template<int P>
27
    constexpr void dft(std::vector MInt<P>> &a) {
28
        int n = a.size();
29
30
        if (int(rev.size()) != n) {
31
            int k = \underline{\text{builtin\_ctz}(n)} - 1;
32
            rev.resize(n);
33
            for (int i = 0; i < n; i++) {
34
                 rev[i] = rev[i >> 1] >> 1 | (i & 1) << k;
35
36
        }
37
38
        for (int i = 0; i < n; i++) {
39
```

```
if (rev[i] < i) {
40
                 std::swap(a[i], a[rev[i]]);
41
42
43
         }
         if (roots<P>.size() < n) {</pre>
44
             int k = __builtin_ctz(roots<P>.size());
45
             roots<P>.resize(n);
 46
47
             while ((1 << k) < n) {
                 auto e = power(primitiveRoot<P>, 1 << (_builtin_ctz(P - 1) - k - 1));
48
                 for (int i = 1 \ll (k - 1); i < (1 \ll k); i++) {
49
                     roots<P>[2 * i] = roots<P>[i];
50
                     roots<P>[2 * i + 1] = roots<P>[i] * e;
51
52
                 k++;
53
             }
54
         }
55
         for (int k = 1; k < n; k *= 2) {
56
             for (int i = 0; i < n; i += 2 * k) {
57
                 for (int j = 0; j < k; j++) {
58
                     MInt<P> u = a[i + j];
59
                     MInt<P> v = a[i + j + k] * roots<P>[k + j];
60
                     a[i + j] = u + v;
61
                     a[i + j + k] = u - v;
62
                 }
63
             }
64
65
         }
     }
66
67
68
     template<int P>
69
     constexpr void idft(std::vector<MInt<P>> &a) {
         int n = a.size();
70
         std::reverse(a.begin() + 1, a.end());
71
         dft(a);
72
         MInt<P> inv = (1 - P) / n;
73
         for (int i = 0; i < n; i++) {
74
             a[i] *= inv;
75
         }
76
     }
77
78
     templateint P = 998244353>
79
     struct Poly : public std::vector<MInt<P>>> {
80
         using Value = MInt<P>;
81
82
         Poly(): std::vector<Valu⇔() {}
83
         explicit constexpr Poly(int n) : std::vector<Valu⇔(n) {}
84
85
         explicit constexpr Poly(const std::vector<Value> &a) : std::vector<Value>(a) {}
86
         constexpr Poly(const std::initializer_list<Value> &a) : std::vector<Value>(a) {}
87
88
         template<class InputIt, class = std::_RequireInputIter<InputIt>>
89
         explicit constexpr Poly(InputIt first, InputIt last) : std::vector≺Valu⇔(first, last) {}
90
91
         template<class F>
92
         explicit constexpr Poly(int n, F f) : std::vector<Valu⇔(n) {
93
             for (int i = 0; i < n; i++) {
94
                 (*this)[i] = f(i);
95
96
97
98
         constexpr Poly shift(int k) const {
99
             if (k >= 0) {
100
                 auto b = *this;
101
                 b.insert(b.begin(), k, 0);
102
                 return b;
103
```

```
} else if (this->size() <= -k) {
104
                 return Poly();
105
             } else {
106
                 return Poly(this—>begin() + (-k), this—>end());
107
108
         }
109
         constexpr Poly trunc(int k) const {
110
             Poly f = *this;
111
             f.resize(k);
112
             return f;
113
         }
114
         constexpr friend Poly operator+(const Poly &a, const Poly &b) {
115
116
             Poly res(std::max(a.size(), b.size()));
             for (int i = 0; i < a.size(); i++) {
117
118
                 res[i] += a[i];
119
             for (int i = 0; i < b.size(); i++) {
120
121
                 res[i] += b[i];
122
123
             return res;
         }
124
125
         constexpr friend Poly operator—(const Poly &a, const Poly &b) {
             Poly res(std::max(a.size(), b.size()));
126
             for (int i = 0; i < a.size(); i++) {
127
                 res[i] += a[i];
128
129
             for (int i = 0; i < b.size(); i++) {
130
                 res[i] = b[i];
131
132
             return res;
133
         }
134
         constexpr friend Poly operator—(const Poly &a) {
135
             std::vector<Value> res(a.size());
136
             for (int i = 0; i < int(res.size()); i++) {
137
138
                 res[i] = -a[i];
139
             return Poly(res);
140
         }
141
         constexpr friend Poly operator*(Poly a, Poly b) {
142
             if (a.size() == 0 || b.size() == 0) {
143
                 return Poly();
144
145
             if (a.size() < b.size()) {
146
                 std::swap(a, b);
147
148
             int n = 1, tot = a.size() + b.size() - 1;
149
             while (n < tot) {
150
151
                 n *= 2;
152
             if (((P-1) & (n-1)) != 0 | | b.size() < 128) {
153
                 Poly c(a.size() + b.size() - 1);
154
                 for (int i = 0; i < a.size(); i++) {
155
                     for (int j = 0; j < b.size(); j++) {
156
                          c[i + j] += a[i] * b[j];
157
158
                 }
159
                 return c;
160
161
             a.resize(n);
162
             b.resize(n);
163
164
             dft(a);
             dft(b);
165
             for (int i = 0; i < n; ++i) {
166
```

```
a[i] *= b[i];
167
168
              idft(a);
169
              a.resize(tot);
170
              return a;
171
         }
172
         constexpr friend Poly operator*(Value a, Poly b) {
173
              for (int i = 0; i < int(b.size()); i++) {
174
                  b[i] *= a;
175
176
             return b;
177
         }
178
         constexpr friend Poly operator*(Poly a, Value b) {
179
              for (int i = 0; i < int(a.size()); i++) {</pre>
180
                  a[i] *= b;
181
182
              return a;
183
         }
184
         constexpr friend Poly operator/(Poly a, Value b) {
185
186
              for (int i = 0; i < int(a.size()); i++) {
187
                  a[i] /= b;
188
              return a;
189
         }
190
         constexpr Poly & Soperator+=(Poly b) {
191
192
              return (*this) = (*this) + b;
193
         constexpr Poly & Soperator == (Poly b) {
194
              return (*this) = (*this) - b;
195
196
         constexpr Poly & Soperator*=(Poly b) {
197
              return (*this) = (*this) * b;
198
         }
199
         constexpr Poly & Soperator*=(Value b) {
200
              return (*this) = (*this) * b;
201
         }
202
         constexpr Poly & Soperator/=(Value b) {
203
              return (*this) = (*this) / b;
204
         }
205
         constexpr Poly deriv() const {
206
              if (this->empty()) {
207
                  return Poly();
208
209
              Poly res(this\rightarrowsize() - 1);
210
              for (int i = 0; i < this \rightarrow size() - 1; ++i) {
211
                  res[i] = (i + 1) * (*this)[i + 1];
212
213
              return res;
214
215
         }
         constexpr Poly integr() const {
216
              Poly res(this—>size() + 1);
217
              for (int i = 0; i < this \rightarrow size(); ++i) {
218
                  res[i + 1] = (*this)[i] / (i + 1);
219
220
221
              return res;
         }
222
         constexpr Poly inv(int m) const {
223
              Poly x{(*this)[0].inv()};
224
              int k = 1;
225
             while (k < m) {
226
                  k *= 2;
227
                  x = (x * (Poly{2} - trunc(k) * x)).trunc(k);
228
229
```

```
230
             return x.trunc(m);
         }
231
         constexpr Poly log(int m) const {
232
             return (deriv() * inv(m)).integr().trunc(m);
233
234
         constexpr Poly exp(int m) const {
235
             Poly x\{1\};
236
             int k = 1;
237
             while (k < m) {
238
                 k *= 2;
239
                 x = (x * (Poly{1} - x.log(k) + trunc(k))).trunc(k);
240
241
             return x.trunc(m);
         }
243
244
         constexpr Poly pow(int k, int m) const {
245
             int i = 0;
             while (i < this\rightarrow size() \&\& (*this)[i] == 0) {
246
247
248
             if (i == this \rightarrow size() || 1LL * i * k >= m) {
249
                 return Poly(m);
250
251
             Value v = (*this)[i];
252
             auto f = shift(-i) * v.inv();
253
             return (f.\log(m-i*k)*k).\exp(m-i*k).\sinh(i*k)*power(v, k);
254
255
         constexpr Poly sqrt(int m) const {
256
             Poly x\{1\};
257
             int k = 1;
258
             while (k < m) {
259
                 k *= 2;
260
                 x = (x + (trunc(k) * x.inv(k)).trunc(k)) * CInv<2, P>;
261
262
263
             return x.trunc(m);
         }
264
         constexpr Poly mulT(Poly b) const {
265
             if (b.size() == 0) {
266
                 return Poly();
267
268
269
             int n = b.size();
270
             std::reverse(b.begin(), b.end());
             return ((*this) * b).shift(-(n-1));
271
         }
272
         constexpr std::vector<Value> eval(std::vector<Value> x) const {
273
             if (this\rightarrow size() == 0) {
274
                 return std::vector<Value>(x.size(), 0);
276
             const int n = std::max(x.size(), this->size());
277
             std::vector<Poly> q(4 * n);
278
             std::vector<Value> ans(x.size());
279
280
             x.resize(n);
             std::functionrvoid(int, int, int)> build = [8](int p, int l, int r) {
281
                 if (r - l == 1) {
282
                      q[p] = Poly{1, -x[l]};
283
                 } else {
284
                      int m = (l + r) / 2;
285
                      build(2 * p, l, m);
286
                      build(2 * p + 1, m, r);
287
                      q[p] = q[2 * p] * q[2 * p + 1];
288
289
             };
290
291
             build(1, 0, n);
             std::functionrvoid(int, int, int, const Poly &)> work = [&](int p, int l, int r, const Poly &num) {
292
```

```
if (r - l == 1) {
293
                      if (l < int(ans.size())) {</pre>
294
                          ans[l] = num[0];
295
296
                  } else {
297
                      int m = (l + r) / 2;
298
                      work(2 * p, l, m, num.mulT(q[2 * p + 1]).resize(m - l));
299
                      work(2 * p + 1, m, r, num.mulT(q[2 * p]).resize(r - m));
300
                  }
301
             };
302
303
             work(1, 0, n, mulT(q[1].inv(n)));
304
             return ans;
         }
305
     };
306
307
     template<int P = 998244353>
308
     Poly<P> berlekampMassey(const Poly<P> &s) {
309
         Poly<P> c;
310
         Poly<P> oldC;
311
         int f = -1;
312
         for (int i = 0; i < s.size(); i++) {
313
             auto delta = s[i];
314
             for (int j = 1; j <= c.size(); j++) {
315
                 delta = c[j - 1] * s[i - j];
316
317
             if (delta == 0) {
318
                 continue;
319
320
             if (f == -1) {
321
322
                  c.resize(i + 1);
323
                  f = i;
             } else {
324
                 auto d = oldC;
325
                 d *= -1;
326
                 d.insert(d.begin(), 1);
327
328
                 MInt<P> df1 = 0;
329
                 for (int j = 1; j <= d.size(); j++) {
                      df1 += d[j-1] * s[f + 1 - j];
330
331
                 assert(df1 != 0);
332
333
                 auto coef = delta / df1;
334
                 d *= coef;
335
                 Poly<P> zeros(i - f - 1);
                 zeros.insert(zeros.end(), d.begin(), d.end());
336
                 d = zeros;
337
                 auto temp = c;
338
339
                 c += d;
                 if (i - temp.size() > f - oldC.size()) {
340
                      oldC = temp;
341
                      f = i;
342
                  }
343
             }
344
         }
345
         c *= -1;
346
         c.insert(c.begin(), 1);
347
         return c;
348
     }
349
350
351
     template<int P = 998244353>
352
     MInt<P> linearRecurrence(Poly<P> p, Poly<P> q, i64 n) {
353
354
         int m = q.size() - 1;
         while (n > 0) {
355
             auto newq = q;
356
```

```
for (int i = 1; i \le m; i += 2) {
357
                  newq[i] *= -1;
358
359
             auto newp = p * newq;
360
              newq = q * newq;
361
              for (int i = 0; i < m; i++) {
362
                  p[i] = newp[i * 2 + n % 2];
363
364
              for (int i = 0; i <= m; i++) {
365
                  q[i] = newq[i * 2];
366
367
              n /= 2;
368
369
         return p[0] / q[0];
370
371
     }
372
     struct Comb {
373
374
         int n;
375
         std::vector<Z> _fac;
         std::vector<Z> _invfac;
376
         std::vector<Z> _inv;
377
378
         Comb(): n{0}, _fac{1}, _invfac{1}, _inv{0} {}
379
         Comb(int n) : Comb() {
380
              init(n);
381
382
383
         void init(int m) {
384
             m = std::min(m, Z::getMod() - 1);
385
              if (m <= n) return;
386
              fac.resize(m + 1);
387
              _invfac.resize(m + 1);
388
389
              _{inv.resize(m + 1);}
390
              for (int i = n + 1; i \le m; i++) {
391
                  _{fac[i]} = _{fac[i-1]} * i;
392
393
              _{invfac[m] = _{fac[m].inv();}
394
              for (int i = m; i > n; i—) {
395
                  _{invfac[i-1] = _{invfac[i]} * i;}
396
                  _{inv[i]} = _{invfac[i]} * _{fac[i-1]};
397
398
399
              n = m;
         }
400
401
402
         Z fac(int m) {
              if (m > n) init(2 * m);
403
             return _fac[m];
404
405
         Z invfac(int m) {
406
              if (m > n) init(2 * m);
407
              return _invfac[m];
408
         }
409
         Z inv(int m) {
410
              if (m > n) init(2 * m);
411
              return _inv[m];
412
         }
413
         Z binom(int n, int m) {
414
              if (n < m \mid | m < 0) return 0;
415
              return fac(n) * invfac(m) * invfac(n - m);
416
417
     } comb;
418
419
     Poly<P> get(int n, int m) {
420
```

```
if (m == 0) {
421
             return Poly(n + 1);
422
423
         if (m % 2 == 1) {
424
             auto f = get(n, m - 1);
425
             Z p = 1;
426
             for (int i = 0; i <= n; i++) {
427
428
                  f[n-i] += comb.binom(n, i) * p;
429
                  p *= m;
             }
430
431
             return f;
         }
432
         auto f = get(n, m / 2);
433
434
         auto fm = f;
435
         for (int i = 0; i <= n; i++) {
             fm[i] *= comb.fac(i);
436
437
         Poly pw(n + 1);
438
         pw[0] = 1;
439
         for (int i = 1; i <= n; i++) {
440
             pw[i] = pw[i - 1] * (m / 2);
441
442
         for (int i = 0; i <= n; i++) {
443
             pw[i] *= comb.invfac(i);
444
445
         fm = fm.mulT(pw);
446
         for (int i = 0; i <= n; i++) {
447
             fm[i] *= comb.invfac(i);
448
449
450
         return f + fm;
     }
451
```

### 5.35 12C - 多项式乘法

#### Listing 61: math/12C-Poly.cpp

```
/**
           多项式乘法
 1
          2024-03-10: https://qoj.ac/submission/350298
 2
    **/
 3
 4
    constexpr int P = 998244353;
 5
    int power(int a, int b) {
 6
 7
        int res = 1;
        for (; b; b /= 2, a = 1LL * a * a % P) {
 8
             if (b % 2) {
 9
                 res = 1LL * res * a % P;
10
11
12
        return res;
13
    }
14
15
    std::vectorint> rev, roots {0, 1};
16
17
    void dft(std::vector<int> &a) {
18
        int n = a.size();
19
        if (int(rev.size()) != n) {
20
             int k = \underline{\text{builtin\_ctz}(n)} - 1;
21
             rev.resize(n);
22
             for (int i = 0; i < n; i++) {
23
                 rev[i] = rev[i >> 1] >> 1 | (i & 1) << k;
24
25
26
        for (int i = 0; i < n; i++) {
27
```

```
if (rev[i] < i) {
28
                 std::swap(a[i], a[rev[i]]);
29
30
        }
31
        if (roots.size() < n) {</pre>
32
            int k = __builtin_ctz(roots.size());
33
34
            roots.resize(n);
35
            while ((1 << k) < n)  {
                 int e = power(31, 1 << (\_builtin\_ctz(P - 1) - k - 1));
36
                 for (int i = 1 \iff (k - 1); i \iff (1 \iff k); i++) 
37
                     roots[2 * i] = roots[i];
38
                     roots[2 * i + 1] = 1LL * roots[i] * e % P;
39
40
                 k++;
41
            }
42
        }
43
44
45
        for (int k = 1; k < n; k *= 2) {
            for (int i = 0; i < n; i += 2 * k) {
46
                 for (int j = 0; j < k; j++) {
47
                     int u = a[i + j];
48
                     int v = 1LL * a[i + j + k] * roots[k + j] % P;
49
                     a[i + j] = (u + v) \% P;
50
                     a[i + j + k] = (u - v) \% P;
51
                 }
52
            }
53
        }
54
    }
55
56
    void idft(std::vector<int> &a) {
57
        int n = a.size();
58
        std::reverse(a.begin() + 1, a.end());
59
        dft(a);
60
        int inv = (1 - P) / n;
61
        for (int i = 0; i < n; i++) {
62
            a[i] = 1LL * a[i] * inv % P;
63
64
65
66
    std::vectorint> mul(std::vectorint> a, std::vectorint> b) {
67
        int n = 1, tot = a.size() + b.size() - 1;
68
        while (n < tot) {
69
            n *= 2;
70
71
        if (tot < 128) {
72
73
            std::vectorint> c(a.size() + b.size() - 1);
            for (int i = 0; i < a.size(); i++) {
74
                 for (int j = 0; j < b.size(); j++) {
75
                     c[i + j] = (c[i + j] + 1LL * a[i] * b[j]) % P;
76
77
            }
78
            return c;
79
        }
80
        a.resize(n);
81
        b.resize(n);
82
        dft(a);
83
        dft(b);
84
        for (int i = 0; i < n; i++) {
85
            a[i] = 1LL * a[i] * b[i] % P;
86
87
        idft(a);
88
        a.resize(tot);
89
90
        return a;
```

91 }

# 5.36 13A - 生成函数 (q-int)

### Listing 62: math/13A-Q-Int.cpp

```
/**
          生成函数 (q-int)
 1
     *
          2023-09-04: https://qoj.ac/submission/163986
 2
    **/
 3
    using i64 = long long;
 4
    using i128 = int128;
 5
 6
    i64 power(i64 a, i64 b, i64 p) {
 7
 8
        i64 \text{ res} = 1;
        for (; b; b /= 2, a = i128(a) * a % p) {
 9
            if (b % 2) {
10
                 res = i128(res) * a % p;
11
12
13
14
        return res;
    }
15
16
    std::pair<int, int> qint(int q, int n, int p) {
17
18
        q %= p;
        for (int x = 2; x * x <= n; x++) {
19
            if (n \% x == 0) {
20
                 auto [v1, e1] = qint(q, x, p);
21
22
                 auto [v2, e2] = qint(power(q, x, p), n / x, p);
23
                 return \{1LL * v1 * v2 % p, e1 + e2\};
24
25
        if (q == 1) {
26
            if (n == p) {
27
                 return {0, 1};
28
29
30
            return {n, 0};
31
        // std::cerr << q << " " << n << " " << p << "\n";
32
33
        i64 v = 1 - power(q, n, 1LL * p * p);
        if (v < 0) {
34
            v += 1LL * p * p;
35
36
        assert(v != 0);
37
        int inv = power(1 - q + p, p - 2, p);
38
        if (v \% p == 0) {
39
            return \{(v / p) * inv % p, 1\};
40
41
        } else {
42
            return \{v \% p * inv \% p, 0\};
43
    }
44
```

# 5.37 13B - 生成函数(q-Binomial)

#### Listing 63: math/13B-Q-Binomial.cpp

```
/**
         生成函数 (q-Binomial)
1
   *
         2023-09-04: https://qoj.ac/submission/164128
2
3
   **/
4
   int power(int a, int b, int p) {
       int res = 1;
5
       for (; b; b /= 2, a = 1LL * a * a % p) {
6
7
           if (b % 2) {
8
               res = 1LL * res * a % p;
```

```
9
10
        return res;
11
    }
12
13
    int qint(int n, int q, int p) {
14
        return 1LL * (power(q, n, p) - 1) * power(q - 1, p - 2, p) % p;
15
16
17
    int qBinomial(int n, int k, int q, int p) {
18
        if (q == 0) {
19
            return 1;
20
21
        int r = 0;
22
        int x = 1;
23
24
        do {
            x = 1LL * x * q % p;
25
26
            r++;
27
        } while (x != 1);
28
        if (n / r > k / r + (n - k) / r) {
29
            return 0;
30
31
32
        int num = 1, den = 1;
        for (int i = 1; i <= k % r; i++) {
33
            num = 1LL * num * qint(n % r - i + 1, q, p) % p;
34
            den = 1LL * den * qint(i, q, p) % p;
35
        }
36
        n /= r, k /= r;
37
        while (n > 0 | | k > 0) {
38
            if (n % p < k % p) {
39
40
                return 0;
41
42
            for (int i = 1; i \le k \% p; i++) {
                num = 1LL * num * (n % p - i + 1) % p;
43
                den = 1LL * den * i % p;
44
45
46
            n /= p, k /= p;
47
        int ans = 1LL * num * power(den, p - 2, p) % p;
48
        return ans;
49
50
```

# 5.38 13C - 生成函数 (Binomial 任意模数二项式)

#### Listing 64: math/13C-Q-Binomial.cpp

```
/**
          生成函数 (Binomial 任意模数二项式)
1
    *
          2023-08-22: https://codeforces.com/contest/896/submission/219861532
2
    **/
3
    std::vector<std::pair<int, int>> factorize(int n) {
4
5
        std::vector<std::pair<int, int>> factors;
        for (int i = 2; static_cast<long long>(i) * i <= n; i++) {
6
            if (n \% i == 0) {
7
                int t = 0;
8
                for (; n % i == 0; n /= i)
9
                    ++t;
10
                factors.emplace_back(i, t);
11
12
13
        if (n > 1)
14
15
            factors.emplace_back(n, 1);
16
        return factors;
17
    }
```

```
constexpr int power(int base, i64 exp) {
18
19
        int res = 1;
        for (; exp > 0; base \star= base, exp /= 2) {
20
             if (\exp \% 2 == 1) {
21
22
                 res *= base;
23
        }
24
        return res;
25
26
    }
    constexpr int power(int base, i64 exp, int mod) {
27
        int res = 1 % mod;
28
        for (; exp > 0; base = 1LL * base * base % mod, exp \neq 2) {
29
             if (exp % 2 == 1) {
30
                 res = 1LL * res * base % mod;
31
32
        }
33
        return res;
34
35
    }
    int inverse(int a, int m) {
36
        int g = m, r = a, x = 0, y = 1;
37
        while (r != 0) {
38
             int q = g / r;
39
            g %= r;
40
            std::swap(g, r);
41
             x = q * y;
42
             std::swap(x, y);
43
        }
44
        return x < 0 ? x + m : x;
45
    }
46
    int solveModuloEquations(const std::vector<std::pair<int, int>> &e) {
47
        int m = 1;
48
        for (std::size_t i = 0; i < e.size(); i++) {
49
50
             m \neq e[i].first;
51
        int res = 0;
52
        for (std::size_t i = 0; i < e.size(); i++) {</pre>
53
             int p = e[i].first;
54
             res = (res + 1LL * e[i].second * (m / p) * inverse(m / p, p)) % m;
55
56
57
        return res;
    }
58
    constexpr int N = 1E5;
59
    class Binomial {
60
        const int mod;
61
    private:
62
        const std::vector<std::pair<int, int>> factors;
63
        std::vector<int> pk;
64
        std::vector<std::vector<int>> prod;
65
        static constexpr i64 exponent(i64 n, int p) {
66
             i64 \text{ res} = 0;
67
             for (n \neq p; n > 0; n \neq p) {
68
                 res += n;
69
70
71
             return res;
72
        }
        int product(i64 n, std::size_t i) {
73
             int res = 1;
74
             int p = factors[i].first;
75
             for (; n > 0; n \neq p) {
76
                 res = 1LL * res * power(prod[i].back(), n / pk[i], pk[i]) % pk[i] * prod[i][n % pk[i]] % pk[i];
77
78
             return res;
79
80
    public:
81
```

```
Binomial(int mod) : mod(mod), factors(factorize(mod)) {
82
             pk.resize(factors.size());
83
             prod.resize(factors.size());
84
             for (std::size_t i = 0; i < factors.size(); i++) {</pre>
85
                  int p = factors[i].first;
86
                  int k = factors[i].second;
87
88
                 pk[i] = power(p, k);
89
                 prod[i].resize(std::min(N + 1, pk[i]));
90
                 prod[i][0] = 1;
                  for (int j = 1; j < prod[i].size(); j++) {
91
                      if (j \% p == 0) {
92
                          prod[i][j] = prod[i][j-1];
93
                      } else {
94
                          prod[i][j] = 1LL * prod[i][j - 1] * j % pk[i];
95
96
                      }
                  }
97
             }
98
         }
99
100
         int operator()(i64 n, i64 m) {
             if (n < m \mid | m < 0) {
101
                 return 0:
102
103
             std::vector<std::pair<int, int>> ans(factors.size());
104
             for (int i = 0; i < factors.size(); i++) {</pre>
105
                  int p = factors[i].first;
106
                  int k = factors[i].second;
107
                 int e = exponent(n, p) - exponent(m, p) - exponent(n - m, p);
108
                  if (e >= k) {
109
                      ans[i] = std::make pair(pk[i], \emptyset);
110
                  } else {
111
                      int pn = product(n, i);
112
                      int pm = product(m, i);
113
                      int pd = product(n - m, i);
114
                      int res = 1LL * pn * inverse(pm, pk[i]) % pk[i] * inverse(pd, pk[i]) % pk[i] * power(p, e)
115
                        % pk[i];
                      ans[i] = std::make pair(pk[i], res);
116
117
118
             return solveModuloEquations(ans);
119
         }
120
     };
121
```

# 5.39 14 - 自适应辛普森法 Simpson

#### Listing 65: math/14-Simpson.cpp

```
/**
          自适应辛普森法 Simpson
1
          2023-09-02: https://qoj.ac/submission/161388
    *
2
    **/
3
    const double Pi = std::acos(-1.0);
    constexpr double EPS = 1e-9;
5
6
   double v, r, d;
    double f(double x) {
7
        double s = std::sin(x);
8
        return 1 / v / (std::sqrt(s * s + 3) - s);
9
    }
10
    double simpson(double l, double r) {
11
        return (f(l) + 4 * f((l + r) / 2) + f(r)) * (r - l) / 6;
12
13
    double integral(double l, double r, double eps, double st) {
14
        double mid = (l + r) / 2;
15
        double sl = simpson(l, mid);
16
17
        double sr = simpson(mid, r);
```

```
if (std::abs(sl + sr - st) <= 15 * eps)
return sl + sr + (sl + sr - st) / 15;
return integral(l, mid, eps / 2, sl) + integral(mid, r, eps / 2, sr);
double integral(double l, double r) {
return integral(l, r, EPS, simpson(l, r));
}</pre>
```

## 5.40 15 - 矩阵 (Matrix)

### Listing 66: math/15-Matrix.cpp

```
/**
          矩阵 (Matrix)
 1
     *
          2024-03-14: https://qoj.ac/submission/353771
 2
    **/
 3
 4
    using i64 = long long;
    using u64 = unsigned long long;
 5
 6
    using Matrix = std::array<u64, 65>;
 7
 8
    Matrix operator*(const Matrix &a, const Matrix &b) {
 9
        Matrix c{};
10
        for (int i = 0; i <= 64; i++) {
11
            for (int j = 0; j \le 64; j++) {
12
                 if (j == 64 ? i == 64 : (a[i] >> j & 1)) {
13
                     c[i] ^= b[j];
14
15
            }
16
17
18
        return c;
    }
19
20
    u64 operator*(u64 a, const Matrix &b) {
21
22
        u64 c = 0;
        for (int i = 0; i \le 64; i++) {
23
            if (i == 64 \mid | (a >> i \& 1)) {
24
                 c = b[i];
25
26
        }
27
        return c;
28
    }
29
30
    Matrix readMatrix() {
31
32
        int m;
        std::cin >> m;
33
34
35
        Matrix f{};
        for (int i = 0; i < m; i++) {
36
37
            int s, o;
            u64 A;
38
39
            std::cin >> s >> o >> A;
40
            if (o == 0) {
41
                for (int j = 0; j < 64; j++) {
42
                     if (A >> ((j + s) % 64) & 1) {
43
                         f[64] = 1ULL << ((j + s) % 64);
44
                     } else {
45
                         f[j] = 1ULL << ((j + s) % 64);
46
47
                 }
48
            } else {
49
                for (int j = 0; j < 64; j++) {
50
                     if (A >> ((j + s) \% 64) \& 1) {
51
```

```
f[j] = 1ULL << ((j + s) % 64);
52
                     }
53
                 }
54
             }
55
        }
56
57
        u64 B;
58
        std::cin >> B;
59
        f[64] ^= B;
60
61
        return f;
62
    }
63
```

## 5.41 16 - 高斯消元 (guess) 【久远】

### Listing 67: math/16-Gauss-Elimination.cpp

```
/**
          高斯消元 (guess) 【久远】
1
          2020—12—02: https://www.codechef.com/viewsolution/39942900
2
    *
     **/
3
    std::vector<double> gauss(std::vector<std::vector<double>> a, std::vector<double> b) {
4
        int n = a.size();
5
        for (int i = 0; i < n; ++i) {
6
            double x = a[i][i];
7
            for (int j = i; j < n; ++j) a[i][j] /= x;
8
            b[i] /= x;
9
            for (int j = 0; j < n; ++j) {
10
                if (i == j) continue;
11
                x = a[j][i];
12
                for (int k = i; k < n; ++k) a[j][k] = a[i][k] * x;
13
                b[j] = b[i] * x;
14
15
16
17
        return b;
    }
18
```

# 5.42 01A - 树状数组(Fenwick 旧版)

### Listing 68: ds/01A-Fenwick.cpp

```
/**
          树状数组(Fenwick 旧版)
1
     *
          2023-08-11: https://ac.nowcoder.com/acm/contest/view-submission?submissionId=63382128
2
    **/
3
    template <typename T>
4
5
    struct Fenwick {
6
        int n;
        std::vector<T> a;
7
8
        Fenwick(int n = 0) {
9
            init(n);
10
        }
11
12
        void init(int n) {
13
            this\rightarrown = n;
14
            a.assign(n, T());
15
        }
16
17
        void add(int x, T v) {
18
            for (int i = x + 1; i \le n; i += i \delta -i) {
19
                 a[i - 1] += v;
20
21
        }
22
23
```

```
T sum(int x) {
24
            auto ans = T();
25
            for (int i = x; i > 0; i = i & -i) {
26
                ans += a[i - 1];
27
28
            return ans;
29
        }
30
31
        T rangeSum(int l, int r) {
32
            return sum(r) - sum(l);
33
34
35
        int kth(T k) {
36
            int x = 0;
37
            for (int i = 1 << std::_lg(n); i; i /= 2) {
38
                 if (x + i \le n \& k \ge a[x + i - 1]) {
39
40
                     x += i;
                     k = a[x - 1];
41
                 }
42
            }
43
            return x;
44
        }
45
    };
46
```

### 5.43 01B - 树状数组(Fenwick 新版)

#### Listing 69: ds/01B-Fenwick.cpp

```
/**
          树状数组 (Fenwick 新版)
 1
 2
     *
          2023-12-28: https://codeforces.com/contest/1915/submission/239262801
    **/
 3
    template <typename T>
 4
    struct Fenwick {
 5
        int n;
 6
 7
        std::vector<T> a;
 8
        Fenwick(int n_{=} = 0) {
 9
10
            init(n );
11
12
        void init(int n ) {
13
14
            n = n_{;}
            a.assign(n, T{});
15
16
17
        void add(int x, const T &v) {
18
            for (int i = x + 1; i \le n; i += i \delta -i) {
19
                a[i-1] = a[i-1] + v;
20
21
        }
22
23
        T sum(int x) {
24
            T ans{};
25
            for (int i = x; i > 0; i = i \delta - i) {
26
                ans = ans + a[i - 1];
27
28
29
            return ans;
        }
30
31
        T rangeSum(int l, int r) {
32
            return sum(r) - sum(l);
33
34
35
        int select(const T &k) {
36
```

```
37
             int x = 0;
             T cur{};
38
             for (int i = 1 \ll std:: lg(n); i; i /= 2) {
39
                 if (x + i \le n \& cur + a[x + i - 1] \le k) {
40
41
                     x += i;
                     cur = cur + a[x - 1];
42
43
             }
44
45
             return x;
        }
46
    };
47
```

### 5.44 02 - 并查集(DSU)

### Listing 70: ds/02-DSU.cpp

```
/**
          并查集 (DSU)
 1
          2023-08-04: https://ac.nowcoder.com/acm/contest/view-submission?submissionId=63239142
 2
    **/
 3
    struct DSU {
 4
        std::vector<int> f, siz;
 5
 6
 7
        DSU() {}
        DSU(int n) {
 8
            init(n);
 9
10
11
        void init(int n) {
12
            f.resize(n);
13
            std::iota(f.begin(), f.end(), 0);
14
            siz.assign(n, 1);
15
16
17
        int find(int x) {
18
19
            while (x != f[x]) {
20
                x = f[x] = f[f[x]];
21
22
            return x;
        }
23
24
25
        bool same(int x, int y) {
            return find(x) == find(y);
26
27
28
        bool merge(int x, int y) {
29
            x = find(x);
30
            y = find(y);
31
32
            if (x == y) {
                return false;
33
34
            siz[x] += siz[y];
35
            f[y] = x;
36
            return true;
37
        }
38
39
        int size(int x) {
40
            return siz[find(x)];
41
42
    };
43
```

## 5.45 03A - 线段树(SegmentTree+Info 区间加 + 单点修改)

#### Listing 71: ds/03A-Segment-Tree.cpp

```
线段树 (SegmentTree+Info 区间加+单点修改)
    /**
 1
          2023-09-13: https://qoj.ac/submission/178310
 2
     *
    **/
 3
    struct SegmentTree {
 4
 5
        int n;
        std::vector<int> tag;
 6
        std::vector<Info> info;
 7
        SegmentTree(int n_{-}): n(n_{-}), tag(4 * n), info(4 * n) {}
 8
 9
10
        void pull(int p) {
            info[p] = info[2 * p] + info[2 * p + 1];
11
12
13
        void add(int p, int v) {
14
            tag[p] += v;
15
            info[p].max += v;
16
        }
17
18
        void push(int p) {
19
            add(2 * p, tag[p]);
20
21
            add(2 * p + 1, tag[p]);
22
            tag[p] = 0;
        }
23
24
        Info query(int p, int l, int r, int x, int y) {
25
            if (l >= y || r <= x) {
26
                return {};
27
28
            if (l >= x && r <= y) {
29
                return info[p];
30
31
            int m = (l + r) / 2;
32
            push(p);
33
34
            return query(2 * p, l, m, x, y) + query(2 * p + 1, m, r, x, y);
        }
35
36
37
        Info query(int x, int y) {
            return query(1, 0, n, x, y);
38
39
40
        void rangeAdd(int p, int l, int r, int x, int y, int v) {
41
            if (l >= y || r <= x) {
42
                return;
43
            }
44
            if (l >= x & r <= y) {
45
                return add(p, v);
46
47
            int m = (l + r) / 2;
48
            push(p);
49
50
            rangeAdd(2 * p, l, m, x, y, v);
            rangeAdd(2 * p + 1, m, r, x, y, v);
51
52
            pull(p);
        }
53
54
        void rangeAdd(int x, int y, int v) {
55
            rangeAdd(1, 0, n, x, y, v);
56
57
58
        void modify(int p, int l, int r, int x, const Info &v) {
59
            if (r - l == 1) {
60
                info[p] = v;
61
62
                return;
```

```
63
            int m = (l + r) / 2;
64
            push(p);
65
66
            if (x < m) {
                modify(2 * p, l, m, x, v);
67
            } else {
68
                modify(2 * p + 1, m, r, x, v);
69
70
            pull(p);
71
        }
72
73
        void modify(int x, const Info &v) {
74
            modify(1, 0, n, x, v);
75
76
    };
77
```

# 5.46 03B - 线段树 (SegmentTree 区间乘 + 单点加)

#### Listing 72: ds/03B-Segment-Tree.cpp

```
/**
          线段树 (SegmentTree 区间乘+单点加)
1
          2023-10-18: https://cf.dianhsu.com/gym/104417/submission/223800089
2
    **/
3
4
    struct SegmentTree {
5
        int n;
        std::vector<int> tag, sum;
6
7
        SegmentTree(int n): n(n), tag(4 * n, 1), sum(4 * n) {}
8
9
        void pull(int p) {
            sum[p] = (sum[2 * p] + sum[2 * p + 1]) % P;
10
11
12
        void mul(int p, int v) {
13
            tag[p] = 1LL * tag[p] * v % P;
14
            sum[p] = 1LL * sum[p] * v % P;
15
        }
16
17
        void push(int p) {
18
            mul(2 * p, tag[p]);
19
            mul(2 * p + 1, tag[p]);
20
21
            tag[p] = 1;
        }
22
23
        int query(int p, int l, int r, int x, int y) {
24
25
            if (l >= y || r <= x) {
                return 0;
26
27
            if (l >= x & r <= y) {
28
29
                return sum[p];
30
            int m = (l + r) / 2;
31
            push(p);
32
            return (query(2 * p, l, m, x, y) + query(2 * p + 1, m, r, x, y)) % P;
33
        }
34
35
        int query(int x, int y) {
36
            return query(1, 0, n, x, y);
37
38
39
        void rangeMul(int p, int l, int r, int x, int y, int v) {
40
            if (l >= y || r <= x) {
41
                return;
42
            }
43
```

```
if (l >= x & r <= y) {
44
                return mul(p, v);
45
46
            int m = (l + r) / 2;
47
48
            push(p);
            rangeMul(2 * p, l, m, x, y, v);
49
            rangeMul(2 * p + 1, m, r, x, y, v);
50
            pull(p);
51
        }
52
53
        void rangeMul(int x, int y, int v) {
54
            rangeMul(1, 0, n, x, y, v);
55
56
57
        void add(int p, int l, int r, int x, int v) {
58
            if (r - l == 1) {
59
                sum[p] = (sum[p] + v) % P;
60
61
                return;
62
            int m = (l + r) / 2;
63
            push(p);
64
            if (x < m) {
65
                add(2 * p, l, m, x, v);
66
67
                add(2 * p + 1, m, r, x, v);
68
69
            pull(p);
70
        }
71
72
        void add(int x, int v) {
73
74
            add(1, 0, n, x, v);
75
    };
76
```

# 5.47 03C-线段树(SegmentTree+Info 初始赋值 + 单点修改 + 查找前驱后继)

#### Listing 73: ds/03C-Segment-Tree.cpp

```
线段树 (SegmentTree+Info 初始赋值+单点修改+查找前驱后继)
    /**
1
          2023-07-17: https://ac.nowcoder.com/acm/contest/view-submission?submissionId=62804432
2
     *
     *
          2024-06-25: https://codeforces.com/contest/1982/submission/267353839
3
   **/
4
    template<class Info>
5
    struct SegmentTree {
6
7
        int n;
        std::vector<Info> info;
8
        SegmentTree(): n(0) {}
9
        SegmentTree(int n_{,} Info v_{-} = Info()) {
10
            init(n , v );
11
12
        template<class T>
13
        SegmentTree(std::vector<T> init ) {
14
            init(init_);
15
        }
16
        void init(int n_, Info v_ = Info()) {
17
            init(std::vector(n_, v_));
18
19
        template<class T>
20
        void init(std::vector<T> init_) {
21
            n = init .size();
22
            info.assign(4 << std::__lg(n), Info());</pre>
23
            std::function<br/>
x oid(int, int, int)> build = [&](int p, int l, int r) {
24
25
                if (r - l == 1) {
```

```
info[p] = init_[l];
26
27
                     return;
                 }
28
                int m = (l + r) / 2;
29
                build(2 * p, l, m);
30
                build(2 * p + 1, m, r);
31
32
                pull(p);
            };
33
34
            build(1, 0, n);
        }
35
36
        void pull(int p) {
            info[p] = info[2 * p] + info[2 * p + 1];
37
38
39
        void modify(int p, int l, int r, int x, const Info &v) {
            if (r - l == 1) {
40
                info[p] = v;
41
                return;
42
43
            int m = (l + r) / 2;
44
            if (x < m) {
45
                modify(2 * p, l, m, x, v);
46
            } else {
47
                modify(2 * p + 1, m, r, x, v);
48
49
            pull(p);
50
        }
51
        void modify(int p, const Info &v) {
52
53
            modify(1, 0, n, p, v);
54
55
        Info rangeQuery(int p, int l, int r, int x, int y) {
            if (l >= y || r <= x) {
56
                return Info();
57
58
            if (l >= x & r <= y) {
59
                return info[p];
60
61
            int m = (l + r) / 2;
62
63
            return rangeQuery(2 * p, l, m, x, y) + rangeQuery(2 * p + 1, m, r, x, y);
        }
64
        Info rangeQuery(int l, int r) {
65
            return rangeQuery(1, 0, n, l, r);
66
67
        template<class F>
68
        int findFirst(int p, int l, int r, int x, int y, F &&pred) {
69
            if (l >= y || r <= x) {
70
                return -1;
71
72
73
            if (l >= x && r <= y && !pred(info[p])) {
74
                return -1;
75
            if (r - l == 1) {
76
77
                return l;
78
            int m = (l + r) / 2;
79
            int res = findFirst(2 * p, l, m, x, y, pred);
80
            if (res == -1) {
81
                res = findFirst(2 * p + 1, m, r, x, y, pred);
82
83
            return res;
84
        }
85
        template<class F>
86
        int findFirst(int l, int r, F &&pred) {
87
            return findFirst(1, 0, n, l, r, pred);
88
```

```
89
         template<class F>
90
         int findLast(int p, int l, int r, int x, int y, F &&pred) {
91
             if (l >= y || r <= x) {
92
                 return -1;
93
94
             if (l >= x \& r <= y \& !pred(info[p])) {
95
                 return -1;
96
97
             if (r - l == 1) {
98
                 return l;
99
100
             int m = (l + r) / 2;
101
             int res = findLast(2 * p + 1, m, r, x, y, pred);
102
103
             if (res == -1) {
                 res = findLast(2 * p, l, m, x, y, pred);
104
105
106
             return res;
107
         template<class F>
108
         int findLast(int l, int r, F &&pred) {
109
             return findLast(1, 0, n, l, r, pred);
110
111
    };
112
```

## 5.48 03D - 线段树(SegmentTree+Info+Merge 初始赋值 + 单点修改 + 区间合并)

## Listing 74: ds/03D-Segment-Tree.cpp

```
/**
          线段树 (SegmentTree+Info+Merge 初始赋值+单点修改+区间合并)
1
    *
          2022-04-23: https://codeforces.com/contest/1672/submission/154766851
2
3
   **/
4
    template<class Info,
        class Merge = std::plus<Info>>
5
   struct SegmentTree {
6
        const int n;
7
        const Merge merge;
8
        std::vector<Info> info;
9
        SegmentTree(int n) : n(n), merge(Merge()), info(4 << std::_lg(n)) {}</pre>
10
        SegmentTree(std::vector<Info> init) : SegmentTree(init.size()) {
11
            std::functionrvoid(int, int, int)> build = [&](int p, int l, int r) {
12
                if (r - l == 1) {
13
                    info[p] = init[l];
14
                    return;
15
                }
16
                int m = (l + r) / 2;
17
                build(2 * p, l, m);
18
                build(2 * p + 1, m, r);
19
20
                pull(p);
            };
21
22
            build(1, 0, n);
23
        void pull(int p) {
24
            info[p] = merge(info[2 * p], info[2 * p + 1]);
25
26
        void modify(int p, int l, int r, int x, const Info &v) {
27
            if (r - l == 1) {
28
                info[p] = v;
29
                return;
30
31
            int m = (l + r) / 2;
32
            if (x < m) {
33
                modify(2 * p, l, m, x, v);
34
```

```
35
            } else {
                modify(2 * p + 1, m, r, x, v);
36
37
            pull(p);
38
        }
39
        void modify(int p, const Info &v) {
40
41
            modify(1, 0, n, p, v);
42
43
        Info rangeQuery(int p, int l, int r, int x, int y) {
            if (l >= y || r <= x) {
44
                return Info();
45
46
            if (l >= x && r <= y) {
47
                return info[p];
48
49
            int m = (l + r) / 2;
50
            return merge(rangeQuery(2 * p, l, m, x, y), rangeQuery(2 * p + 1, m, r, x, y));
51
52
        Info rangeQuery(int l, int r) {
53
            return rangeQuery(1, 0, n, l, r);
54
        }
55
    };
56
```

# 5.49 04 - 懒标记线段树 (LazySegmentTree)

#### Listing 75: ds/04-Lazy-Segt.cpp

```
/**
          懒标记线段树 (LazySegmentTree)
1
          2023-03-03: https://atcoder.jp/contests/joi2023yo2/submissions/39363123
2
          2023-03-12: https://codeforces.com/contest/1804/submission/197106837
3
          2023-07-17: https://ac.nowcoder.com/acm/contest/view-submission?submissionId=62804432
4
          2023—11—12: https://qoj.ac/submission/249505
5
    **/
6
    template<class Info, class Tag>
7
    struct LazySegmentTree {
8
9
        int n;
        std::vector<Info> info;
10
        std::vector<Tag> tag;
11
        LazySegmentTree(): n(0) {}
12
        LazySegmentTree(int n_, Info v_ = Info()) {
13
14
            init(n_, v_);
15
        template<class T>
16
        LazySegmentTree(std::vector<T> init_) {
17
            init(init_);
18
19
        void init(int n_, Info v_ = Info()) {
20
            init(std::vector(n_, v_));
21
22
        template<class T>
23
        void init(std::vector<T> init ) {
24
            n = init_.size();
25
            info.assign(4 << std::__lg(n), Info());</pre>
26
            tag.assign(4 << std::__lg(n), Tag());
27
            std::function<br/><br/>void(int, int, int)> build = [\delta](int p, int l, int r) {
28
                if (r - l == 1) {
29
                     info[p] = init_[l];
30
31
                    return;
32
                int m = (l + r) / 2;
33
                build(2 * p, l, m);
34
                build(2 * p + 1, m, r);
35
                pull(p);
36
```

```
37
            build(1, 0, n);
38
39
        void pull(int p) {
40
            info[p] = info[2 * p] + info[2 * p + 1];
41
42
        void apply(int p, const Tag &v) {
43
            info[p].apply(v);
44
45
            tag[p].apply(v);
        }
46
47
        void push(int p) {
            apply(2 * p, tag[p]);
48
            apply(2 * p + 1, tag[p]);
49
            tag[p] = Tag();
50
51
        void modify(int p, int l, int r, int x, const Info &v) {
52
            if (r - l == 1) {
53
                info[p] = v;
54
55
                return;
56
            int m = (l + r) / 2;
57
            push(p);
58
            if (x < m) {
59
                modify(2 * p, l, m, x, v);
60
            } else {
61
                modify(2 * p + 1, m, r, x, v);
62
63
            pull(p);
64
65
        void modify(int p, const Info &v) {
66
67
            modify(1, 0, n, p, v);
68
        Info rangeQuery(int p, int l, int r, int x, int y) {
69
            if (l >= y || r <= x) {
70
                return Info();
71
72
            if (l >= x & r <= y) {
73
                return info[p];
74
75
            int m = (l + r) / 2;
76
77
            push(p);
78
            return rangeQuery(2 * p, l, m, x, y) + rangeQuery(2 * p + 1, m, r, x, y);
79
        Info rangeQuery(int l, int r) {
80
            return rangeQuery(1, 0, n, l, r);
81
82
        void rangeApply(int p, int l, int r, int x, int y, const Tag &v) {
83
            if (l >= y || r <= x) {
84
                return;
85
86
            if (l >= x & r <= y) {
87
                apply(p, v);
88
                return;
89
90
            int m = (l + r) / 2;
91
92
            push(p);
            rangeApply(2 * p, l, m, x, y, v);
93
            rangeApply(2 * p + 1, m, r, x, y, v);
94
            pull(p);
95
96
        void rangeApply(int l, int r, const Tag &v) {
97
98
            return rangeApply(1, 0, n, l, r, v);
```

```
99
         void half(int p, int l, int r) {
100
             if (info[p].act == 0) {
101
                 return;
102
103
             if ((\inf_{p}].\min + 1) / 2 == (\inf_{p}].\max + 1) / 2) {
104
105
                 apply(p, \{-(info[p].min + 1) / 2\});
106
                 return;
107
108
             int m = (l + r) / 2;
109
             push(p);
             half(2 * p, l, m);
110
             half(2 * p + 1, m, r);
111
112
             pull(p);
         }
113
         void half() {
114
             half(1, 0, n);
115
116
117
         template<class F>
118
         int findFirst(int p, int l, int r, int x, int y, F pred) {
119
             if (l >= y || r <= x || !pred(info[p])) {
120
                 return -1;
121
122
             if (r - l == 1) {
123
                 return l;
124
125
             int m = (l + r) / 2;
126
             push(p);
127
             int res = findFirst(2 * p, l, m, x, y, pred);
128
             if (res == -1) {
129
130
                 res = findFirst(2 * p + 1, m, r, x, y, pred);
131
132
             return res;
133
         template<class F>
134
         int findFirst(int l, int r, F pred) {
135
136
             return findFirst(1, 0, n, l, r, pred);
137
138
         template<class F>
         int findLast(int p, int l, int r, int x, int y, F pred) {
139
             if (l >= y || r <= x || !pred(info[p])) {
140
                 return -1;
141
142
             if (r - l == 1) {
143
                 return l;
144
145
             int m = (l + r) / 2;
146
             push(p);
147
             int res = findLast(2 * p + 1, m, r, x, y, pred);
148
             if (res == -1) {
149
                 res = findLast(2 * p, l, m, x, y, pred);
150
151
             return res;
152
153
         template<class F>
154
         int findLast(int l, int r, F pred) {
155
             return findLast(1, 0, n, l, r, pred);
156
157
158
         void maintainL(int p, int l, int r, int pre) {
159
             if (info[p].difl > 0 && info[p].maxlowl < pre) {</pre>
160
                 return;
161
```

```
162
             if (r - l == 1) {
163
                 info[p].max = info[p].maxlowl;
164
165
                 info[p].maxl = info[p].maxr = l;
                  info[p].maxlowl = info[p].maxlowr = -inf;
166
167
                 return;
168
             int m = (l + r) / 2;
169
             push(p);
170
             maintainL(2 * p, l, m, pre);
171
172
             pre = std::max(pre, info[2 * p].max);
173
             maintainL(2 * p + 1, m, r, pre);
             pull(p);
174
         }
175
         void maintainL() {
176
             maintainL(1, 0, n, -1);
177
178
         void maintainR(int p, int l, int r, int suf) {
179
             if (info[p].difr > 0 && info[p].maxlowr < suf) {
180
181
                 return;
182
             if (r - l == 1) {
183
                  info[p].max = info[p].maxlowl;
184
185
                  info[p].maxl = info[p].maxr = l;
186
                 info[p].maxlowl = info[p].maxlowr = -inf;
187
                 return;
188
             int m = (l + r) / 2;
189
190
             push(p);
             maintainR(2 * p + 1, m, r, suf);
191
             suf = std::max(suf, info[2 * p + 1].max);
192
             maintainR(2 * p, l, m, suf);
193
             pull(p);
194
195
         void maintainR() {
196
197
             maintainR(1, 0, n, -1);
198
     };
199
200
     struct Tag {
201
         int x = 0;
202
         void apply(const Tag &t) & {
203
204
             x = std::max(x, t.x);
205
     };
206
207
     struct Info {
208
         int x = 0;
209
         void apply(const Tag &t) & {
210
211
             x = std::max(x, t.x);
212
     };
213
214
     Info operator+(const Info &a, const Info &b) {
215
216
         return {std::max(a.x, b.x)};
217
     }
```

## 5.50 05A - 取模类(Z 旧版)

Listing 76: ds/05A-ModInt-Old.cpp

```
constexpr int P = 998244353;
using i64 = long long;
```

```
// assume -P \le x \le 2P
 3
    int norm(int x) {
 4
        if (x < 0) {
 5
             x += P;
 6
 7
        if (x >= P) {
 8
             x −= P;
 9
10
        return x;
11
    }
12
    template<class T>
13
    T power(T a, i64 b) {
14
        T res = 1;
15
        for (; b; b /= 2, a *= a) {
16
             if (b % 2) {
17
                 res *= a;
18
19
        }
20
        return res;
21
    }
22
    struct Z {
23
        int x;
24
        Z(int x = 0) : x(norm(x)) \{\}
25
        Z(i64 x) : x(norm(x % P)) {}
26
        int val() const {
27
             return x;
28
29
        Z operator—() const {
30
31
             return Z(norm(P - x));
32
        Z inv() const {
33
            assert(x != 0);
34
             return power(*this, P - 2);
35
36
37
        Z & Soperator*=(const Z & Srhs) {
             x = i64(x) * rhs.x % P;
38
39
             return *this;
40
        Z & Soperator+=(const Z & rhs) {
41
             x = norm(x + rhs.x);
42
             return *this;
43
44
        Z & Soperator = (const Z & rhs) {
45
             x = norm(x - rhs.x);
46
47
             return *this;
48
        Z & Soperator/=(const Z & rhs) {
49
             return *this *= rhs.inv();
50
51
        friend Z operator*(const Z &lhs, const Z &rhs) {
52
             Z res = lhs;
53
             res *= rhs;
54
             return res;
55
56
        friend Z operator+(const Z & Slhs, const Z & Srhs) {
57
            Z res = lhs;
58
             res += rhs;
59
             return res;
60
61
        friend Z operator—(const Z &lhs, const Z &rhs) {
62
             Z res = lhs;
63
             res -= rhs;
64
65
             return res;
        }
66
```

```
friend Z operator/(const Z &lhs, const Z &rhs) {
67
           Z res = lhs;
68
           res /= rhs;
69
           return res;
70
71
       friend std::istream & Soperator>>(std::istream & Sis, Z & a) {
72
73
           is >> v;
74
           a = Z(v);
75
           return is;
76
77
       friend std::ostream & const Z & a) {
78
           return os << a.val();
79
80
   };
81
```

## 5.51 05B - 取模类 (MLong and MInt 新版)

#### Listing 77: ds/05B-ModInt-New.cpp

```
template <class T>
 1
    constexpr T power(T a, i64 b) {
 2
        T res = 1;
 3
        for (; b; b /= 2, a *= a) {
 4
 5
             if (b % 2) {
 6
                 res *= a;
 7
        }
 8
 9
        return res;
    }
10
11
    constexpr i64 mul(i64 a, i64 b, i64 p) {
12
        i64 \text{ res} = a * b - i64(1.L * a * b / p) * p;
13
14
        res %= p;
        if (res < 0) {
15
16
             res += p;
17
18
        return res;
    }
19
    template <i64 P>
20
    struct MLong {
21
        i64 x;
22
        constexpr MLong() : x{} {}
23
        constexpr MLong(i64 x) : x{norm(x % getMod())} {}
24
25
        static i64 Mod;
26
27
        constexpr static i64 getMod() {
             if (P > 0) {
28
                 return P;
29
             } else {
30
                 return Mod;
31
32
        }
33
        constexpr static void setMod(i64 Mod ) {
34
35
             Mod = Mod_{;}
36
        constexpr i64 norm(i64 x) const {
37
             if (x < 0) {
38
                 x += getMod();
39
40
             if (x \ge getMod()) {
41
42
                 x = getMod();
43
44
             return x;
```

```
}
45
         constexpr i64 val() const {
46
             return x;
47
48
         explicit constexpr operator i64() const {
49
50
             return x;
51
         constexpr MLong operator—() const {
52
             MLong res;
53
             res.x = norm(getMod() - x);
54
             return res;
55
         }
56
         constexpr MLong inv() const {
57
             assert(x != 0);
58
59
             return power(*this, getMod() -2);
         }
60
         constexpr MLong & Soperator*=(MLong rhs) & {
61
             x = mul(x, rhs.x, getMod());
62
             return *this;
63
64
         constexpr MLong & Operator+=(MLong rhs) & {
65
             x = norm(x + rhs.x);
66
             return *this;
67
         }
68
         constexpr MLong & Soperator—=(MLong rhs) & {
69
             x = norm(x - rhs.x);
70
             return *this;
71
         }
72
         constexpr MLong & Soperator/=(MLong rhs) & {
73
             return *this *= rhs.inv();
74
         }
75
         friend constexpr MLong operator*(MLong lhs, MLong rhs) {
76
             MLong res = lhs;
77
             res *= rhs;
78
             return res;
79
80
         friend constexpr MLong operator+(MLong lhs, MLong rhs) {
81
             MLong res = lhs;
82
             res += rhs;
83
             return res;
84
85
         friend constexpr MLong operator—(MLong lhs, MLong rhs) {
86
             MLong res = lhs;
87
             res -= rhs;
88
             return res;
89
90
         friend constexpr MLong operator/(MLong lhs, MLong rhs) {
91
             MLong res = lhs;
92
             res /= rhs;
93
             return res;
94
95
         friend constexpr std::istream & Soperator >> (std::istream & is, MLong & a) {
96
97
             i64 v;
             is >> v;
98
             a = MLong(v);
99
             return is;
100
101
         friend constexpr std::ostream & Soperator<<(std::ostream & Sos, const MLong & ) {</pre>
102
             return os << a.val();
103
104
         friend constexpr bool operator==(MLong lhs, MLong rhs) {
105
             return lhs.val() == rhs.val();
106
107
108
         friend constexpr bool operator!=(MLong lhs, MLong rhs) {
```

```
return lhs.val() != rhs.val();
109
         }
110
     };
111
112
     template <>
113
     i64 MLong<0LL>::Mod = i64(1E18) + 9;
114
115
     template <int P>
116
     struct MInt {
117
         int x;
118
         constexpr MInt() : x{} {}
119
         constexpr MInt(i64 x) : x{norm(x % getMod())} {}
120
121
         static int Mod;
122
         constexpr static int getMod() {
123
             if (P > 0) {
124
                  return P;
125
126
             } else {
127
                  return Mod;
128
129
         }
         constexpr static void setMod(int Mod_) {
130
             Mod = Mod_;
131
132
         constexpr int norm(int x) const {
133
             if (x < 0) {
134
                  x += getMod();
135
136
             if (x \ge getMod()) {
137
138
                  x = getMod();
139
140
             return x;
         }
141
142
         constexpr int val() const {
143
             return x;
144
145
         explicit constexpr operator int() const {
146
             return x;
147
         constexpr MInt operator—() const {
148
             MInt res;
149
             res.x = norm(getMod() - x);
150
             return res;
151
152
         constexpr MInt inv() const {
153
             assert(x != 0);
154
             return power(*this, getMod() -2);
155
         }
156
         constexpr MInt & Soperator *= (MInt rhs) & {
157
             x = 1LL * x * rhs.x % getMod();
158
             return *this;
159
160
         constexpr MInt & Operator += (MInt rhs) & {
161
             x = norm(x + rhs.x);
162
             return *this;
163
164
         constexpr MInt & Operator—=(MInt rhs) & {
165
             x = norm(x - rhs.x);
166
             return *this;
167
168
         constexpr MInt & Operator /= (MInt rhs) & {
169
             return *this *= rhs.inv();
170
171
         friend constexpr MInt operator*(MInt lhs, MInt rhs) {
172
```

```
173
             MInt res = lhs;
174
             res *= rhs;
             return res;
175
176
         friend constexpr MInt operator+(MInt lhs, MInt rhs) {
177
             MInt res = lhs;
178
             res += rhs;
179
             return res;
180
181
         friend constexpr MInt operator—(MInt lhs, MInt rhs) {
182
             MInt res = lhs;
183
             res -= rhs;
184
             return res;
185
186
         friend constexpr MInt operator/(MInt lhs, MInt rhs) {
187
             MInt res = lhs;
188
             res /= rhs;
189
             return res;
190
191
         friend constexpr std::istream & Soperator>>(std::istream & is, MInt & a) {
192
             i64 v;
193
             is >> v;
194
             a = MInt(v);
195
             return is;
196
197
         friend constexpr std::ostream & Soperator<<(std::ostream & Sos, const MInt & a) {</pre>
198
             return os << a.val();
199
200
         friend constexpr bool operator==(MInt lhs, MInt rhs) {
201
             return lhs.val() == rhs.val();
202
203
         friend constexpr bool operator!=(MInt lhs, MInt rhs) {
204
             return lhs.val() != rhs.val();
205
206
     };
207
208
     template <>
209
     int MInt<0>::Mod = 998244353;
210
211
     template <int V, int P>
212
     constexpr MInt<P> CInv = MInt<P>(V).inv();
213
214
     constexpr int P = 1000000007;
215
     using Z = MInt<P>;
216
```

### 5.52 05C - 动态取模类(ModIntBase)

#### Listing 78: ds/05C-Dynamic-ModInt.cpp

```
/**
          动态取模类 (ModIntBase)
1
          2024-08-14: https://ac.nowcoder.com/acm/contest/view-submission?submissionId=70980889&returnHomeType
2
       =1&uid=329687984
     **/
3
    // TODO: Dynamic ModInt
4
5
    template <typename T>
6
    constexpr T power(T a, u64 b) {
7
        T res\{1\};
8
        for (; b != 0; b /= 2, a \star= a) {
9
            if (b % 2 == 1) {
10
                res *= a;
11
12
        }
13
14
        return res;
```

```
}
15
16
    template <u32 P>
17
    constexpr u32 mulMod(u32 a, u32 b) {
18
19
        return 1ULL * a * b % P;
20
21
    template <u64 P>
22
    constexpr u64 mulMod(u64 a, u64 b) {
23
        u64 \text{ res} = a * b - u64(1.L * a * b / P - 0.5L) * P;
24
        res %= P;
25
26
        return res;
    }
27
28
    template <typename U, U P>
29
        requires std::unsigned_integral<U>
30
    struct ModIntBase {
31
    public:
32
        constexpr ModIntBase() : x{0} {}
33
34
        template <typename T>
35
             requires std::integral<T>
36
        constexpr ModIntBase(T x_{-}) : x\{norm(x_{-} % T\{P\})\}  {}
37
38
39
        constexpr static U norm(U x) {
             if ((x >> (8 * sizeof(U) - 1) & 1) == 1) {
40
                 x += P;
41
42
             if (x >= P) {
43
                 X -= P;
44
             }
45
46
             return x;
        }
47
48
49
        constexpr U val() const {
50
             return x;
51
52
53
        constexpr ModIntBase operator—() const {
54
            ModIntBase res;
             res.x = norm(P - x);
55
56
             return res;
        }
57
58
59
        constexpr ModIntBase inv() const {
             return power(*this, P-2);
60
61
62
        constexpr ModIntBase & Soperator*=(const ModIntBase & Srhs) & {
63
             x = mulMod P>(x, rhs.val());
64
             return *this;
65
        }
66
67
        constexpr ModIntBase & Soperator+=(const ModIntBase & rhs) & {
68
             x = norm(x + rhs.x);
69
             return *this;
70
        }
71
72
        constexpr ModIntBase & Soperator—=(const ModIntBase & Srhs) & {
73
             x = norm(x - rhs.x);
74
             return *this;
75
        }
76
77
        constexpr ModIntBase & Soperator/=(const ModIntBase & Srhs) & {
78
```

```
return *this *= rhs.inv();
 79
         }
 80
81
         friend constexpr ModIntBase operator*(ModIntBase lhs, const ModIntBase &rhs) {
 82
             lhs *= rhs;
 83
             return lhs;
 84
 85
 86
         friend constexpr ModIntBase operator+(ModIntBase lhs, const ModIntBase &rhs) {
 87
             lhs += rhs;
88
             return lhs;
 89
 90
 91
         friend constexpr ModIntBase operator—(ModIntBase lhs, const ModIntBase &rhs) {
 92
             lhs -= rhs;
93
             return lhs;
 94
 95
96
         friend constexpr ModIntBase operator/(ModIntBase lhs, const ModIntBase &rhs) {
97
             lhs /= rhs;
98
             return lhs;
99
100
101
         friend constexpr std::ostream &operator<<(std::ostream &os, const ModIntBase &a) {
102
103
             return os << a.val();
104
105
106
         friend constexpr bool operator==(ModIntBase lhs, ModIntBase rhs) {
             return lhs.val() == rhs.val();
107
         }
108
109
         friend constexpr bool operator!=(ModIntBase lhs, ModIntBase rhs) {
110
             return lhs.val() != rhs.val();
111
112
113
         friend constexpr bool operator<(ModIntBase lhs, ModIntBase rhs) {</pre>
114
             return lhs.val() < rhs.val();</pre>
115
116
117
     private:
118
         U x;
119
     };
120
121
     template <u32 P>
122
     using ModInt = ModIntBas≪u32, P>;
123
124
125
     template <u64 P>
126
     using ModInt64 = ModIntBas≪u64, P>;
127
     constexpr u32 P = 998244353;
128
     using Z = ModInt<P>;
129
             06 - 状压 RMQ (RMQ)
                                             Listing 79: ds/06-RMQ.cpp
     /**
           状压RMQ (RMQ)
 1
           2023-03-02: https://atcoder.jp/contests/joi2022ho/submissions/39351739
 2
           2023-09-04: https://qoj.ac/submission/163598
 3
     **/
  4
 5
     template<class T,
         class Cmp = std::less<T>>
 6
     struct RMQ {
 7
 8
         const Cmp cmp = Cmp();
```

```
9
        static constexpr unsigned B = 64;
10
        using u64 = unsigned long long;
11
        int n;
        std::vector<std::vector<T>> a;
12
        std::vector<T> pre, suf, ini;
13
        std::vector<u64> stk;
14
15
        RMQ() \{ \}
        RMQ(const std::vector<T> &v) {
16
            init(v);
17
18
        void init(const std::vector<T> &v) {
19
20
            n = v.size();
            pre = suf = ini = v;
21
22
            stk.resize(n);
23
            if (!n) {
24
                return;
25
26
            const int M = (n - 1) / B + 1;
            const int lg = std::__lg(M);
27
            a.assign(lg + 1, std::vector<T>(M));
28
            for (int i = 0; i < M; i++) {
29
30
                a[0][i] = v[i * B];
                for (int j = 1; j < B \& i * B + j < n; j++) {
31
                    a[0][i] = std::min(a[0][i], v[i * B + j], cmp);
32
33
34
            for (int i = 1; i < n; i++) {
35
                if (i % B) {
36
37
                    pre[i] = std::min(pre[i], pre[i - 1], cmp);
38
39
            for (int i = n - 2; i \ge 0; i = 0) {
40
                if (i \% B != B - 1) {
41
                    suf[i] = std::min(suf[i], suf[i + 1], cmp);
42
43
44
45
            for (int j = 0; j < lg; j++) {
                for (int i = 0; i + (2 << j) <= M; i++) {
46
                    a[j + 1][i] = std::min(a[j][i], a[j][i + (1 << j)], cmp);
47
48
49
            for (int i = 0; i < M; i++) {
50
                const int l = i * B;
51
                const int r = std::min(1U * n, l + B);
52
                u64 s = 0;
53
                for (int j = l; j < r; j++) {
54
                    while (s && cmp(v[j], v[std::__lg(s) + l])) {
55
                         s ^= 1ULL << std::__lg(s);
56
57
                     s = 1ULL << (j - l);
58
                    stk[j] = s;
59
60
            }
61
62
        T operator()(int l, int r) {
63
            if (l / B != (r - 1) / B) {
64
                T ans = std::min(suf[l], pre[r - 1], cmp);
65
                l = l / B + 1;
66
                r = r / B;
67
                if (l < r) {
68
                     int k = std::__lg(r - l);
69
                    ans = std::min({ans, a[k][l], a[k][r - (1 << k)]}, cmp);
70
                 }
71
```

```
72
                return ans;
            } else {
73
74
               int x = B * (l / B);
               return ini[_builtin_ctzll(stk[r - 1] >> (l - x)) + l];
75
76
77
        }
   };
78
           07A - Splay
    5.54
   struct Node {
1
```

```
Listing 80: ds/07A-Splay.cpp
         Node *l = nullptr;
 2
         Node *r = nullptr;
 3
         int cnt = 0;
 4
 5
         i64 \text{ sum} = 0;
 6
    };
 7
 8
    Node *add(Node *t, int l, 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;
         if (r - l == 1) {
15
             return x;
16
17
         int m = (l + r) / 2;
18
         if (p < m) {
19
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
27
    int find(Node *tl, Node *tr, int l, int r, int x) {
         if (r <= x) {
28
              return -1;
29
30
         if (l >= 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 - l == 1) {
36
                  return l;
37
38
         }
39
         int m = (l + r) / 2;
40
         int res = find(tl ? tl\rightarrowl : tl, tr ? tr\rightarrowl : tr, l, m, x);
41
         if (res == -1) {
42
43
              res = find(tl ? tl\rightarrowr : tl, tr ? tr\rightarrowr : tr, m, r, x);
44
         return res;
45
    }
46
47
    std::pair<int, i64> get(Node *t, int l, int r, int x, int y) {
48
         if (l >= y || r <= x || !t) {
49
              return {0, 0LL};
50
51
         if (l >= x & r <= y) {
52
```

```
return {t->cnt, t->sum};
 53
            }
 54
            int m = (l + r) / 2;
 55
            auto [cl, sl] = get(t\rightarrow l, l, m, x, y);
 56
            auto [cr, sr] = get(t\rightarrowr, m, r, x, y);
 57
            return {cl + cr, sl + sr};
 58
      }
 59
 60
      struct Tree {
 61
            int add = 0;
 62
 63
            int val = 0;
 64
            int id = 0;
            Tree *ch[2] = \{\};
 65
            Tree *p = nullptr;
 66
      };
 67
 68
      int pos(Tree *t) {
 69
            return t\rightarrow p\rightarrow ch[1] == t;
 70
 71
 72
 73
      void add(Tree *t, int v) {
 74
            t->val += v;
            t\rightarrow add += v;
 75
      }
 76
 77
      void push(Tree *t) {
 78
 79
            if (t\rightarrow ch[0]) {
                 add(t\rightarrow ch[0], t\rightarrow add);
 80
 81
            if (t-xh[1]) {
 82
                 add(t\rightarrow ch[1], t\rightarrow add);
 83
 84
            t\rightarrow add = 0;
 85
      }
 86
 87
      void rotate(Tree *t) {
 88
            Tree *q = t \rightarrow p;
 89
            int x = !pos(t);
 90
            q\rightarrow ch[!x] = t\rightarrow ch[x];
 91
            if (t\rightarrow ch[x]) t\rightarrow ch[x]\rightarrow p = q;
 92
 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;
            for (Tree *i = t; i\rightarrow p; i = i\rightarrow p) s.push_back(i\rightarrow p);
101
            while (!s.empty()) {
102
                 push(s.back());
103
104
                 s.pop_back();
            }
105
            push(t);
106
           while (t\rightarrow p) {
107
                 if (t\rightarrow p\rightarrow p) {
108
                       if (pos(t) == pos(t \rightarrow p)) rotate(t \rightarrow p);
109
                      else rotate(t);
110
111
                 rotate(t);
112
            }
113
114
      void insert(Tree *&t, Tree *x, Tree *p = nullptr) {
```

```
if (!t) {
117
118
               t = x;
               x \rightarrow p = p;
119
               return;
120
121
122
          push(t);
123
          if (x\rightarrow val < t\rightarrow val) {
124
               insert(t\rightarrow ch[0], x, t);
125
          } else {
126
               insert(t\rightarrow ch[1], x, t);
127
128
     }
129
130
      void dfs(Tree *t) {
131
          if (!t) {
132
               return;
133
          }
134
          push(t);
135
          dfs(t\rightarrow ch[0]);
136
          std::cerr << t->val << "_";
137
138
          dfs(t\rightarrow ch[1]);
      }
139
140
     std::pair<Tree *, Tree *> split(Tree *t, int x) {
141
142
          if (!t) {
               return {t, t};
143
144
145
          Tree *v = nullptr;
146
          Tree *j = t;
          for (Tree *i = t; i; ) {
147
               push(i);
148
               j = i;
149
               if (i\rightarrow xal >= x) {
150
                    v = i;
151
                    i = i\rightarrow ch[0];
152
               } else {
153
                    i = i\rightarrow ch[1];
154
155
          }
156
157
          splay(j);
158
          if (!v) {
159
               return {j, nullptr};
160
          }
161
162
          splay(v);
163
164
          Tree *u = v \rightarrow ch[0];
165
          if (u) {
166
               v\rightarrow ch[0] = u\rightarrow p = nullptr;
167
168
          // std::cerr << "split " << x << "\n";
169
          // dfs(u);
170
          // std::cerr << "\n";
171
          // dfs(v);
172
          // std::cerr << "\n";
173
          return {u, v};
174
      }
175
176
     Tree *merge(Tree *l, Tree *r) {
177
          if (!l) {
178
179
               return r;
180
```

```
if (!r) {
181
                 return l;
182
183
           Tree *i = l;
184
           while (i\rightarrow ch[1]) {
185
                 i = i\rightarrow ch[1];
186
187
           splay(i);
188
189
            i-xh[1] = r;
           r\rightarrow p = i;
190
           return i;
191
192
      5.55
                 07B - Splay
                                                        Listing 81: ds/07B-Splay.cpp
      struct Node {
  1
           Node *ch[2], *p;
  2
           bool rev;
  3
  4
           int siz = 1;
           Node() : ch{nullptr, nullptr}, p(nullptr), rev(false) {}
  5
      };
  6
  7
      void reverse(Node *t) {
           if (t) {
  8
                 std::swap(t\rightarrow ch[0], t\rightarrow ch[1]);
  9
                 t->rev ^= 1;
 10
 11
 12
      void push(Node *t) {
 13
           if (t->rev) {
 14
 15
                 reverse(t\rightarrow ch[0]);
                 reverse(t\rightarrow ch[1]);
 16
 17
                 t->rev = false;
 18
            }
 19
      }
      void pull(Node *t) {
 20
            t\rightarrow siz = (t\rightarrow ch[0] ? t\rightarrow ch[0]\rightarrow siz : 0) + 1 + (t\rightarrow ch[1] ? t\rightarrow ch[1]\rightarrow siz : 0);
 21
      }
 22
      bool isroot(Node *t) {
 23
           return t->p == nullptr || (t->p->ch[0] != t && t->p->ch[1] != t);
 24
 25
 26
      int pos(Node *t) {
 27
           return t\rightarrow p\rightarrow ch[1] == t;
 28
 29
      void pushAll(Node *t) {
           if (!isroot(t)) {
 30
                 pushAll(t->p);
 31
 32
           push(t);
 33
      }
 34
      void rotate(Node *t) {
 35
 36
           Node *q = t \rightarrow p;
 37
           int x = !pos(t);
           q\rightarrow ch[!x] = t\rightarrow ch[x];
 38
           if (t\rightarrow ch[x]) {
 39
                 t\rightarrow ch[x]\rightarrow p = q;
 40
 41
 42
           t\rightarrow p = q\rightarrow p;
           if (!isroot(q)) {
 43
                 q\rightarrow p\rightarrow ch[pos(q)] = t;
 44
 45
           t\rightarrow ch[x] = q;
 46
```

```
47
         q \rightarrow p = t;
         pull(q);
48
    }
49
    void splay(Node *t) {
50
         pushAll(t);
51
52
         while (!isroot(t)) {
             if (!isroot(t->p)) {
53
                  if (pos(t) == pos(t \rightarrow p)) {
54
                      rotate(t->p);
55
                  } else {
56
                      rotate(t);
57
58
59
             rotate(t);
60
         }
61
         pull(t);
62
    }
63
    void access(Node *t) {
64
         for (Node \star i = t, \star q = nullptr; i; q = i, i = i \rightarrow p) {
65
             splay(i);
66
             i-xch[1] = q;
67
             pull(i);
68
69
70
         splay(t);
71
72
    void makeroot(Node *t) {
         access(t);
73
         reverse(t);
74
75
    }
76
    void link(Node *x, Node *y) {
77
         makeroot(x);
         x \rightarrow p = y;
78
79
    }
    void split(Node *x, Node *y) {
80
         makeroot(x);
81
         access(y);
82
    }
83
    void cut(Node *x, Node *y) {
84
85
         split(x, y);
         x\rightarrow p = y\rightarrow ch[0] = nullptr;
86
         pull(y);
87
88
    }
    int dist(Node *x, Node *y) {
89
         split(x, y);
90
91
         return y\rightarrow siz - 1;
    }
92
             07C - Splay
    5.56
                                               Listing 82: ds/07C-Splay.cpp
    struct Matrix : std::array<std::array<i64, 4>, 4> {
 1
         Matrix(i64 v = 0) {
 2
             for (int i = 0; i < 4; i++) {
 3
                  for (int j = 0; j < 4; j++) {
 4
                      (*this)[i][j] = (i == j ? v : inf);
 5
 6
             }
 7
         }
 8
 9
    };
10
    Matrix operator*(const Matrix &a, const Matrix &b) {
```

```
12
          Matrix c(inf);
          for (int i = 0; i < 3; i++) {
13
               for (int j = 0; j < 3; j++) {
14
                    for (int k = 0; k < 4; k++) {
15
16
                         c[i][k] = std::min(c[i][k], a[i][j] + b[j][k]);
17
               }
18
19
               c[i][3] = std::min(c[i][3], a[i][3]);
20
21
          c[3][3] = 0;
22
          return c;
     }
23
24
     struct Node {
25
          Node \starch[2], \starp;
26
27
          i64 \text{ sumg} = 0;
          i64 \text{ sumh} = 0;
28
29
          i64 \text{ sumb} = 0;
30
          i64 g = 0;
31
          i64 h = 0;
32
          i64 b = 0;
33
          Matrix mat;
34
          Matrix prd;
          std::array<i64, 4> ans{};
35
36
          Node() : ch{nullptr, nullptr}, p(nullptr) {}
37
38
          void update() {
               mat = Matrix(inf);
39
               mat[0][0] = b + h - g + sumg;
40
               mat[1][1] = mat[1][2] = mat[1][3] = h + sumh;
41
42
               mat[2][0] = mat[2][1] = mat[2][2] = mat[2][3] = b + h + sumb;
43
               mat[3][3] = 0;
44
45
     };
46
     void push(Node *t) {
47
     }
     void pull(Node *t) {
48
          t \rightarrow prd = (t \rightarrow ch[0] ? t \rightarrow ch[0] \rightarrow prd : Matrix()) * t \rightarrow mat * (t \rightarrow ch[1] ? t \rightarrow ch[1] \rightarrow prd : Matrix());
49
     }
50
     bool isroot(Node *t) {
51
          return t->p == nullptr || (t->p->ch[0] != t && t->p->ch[1] != t);
52
     }
53
     int pos(Node *t) {
54
          return t\rightarrow p\rightarrow ch[1] == t;
55
56
57
     void pushAll(Node *t) {
          if (!isroot(t)) {
58
               pushAll(t->p);
59
60
          push(t);
61
     }
62
     void rotate(Node *t) {
63
64
          Node *q = t \rightarrow p;
          int x = !pos(t);
65
          q\rightarrow ch[!x] = t\rightarrow ch[x];
66
          if (t\rightarrow ch[x]) {
67
               t\rightarrow ch[x]\rightarrow p = q;
68
69
70
          t\rightarrow p = q\rightarrow p;
          if (!isroot(q)) {
71
               q\rightarrow p\rightarrow ch[pos(q)] = t;
72
73
          t\rightarrow ch[x] = q;
74
75
          q \rightarrow p = t;
```

```
pull(q);
 76
     }
77
     void splay(Node *t) {
 78
          pushAll(t);
 79
          while (!isroot(t)) {
 80
              if (!isroot(t->p)) {
 81
                   if (pos(t) == pos(t \rightarrow p)) {
82
                       rotate(t->p);
83
                   } else {
84
                       rotate(t);
85
                   }
86
              }
87
              rotate(t);
88
89
          pull(t);
 90
     }
91
92
     std::array<i64, 4> get(Node *t) {
 93
94
          std::array<i64, 4> ans;
          ans.fill(inf);
 95
 96
          ans[3] = 0;
          for (int i = 0; i < 3; i++) {
97
              for (int j = 0; j < 4; j++) {
98
                   ans[i] = std::min(ans[i], t->prd[i][j]);
99
100
101
          return ans;
102
     }
103
104
     void access(Node *t) {
105
          std::array<i64, 4> old{};
106
          for (Node *i = t, *q = nullptr; i; q = i, i = i \rightarrow p) {
107
              splay(i);
108
              if (i\rightarrow ch[1]) {
109
                   auto res = get(i\rightarrow ch[1]);
110
                   i\rightarrow sumg += res[0];
111
                   i->sumh += std::min({res[1], res[2], res[3]});
112
113
                   i—>sumb += std::min({res[0], res[1], res[2], res[3]});
114
              i\rightarrow ch[1] = q;
115
              i\rightarrow sumg = old[0];
116
              i\rightarrow sumh = std: min({old[1], old[2], old[3]});
117
              i—>sumb == std::min({old[0], old[1], old[2], old[3]});
118
              old = get(i);
119
              i—>update();
120
              pull(i);
121
          }
122
          splay(t);
123
124
125
126
127
         срр
                Splay
          /**
128
                2024-06-24: https://cf.dianhsu.com/gym/105229/submission/267199687?version=1.5
129
           **/
130
          constexpr int D = 27;
131
     struct Info {
132
          int up[D][2]{};
133
          int down[D][2]{};
134
135
          int t = 0;
136
          i64 \text{ ans} = 0;
137
     };
138
139
     Info operator+(const Info &a, const Info &b) {
```

```
140
          Info c;
          c.t = a.t ^ b.t;
141
          c.ans = a.ans + b.ans;
142
          for (int i = 0; i < D; i++) {
143
               for (int j = 0; j < 2; j++) {
144
                    c.ans += (1LL << i) * a.down[i][j] * b.up[i][j ^ 1];</pre>
145
                    c.up[i][j] += a.up[i][j] + b.up[i][j ^ (a.t >> i & 1)];
146
                    c.down[i][j] += b.down[i][j] + a.down[i][j ^ (b.t >> i & 1)];
147
148
149
          return c;
150
151
152
     struct Node {
153
          Node *ch[2], *p;
154
          Info val;
          Info tot;
155
          int cnt[D][2];
156
          i64 pair[D][2];
157
          i64 sum;
158
          Node(): ch{nullptr, nullptr}, p(nullptr), cnt{}, pair{}, sum{} {}
159
     };
160
     void pull(Node *t) {
161
          t\to t = (t\to ch[0] ? t\to ch[0]\to tot : Info{}) + t\to val + (t\to ch[1] ? t\to ch[1]\to tot : Info{});
162
163
     bool isroot(Node *t) {
164
          return t->p == nullptr || (t->p->ch[0] != t && t->p->ch[1] != t);
165
166
     int pos(Node *t) {
167
          return t\rightarrow p\rightarrow ch[1] == t;
168
     }
169
     void rotate(Node *t) {
170
          Node *q = t \rightarrow p;
171
172
          int x = !pos(t);
          q\rightarrow ch[!x] = t\rightarrow ch[x];
173
174
          if (t\rightarrow ch[x]) {
175
               t\rightarrow ch[x]\rightarrow p = q;
176
177
          t\rightarrow p = q\rightarrow p;
          if (!isroot(q)) {
178
               q\rightarrow p\rightarrow ch[pos(q)] = t;
179
180
          t\rightarrow ch[x] = q;
181
182
          q \rightarrow p = t;
          pull(q);
183
     }
184
     void update(Node *t) {
185
          t->val.ans = t->val.t + t->sum;
186
          for (int i = 0; i < D; i++) {
187
               t->val.ans += (1LL << i) * t->pair[i][t->val.t >> i & 1];
188
               for (int j = 0; j < 2; j++) {
189
                    t-\text{val.up}[i][j] = \text{t-}\text{cnt}[i][j ^ (t-\text{val.t} >> i & 1)];
190
                    t\rightarrow val.down[i][j] = t\rightarrow cnt[i][j ^ (t\rightarrow val.t >> i & 1)];
191
192
               t->val.up[i][t->val.t >> i & 1]++;
193
               t->val.down[i][t->val.t >> i & 1]++;
194
          }
195
          pull(t);
196
197
     void splay(Node *t) {
198
199
          while (!isroot(t)) {
               if (!isroot(t->p)) {
200
                    if (pos(t) == pos(t \rightarrow p)) {
201
                        rotate(t->p);
202
```

```
} else {
203
                        rotate(t);
204
205
               }
206
               rotate(t);
207
          }
208
          pull(t);
209
     }
210
     void add(Node *t, Info s) {
211
212
          for (int i = 0; i < D; i++) {
213
               for (int x = 0; x < 2; x++) {
                    t\rightarrow pair[i][x] += s.up[i][1 ^ x];
214
                   for (int j = 0; j < 2; j++) {
215
                         t->pair[i][x] += t->cnt[i][j] * s.up[i][j ^ 1 ^ x];
216
217
218
               for (int j = 0; j < 2; j++) {
219
220
                    t->cnt[i][j] += s.up[i][j];
221
222
          t->sum += s.ans;
223
     }
224
     void del(Node *t, Info s) {
225
226
          t\rightarrow sum = s.ans;
227
          for (int i = 0; i < D; i++) {
               for (int j = 0; j < 2; j++) {
228
                    t-xnt[i][j] -= s.up[i][j];
229
230
               for (int x = 0; x < 2; x++) {
231
                   for (int j = 0; j < 2; j++) {
232
                         t\rightarrow pair[i][x] = t\rightarrow cnt[i][j] * s.up[i][j ^ 1 ^ x];
233
234
                    t->pair[i][x] -= s.up[i][1 ^ x];
235
               }
236
          }
237
     }
238
     void access(Node *t, int v) {
239
240
          Info lst;
          for (Node \star i = t, \star q = nullptr; i; q = i, i = i \rightarrow p) {
241
               splay(i);
242
               if (i\rightarrow ch[1]) {
243
244
                   add(i, i\rightarrow ch[1]\rightarrow tot);
245
246
               i\rightarrow ch[1] = q;
247
               if (q) {
                   del(i, lst);
248
               } else {
249
                    i\rightarrow val.t = v;
250
251
252
               lst = i \rightarrow tot;
               update(i);
253
254
255
          splay(t);
256
     }
```

### 5.57 08A - 其他平衡树

Listing 83: ds/08A-Mysterious-Balanced-Tree.cpp

```
struct Node {
Node *l = nullptr;
Node *r = nullptr;
int sum = 0;
```

```
int sumodd = 0;
  5
  6
  7
                       Node(Node *t) {
                                   if (t) {
  8
                                              *this = *t;
  9
10
                        }
11
           };
12
13
            Node *add(Node *t, int l, int r, int x, int v) \{
14
                       t = new Node(t);
15
                       t\rightarrow sum += v;
16
                       t - sumodd += (x \% 2) * v;
17
                       if (r - l == 1) {
18
19
                                   return t;
20
                       int m = (l + r) / 2;
21
22
                       if (x < m) {
                                   t \rightarrow l = add(t \rightarrow l, l, m, x, v);
23
                       } else {
24
                                   t\rightarrow r = add(t\rightarrow r, m, r, x, v);
25
26
                       return t;
27
            }
28
29
            int query1(Node *t1, Node *t2, int l, int r, int k) {
30
                       if (r - l == 1) {
31
                                   return l;
32
33
                       int m = (l + r) / 2;
34
                       int odd = (t1 && t1->r ? t1->r->sumodd : 0) - (t2 && t2->r ? t2->r->sumodd : 0);
35
                       int cnt = (t1 \& t1-r) (t2 \& t2-r) (t2 \& 
36
                       if (odd > 0 || cnt > k) {
37
                                   return query1(t1 ? t1->r : t1, t2 ? t2->r : t2, m, r, k);
38
                       } else {
39
                                   return query1(t1 ? t1\rightarrowl : t1, t2 ? t2\rightarrowl : t2, l, m, k - cnt);
40
41
            }
42
43
            std::array<int, 3> query2(Node *t1, Node *t2, int l, int r, int k) {
44
                       if (r - l == 1) {
45
                                   int cnt = (t1 ? t1 \rightarrow sumodd : 0) - (t2 ? t2 \rightarrow sumodd : 0);
46
                                   return {l, cnt, k};
47
48
                       int m = (l + r) / 2;
49
50
                       int cnt = (t1 && t1->r ? t1->r->sumodd : 0) - (t2 && t2->r ? t2->r->sumodd : 0);
                       if (cnt > k) {
51
                                   return query2(t1 ? t1\rightarrowr : t1, t2 ? t2\rightarrowr : t2, m, r, k);
52
                       } else {
53
                                   return query2(t1 ? t1\rightarrowl : t1, t2 ? t2\rightarrowl : t2, l, m, k - cnt);
54
55
           }
56
```

### 5.58 08B - 其他平衡树

Listing 84: ds/08B-Mysterious-Balanced-Tree.cpp

```
1 struct Node {
2     Node *l = nullptr;
3     Node *r = nullptr;
4     int cnt = 0;
5 };
```

```
Node *add(Node *t, int l, int r, int x) {
 7
         if (t) {
 8
              t = new Node(*t);
 9
         } else {
10
              t = new Node;
11
12
         t->cnt += 1;
13
         if (r - l == 1) {
14
              return t;
15
16
         int m = (l + r) / 2;
17
         if (x < m) {
18
19
              t \rightarrow l = add(t \rightarrow l, l, m, x);
20
         } else {
21
              t\rightarrow r = add(t\rightarrow r, m, r, x);
22
23
         return t;
     }
24
25
     int query(Node *t1, Node *t2, int l, int r, int x) {
26
27
         int cnt = (t2 ? t2 \rightarrow cnt : 0) - (t1 ? t1 \rightarrow cnt : 0);
         if (cnt == 0 || l >= x) {
28
              return -1;
29
30
         if (r - l == 1) {
31
              return l;
32
33
         int m = (l + r) / 2;
34
         int res = query(t1 ? t1\rightarrowr : t1, t2 ? t2\rightarrowr : t2, m, r, x);
35
         if (res == -1) {
36
37
              res = query(t1 ? t1\rightarrowl : t1, t2 ? t2\rightarrowl : t2, l, m, x);
38
39
         return res;
    }
40
```

# 5.59 08C - 其他平衡树

#### Listing 85: ds/08C-Mysterious-Balanced-Tree.cpp

```
struct Info {
 1
         int imp = 0;
 2
         int id = 0;
 3
    };
 4
 5
    Info operator+(Info a, Info b) {
 6
         return {std::max(a.imp, b.imp), 0};
 7
    }
 8
 9
    struct Node {
10
         int w = rng();
11
         Info info;
12
         Info sum;
13
         int siz = 1;
14
         Node *l = nullptr;
15
16
         Node *r = nullptr;
17
    };
18
     void pull(Node *t) {
19
20
         t->sum = t->info;
21
         t\rightarrow siz = 1;
         if (t\rightarrow l) {
22
23
               t\rightarrow sum = t\rightarrow l\rightarrow sum + t\rightarrow sum;
24
               t->siz += t->l->siz;
         }
25
```

```
if (t\rightarrow r) {
26
                t\rightarrow sum = t\rightarrow sum + t\rightarrow r\rightarrow sum;
27
                t->siz += t->r->siz;
28
29
     }
30
31
     std::pair<Node *, Node *> splitAt(Node *t, int p) {
32
           if (!t) {
33
                return {t, t};
34
35
           if (p <= (t \rightarrow l ? t \rightarrow l \rightarrow siz : 0)) {
36
                auto [l, r] = splitAt(t\rightarrow l, p);
37
                t\rightarrow l = r;
38
                pull(t);
39
                return {l, t};
40
           } else {
41
                auto [l, r] = splitAt(t\rightarrowr, p - 1 - (t\rightarrowl ? t\rightarrowl\rightarrowsiz : 0));
42
                t\rightarrow r = l;
43
                pull(t);
44
                return {t, r};
45
           }
46
     }
47
48
     void insertAt(Node *&t, int p, Node *x) {
49
           if (!t) {
50
                t = x;
51
                return;
52
           }
53
           if (x-w < t-w) {
54
55
                auto [l, r] = splitAt(t, p);
56
                t = x;
                t->l = l;
57
58
                t\rightarrow r = r;
59
                pull(t);
60
                return;
           }
61
62
           if (p <= (t \rightarrow l ? t \rightarrow l \rightarrow siz : 0)) {
                insertAt(t\rightarrowl, p, x);
63
           } else {
64
                insertAt(t\rightarrowr, p - 1 - (t\rightarrowl ? t\rightarrowl\rightarrowsiz : 0), x);
65
66
           pull(t);
67
     }
68
69
     Node *merge(Node *a, Node *b) {
70
71
           if (!a) {
72
                return b;
73
74
           if (!b) {
75
                return a;
           }
76
77
78
           if (a\rightarrow w < b\rightarrow w) {
                a\rightarrow r = merge(a\rightarrow r, b);
79
                pull(a);
80
                return a;
81
           } else {
82
                b\rightarrow l = merge(a, b\rightarrow l);
83
84
                pull(b);
85
                return b;
86
87
88
     int query(Node *t, int v) {
```

```
if (!t) {
 90
 91
                 return 0;
 92
 93
           if (t\rightarrow sum.imp < v) {
 94
                 return t—>siz;
 95
           int res = query(t \rightarrow r, v);
 96
           if (res != (t\rightarrow r ? t\rightarrow r\rightarrow siz : 0)) {
 97
                 return res;
 98
 99
           if (t\rightarrow info.imp > v) {
100
                 return res;
101
102
           return res + 1 + query(t\rightarrowl, v);
103
      }
104
105
      void dfs(Node *t) {
106
           if (!t) {
107
108
                 return;
109
           dfs(t\rightarrow l);
110
           std::cout << t->info.id << "_";
111
112
           dfs(t\rightarrow r);
113
```

#### 08D - 其他平衡树 5.60

1

#### Listing 86: ds/08D-Mysterious-Balanced-Tree.cpp

```
struct Node {
          Node *l = nullptr;
 2
 3
          Node *r = nullptr;
 4
          int cnt = 0;
          int cntnew = 0;
 5
 6
     };
     Node *add(int l, int r, int x, int isnew) {
 8
          Node *t = new Node;
 9
          t\rightarrow cnt = 1;
10
          t—>cntnew = isnew;
11
          if (r - l == 1) {
12
               return t;
13
14
          int m = (l + r) / 2;
15
          if (x < m) {
16
17
                t\rightarrow l = add(l, m, x, isnew);
18
          } else {
19
                t \rightarrow r = add(m, r, x, isnew);
20
21
          return t;
22
     }
23
     struct Info {
24
          Node *t = nullptr;
25
          int psum = 0;
26
          bool rev = false;
27
     };
28
29
     void pull(Node *t) {
30
          t\rightarrow cnt = (t\rightarrow l ? t\rightarrow l\rightarrow cnt : 0) + (t\rightarrow r ? t\rightarrow r\rightarrow cnt : 0);
31
          t\rightarrow xntnew = (t\rightarrow l ? t\rightarrow l\rightarrow xntnew : 0) + (t\rightarrow r ? t\rightarrow r\rightarrow xntnew : 0);
32
     }
33
34
     std::pair<Node *, Node *> split(Node *t, int l, int r, int x, bool rev) {
```

```
if (!t) {
36
               return {t, t};
37
38
          if (x == 0) {
39
               return {nullptr, t};
40
41
          if (x == t\rightarrow xnt) {
42
43
               return {t, nullptr};
44
          if (r - l == 1) {
45
46
               Node *t2 = new Node;
47
               t2\rightarrow cnt = t\rightarrow cnt - x;
48
               t\rightarrow cnt = x;
               return {t, t2};
49
50
          Node *t2 = new Node;
51
          int m = (l + r) / 2;
52
53
          if (!rev) {
               if (t\to 1 \& x <= t\to 1\to x)
54
                    std::tie(t\rightarrow l, t2\rightarrow l) = split(t\rightarrow l, l, m, x, rev);
55
                    t2\rightarrow r = t\rightarrow r;
56
                    t->r = nullptr;
57
               } else {
58
                    std::tie(t\rightarrow r, t2\rightarrow r) = split(t\rightarrow r, m, r, x - (t\rightarrow l? t\rightarrow l\rightarrow cnt: 0), rev);
59
60
          } else {
61
               if (t->r && x <= t->r->cnt) {
62
                    std::tie(t\rightarrow r, t2\rightarrow r) = split(t\rightarrow r, m, r, x, rev);
63
                    t2->l = t->l;
64
                    t->l = nullptr;
65
               } else {
66
                    std::tie(t\rightarrowl, t2\rightarrowl) = split(t\rightarrowl, l, m, x - (t\rightarrowr ? t\rightarrowr-\timescnt : 0), rev);
67
68
          }
69
70
          pull(t);
          pull(t2);
71
          return {t, t2};
72
73
74
75
     Node *merge(Node *t1, Node *t2, int l, int r) {
          if (!t1) {
76
               return t2;
77
78
          if (!t2) {
79
               return t1;
80
81
          if (r - l == 1) {
82
               t1->cnt += t2->cnt;
83
               t1->cntnew += t2->cntnew;
84
85
               delete t2;
86
               return t1;
87
88
          int m = (l + r) / 2;
          t1->l = merge(t1->l, t2->l, l, m);
89
          t1 - r = merge(t1 - r, t2 - r, m, r);
90
91
          delete t2;
          pull(t1);
92
          return t1;
93
94
    }
```

#### 5.61 09 - 分数四则运算(Frac)

### Listing 87: ds/09-Frac.cpp

```
template<class T>
1
2
    struct Frac {
        T num;
3
4
        T den;
5
        Frac(T num_, T den_) : num(num_), den(den_) {
            if (den < 0) {
6
                den = -den;
8
                num = -num;
9
        }
10
11
        Frac(): Frac(0, 1) {}
        Frac(T num_) : Frac(num_, 1) {}
12
        explicit operator double() const {
13
            return 1. * num / den;
14
15
        Frac & Operator+=(const Frac & rhs) {
16
            num = num * rhs.den + rhs.num * den;
17
            den *= rhs.den:
18
            return *this;
19
20
        Frac & Operator—=(const Frac & rhs) {
21
            num = num * rhs.den - rhs.num * den;
22
            den *= rhs.den;
23
24
            return *this;
25
        Frac & Soperator*=(const Frac & rhs) {
26
            num *= rhs.num;
27
28
            den *= rhs.den;
29
            return *this;
30
31
        Frac & Soperator /= (const Frac & rhs) {
32
            num *= rhs.den;
33
            den *= rhs.num;
            if (den < 0) {
34
                num = -num;
35
36
                den = -den;
37
38
            return *this;
39
        friend Frac operator+(Frac lhs, const Frac &rhs) {
40
41
            return lhs += rhs;
42
43
        friend Frac operator—(Frac lhs, const Frac &rhs) {
44
            return lhs -= rhs;
45
        friend Frac operator*(Frac lhs, const Frac &rhs) {
46
            return lhs *= rhs;
47
48
        friend Frac operator/(Frac lhs, const Frac &rhs) {
49
            return lhs /= rhs;
50
51
        friend Frac operator—(const Frac &a) {
52
            return Frac(-a.num, a.den);
53
54
        friend bool operator==(const Frac 8lhs, const Frac 8rhs) {
55
            return lhs.num * rhs.den == rhs.num * lhs.den;
56
57
        friend bool operator!=(const Frac &lhs, const Frac &rhs) {
58
            return lhs.num * rhs.den != rhs.num * lhs.den;
59
60
        friend bool operator<(const Frac &lhs, const Frac &rhs) {</pre>
61
            return lhs.num * rhs.den < rhs.num * lhs.den;
62
        }
63
```

```
friend bool operator>(const Frac &lhs, const Frac &rhs) {
64
            return lhs.num * rhs.den > rhs.num * lhs.den;
65
66
        friend bool operator<=(const Frac 8lhs, const Frac 8rhs) {
67
            return lhs.num * rhs.den <= rhs.num * lhs.den;
68
69
        friend bool operator>=(const Frac 8lhs, const Frac 8rhs) {
70
71
            return lhs.num * rhs.den >= rhs.num * lhs.den;
72
        friend std::ostream & Soperator<<(std::ostream & Frac x) {</pre>
73
            T g = std::gcd(x.num, x.den);
74
            if (x.den == g) {
75
                return os << x.num / g;
76
            } else {
77
                return os << x.num / g << "/" << <math>x.den / g;
78
79
80
        }
    };
```

# 5.62 10 - 线性基(Basis)

#### Listing 88: ds/10-Basis.cpp

```
struct Basis {
1
        int a[20] {};
2
        int t[20] {};
3
4
        Basis() {
5
6
            std::fill(t, t + 20, -1);
7
8
        void add(int x, int y = 1E9) {
9
            for (int i = 0; i < 20; i++) {
10
                 if (x >> i & 1) {
11
12
                     if (y > t[i]) {
                         std::swap(a[i], x);
13
                         std::swap(t[i], y);
14
                     }
15
                     x ^= a[i];
16
                 }
17
            }
18
19
20
        bool query(int x, int y = 0) {
21
            for (int i = 0; i < 20; i++) {
22
                if ((x >> i & 1) & t[i] >= y) {
23
                     x ^= a[i];
24
25
26
27
            return x == 0;
        }
28
    };
29
```

# 5.63 143 - 高精度 (BigInt)

```
Listing 89: ds/143-BigInt.cpp
```

```
1  /** 高精度 (BigInt)
2  * 2023-09-11: https://qoj.ac/submission/176420
3  **/
4  constexpr int N = 1000;
5
6  struct BigInt {
```

```
int a[N];
 7
        BigInt(int x = 0): a{} {
 8
            for (int i = 0; x; i++) {
 9
                a[i] = x % 10;
10
                x /= 10;
11
12
        }
13
        BigInt & Operator*=(int x) {
14
            for (int i = 0; i < N; i++) {
15
                a[i] *= x;
16
17
            for (int i = 0; i < N - 1; i++) {
18
                a[i + 1] += a[i] / 10;
19
                a[i] %= 10;
20
21
            return *this;
22
        }
23
        BigInt & Operator /= (int x) {
24
            for (int i = N - 1; i \ge 0; i—) {
25
                if (i) {
26
                     a[i-1] += a[i] % x * 10;
27
28
                a[i] /= x;
29
            }
30
            return *this;
31
        }
32
        BigInt & Soperator+=(const BigInt & x) {
33
            for (int i = 0; i < N; i++) {
34
                a[i] += x.a[i];
35
                 if (a[i] >= 10) {
36
                     a[i + 1] += 1;
37
                     a[i] = 10;
38
39
            }
40
            return *this;
41
        }
42
    };
43
44
    std::ostream &operator<<(std::ostream &o, const BigInt &a) {
45
        int t = N - 1;
46
        while (a.a[t] == 0) {
47
48
49
        for (int i = t; i >= 0; i—) {
50
            o << a.a[i];
51
52
        return o;
53
    }
54
```

# 6 Watashi 代码库 (备用)

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

Listing 90: rmq.cpp

```
#include <algorithm> // copy
#include <climits> // CHAR_BIT

using namespace std;

template <typename T>
struct RMQ {
 int n;
 vector<T> e;
```

```
10
        vector<vector<int>> rmq;
11
        static const int INT BIT = sizeof(4) * CHAR BIT;
12
        static inline int LG2(int i) { return INT_BIT - 1 - __builtin_clz(i); }
13
        static inline int BIN(int i) { return 1 << i; }</pre>
14
15
        int cmp(int l, int r) const {
16
            return e[l] <= e[r] ? l : r;
17
18
19
20
        void init(int n, const T e[]) {
21
            this\rightarrown = n;
22
            vector<T>(e, e + n).swap(this\rightarrowe);
23
24
            int m = 1;
25
            while (BIN(m) <= n) {
26
                 ++m;
27
28
            vector<vector<int>(m, vector<int>(n)).swap(rmq);
29
            for (int i = 0; i < n; ++i) {
30
31
                 rmq[0][i] = i;
32
            for (int i = 0; BIN(i + 1) <= n; ++i) {
33
                 for (int j = 0; j + BIN(i + 1) <= n; ++j) {
34
                     rmq[i + 1][j] = cmp(rmq[i][j], rmq[i][j + BIN(i)]);
35
36
            }
37
        }
38
39
        int index(int l, int r) const {
40
            int b = LG2(r - l);
41
            return cmp(rmq[b][l], rmq[b][r - (1 \ll b)]);
42
43
44
        T value(int l, int r) const {
45
            return e[index(l, r)];
46
        }
47
    };
48
```

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

#### Listing 91: lca.cpp

```
#include <algorithm⊳
    #include <cstdi∞
 2
    #include <vector>
 3
   using namespace std;
 5
 6
    const int MAXM = 16;
 7
    const int MAXN = 1 << MAXM;
 8
    // LCA
10
11
    struct LCA {
        vector<int> e[MAXN];
12
        int d[MAXN], p[MAXN][MAXM];
13
14
        void dfs_(int v, int f) {
15
            p[v][0] = f;
16
            for (int i = 1; i < MAXM; ++i) {
17
                p[v][i] = p[p[v][i-1]][i-1];
18
19
            for (int i = 0; i < (int)e[v].size(); ++i) {
20
```

```
int w = e[v][i];
21
                 if (w != f) {
22
                     d[w] = d[v] + 1;
23
                     dfs_{w, v};
24
                 }
25
            }
26
        }
27
28
        int up_(int v, int m) {
29
            for (int i = 0; i < MAXM; ++i) {
30
                 if (m & (1 << i)) {
31
                     v = p[v][i];
32
33
            }
34
            return v;
35
        }
36
37
        int lca(int a, int b) {
38
            if (d[a] > d[b]) {
39
                 swap(a, b);
40
41
42
            b = up_(b, d[b] - d[a]);
            if (a == b) {
43
                 return a;
44
45
            } else {
                 for (int i = MAXM - 1; i >= 0; —i) {
46
                     if (p[a][i] != p[b][i]) {
47
                         a = p[a][i];
48
                         b = p[b][i];
49
                     }
50
51
52
                 return p[a][0];
            }
53
        }
54
55
        void init(int n) {
56
            for (int i = 0; i < n; ++i) {
57
                 e[i].clear();
58
59
        }
60
61
        void add(int a, int b) {
62
            e[a].push_back(b);
63
            e[b].push_back(a);
64
        }
65
66
        void build() {
67
            d[0] = 0;
68
            dfs_{0}(0, 0);
69
70
    } lca;
71
```

#### 6.3 树状数组

Listing 92: bit.cpp

```
#include <vector>
using namespace std;

template<typename T = int>
struct BIT {
    vector<T> a;
```

```
void init(int n) {
 9
        \text{vector} < T > (n + 1).swap(a);
10
11
12
      void add(int i, T v) {
13
        for (int j = i + 1; j < (int)a.size(); j = (j | (j - 1)) + 1) {
14
          a[j] += v;
15
16
      }
17
18
      // [0, i)
19
      T sum(int i) const {
20
21
        T ret = T();
22
        for (int j = i; j > 0; j = j & (j - 1)) {
23
          ret += a[j];
24
25
        return ret;
      }
26
27
      T get(int i) const {
28
29
        return sum(i + 1) - sum(i);
30
31
32
      void set(int i, T v) {
        add(i, v - get(i));
33
34
    };
35
```

#### 6.4 并查集

Listing 93: union-find.cpp

```
#include <vector>
 1
 3
    using namespace std;
 4
 5
    struct DisjointSet {
        vector<int> p;
 6
 7
        void init(int n) {
 8
 9
            p.resize(n);
            for (int i = 0; i < n; ++i) {
10
                p[i] = i;
11
12
        }
13
14
        int getp(int i) {
15
            return i == p[i] ? i : (p[i] = getp(p[i]));
16
17
18
        bool setp(int i, int j) {
19
            i = getp(i);
20
21
            j = getp(j);
22
            p[i] = j;
23
            return i != j;
        }
24
    };
25
```

#### 6.5 轻重权树剖分

Listing 94: chain-decomp.cpp

1 #include <cstdio>

```
#include <vector>
 2
    #include <algorithm>
 3
 4
    using namespace std;
 5
 6
 7
    const int MAXM = 16;
    const int MAXN = 1 << MAXM;
 8
 9
    // Heavy—Light Decomposition
10
    struct TreeDecomposition {
11
      vector<int> e[MAXN], c[MAXN];
12
      int s[MAXN];
                       // subtree size
13
                       // parent id
      int p[MAXN];
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
        p[v] = f;
20
21
        s[v] = 1;
        for (int i = 0; i < (int)e[v].size(); ++i) {</pre>
22
          int w = e[v][i];
23
          if (w != f) {
24
            dfs_(w, v);
25
26
            s[v] += s[w];
27
28
      }
29
30
31
      void decomp_(int v, int f, int k) {
32
        t[v] = ts++;
33
        c[k].push_back(v);
34
        r[v] = k;
35
        int x = 0, y = -1;
36
        for (int i = 0; i < (int)e[v].size(); ++i) {
37
38
          int w = e[v][i];
          if (w != f) {
39
            if (s[w] > x) {
40
               x = s[w];
41
               y = w;
42
43
          }
44
45
        if (y != -1) {
46
          decomp_(y, v, k);
47
48
49
        for (int i = 0; i < (int)e[v].size(); ++i) {
50
          int w = e[v][i];
51
          if (w != f && w != y) {
52
53
            decomp_(w, v, w);
54
55
56
57
58
      void init(int n) {
        for (int i = 0; i < n; ++i) {
59
          e[i].clear();
60
        }
61
      }
62
63
      void add(int a, int b) {
64
        e[a].push_back(b);
65
```

```
66     e[b].push_back(a);
67     }
68
69     void build() { //!!
70         ts = 0;
71         dfs_(0, 0);
72         decomp_(0, 0, 0);
73     }
74     } hld;
```

#### 6.6 强连通分量

Listing 95: scc.cpp

```
#include <algorithm>
 2
    #include <stack>
    #include <vector>
 3
    using namespace std;
 5
 6
 7
    struct SCCTarjan {
 8
        int n;
        vector<vector<int>> e;
 9
10
        vector<int> id;
11
12
        vector<vector<int>> scc;
13
        void init(int n) {
14
15
            this\rightarrown = n;
            vector<vector<int>>(n).swap(e);
16
            id.resize(n);
17
            dfn.resize(n);
18
19
            low.resize(n);
        }
20
21
        void add(int a, int b) {
22
            e[a].push_back(b);
23
24
25
26
        vector<int> dfn, low;
27
        int timestamp;
28
        stack<int> s;
29
30
        void dfs(int v) {
31
            dfn[v] = timestamp++;
            low[v] = dfn[v];
32
            s.push(v);
33
            for (vectorkint>::const_iterator w = e[v].begin(); w != e[v].end(); ++w) {
34
                 if (dfn[*w] == -1) {
35
36
                     dfs(*w);
                     low[v] = min(low[v], low[*w]);
37
                 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
                 do {
45
                     int w = s.top();
46
                     s.pop();
47
                     id[w] = (int)scc.size();
48
                     t.push_back(w);
49
```

```
dfn[w] = -2;
50
                 } while (t.back() != v);
51
                 scc.push_back(t);
52
             }
53
        }
54
55
        int gao() {
56
            scc.clear();
57
58
             stack<int>().swap(s);
59
             timestamp = 0;
60
             fill(dfn.begin(), dfn.end(), -1);
61
             for (int i = 0; i < n; ++i) {
62
                 if (dfn[i] == -1) {
63
                     dfs(i);
64
65
66
             return (int)scc.size();
67
        }
68
    };
69
```

#### 6.7 双连通分量

Listing 96: bcc.cpp

```
#include <algorithm>
 1
 2
    #include <stack>
 3
    #include <utility>
 4
    #include <vector>
    using namespace std;
 6
    // TODO: cannot handle duplicate edges
 8
    struct Tarjan {
 9
10
        int n;
        vector<vector<int>> e;
11
12
        vector<int> cut;
13
        vector<pair<int, int>> bridge;
14
        vector<vector<pair<int, int>>> bcc;
15
16
        void init(int n) {
17
            this\rightarrown = n;
18
            e.clear();
19
            e.resize(n);
20
            dfn.resize(n);
21
            low.resize(n);
22
23
24
        void add(int a, int b) {
25
            // assert(find(e[a].begin(), e[a].end(), b) == e[a].end());
26
            e[a].push_back(b);
27
28
            e[b].push_back(a);
        }
29
30
31
        vector<int> dfn, low;
        int timestamp;
32
        stack<pair<int, int>> s;
33
34
        void dfs(int v, int p) {
35
            int part = p == -1 ? 0 : 1;
36
            dfn[v] = low[v] = timestamp++;
37
            for (vector<int>::const_iterator w = e[v].begin(); w != e[v].end(); ++w) {
38
39
                pair<int, int> f = make_pair(min(v, *w), max(v, *w));
```

```
if (dfn[*w] == -1) {
 40
                      s.push(f);
 41
                      dfs(*w, v);
 42
                      low[v] = min(low[v], low[*w]);
 43
                      if (dfn[v] \leftarrow low[*w]) {
 45
                          // articulation point
                          if (++part == 2) {
 46
                               cut.push_back(v);
 47
 48
                          // articulation edge
 49
                          if (dfn[v] < low[*w]) {
50
                              bridge.push_back(f);
51
52
                          // biconnected component (2—vertex-connected)
53
                          vector<pair<int, int>> t;
 54
55
                               t.push_back(s.top());
 56
                               s.pop();
57
                          } while (t.back() != f);
58
                          bcc.push_back(t);
 59
                      }
60
                  } else if (*w != p \& dfn[*w] < dfn[v]) {
 61
                      s.push(f);
62
                      low[v] = min(low[v], dfn[*w]);
63
64
             }
65
         }
66
 67
         void gao() {
68
             cut.clear();
69
 70
             bridge.clear();
             bcc.clear();
 71
 72
 73
             timestamp = 0;
             stack<pair<int, int>>().swap(s);
 74
             fill(dfn.begin(), dfn.end(), -1);
 75
 76
             for (int i = 0; i < n; ++i) {
 77
                  if (dfn[i] == -1) {
 78
 79
                      dfs(i, -1);
 80
             }
 81
         }
 82
     };
 83
84
     struct BridgeBlockTree {
85
         Tarjan≪MAXN bcc;
86
         DisjointSet<MAXN⊳ ds;
87
         vector<int> e[MAXN];
88
89
         void init(int n) {
90
             bcc.init(n);
91
 92
             ds.init(n);
         }
 93
 94
         void add(int a, int b) {
 95
             bcc.add(a, b);
96
         }
97
98
         void gao() {
99
             bcc.gao();
100
             for (const auto &i : bcc.bcc) {
101
                  if (i.size() > 1) {
102
```

```
for (const auto &j : i) {
103
                          ds.setp(j.first, j.second);
104
105
                  }
106
             }
107
             for (const auto &i : bcc.bridge) {
108
                  int a = ds.getp(i.first);
109
                  int b = ds.getp(i.second);
110
111
                  e[a].push_back(b);
                  e[b].push_back(a);
112
             }
113
         }
114
115
         int id(int v) {
116
             return ds.getp(v);
117
118
     };
119
```

## 6.8 二分图匹配

#### Listing 97: bimatch.cpp

```
// maximum matchings in bipartite graphs
    // maximum cardinality bipartite matching
    // O(|V||E|), generally fast
 3
 4
 5
    #include <algorithm>
 6
    #include <string>
 7
    #include <vector>
 8
    using namespace std;
 9
10
    struct Hungarian {
11
        int nx, ny;
12
        vector<int> mx, my;
13
        vector<vector<int>> e;
14
15
        void init(int nx, int ny) {
16
             this\rightarrownx = nx;
17
             this->ny = ny;
18
            mx.resize(nx);
19
            my.resize(ny);
20
             e.clear();
21
             e.resize(nx);
22
            mark.resize(nx);
23
24
25
        void add(int a, int b) {
26
27
             e[a].push_back(b);
28
29
        // vector<bool> is evil!!!
30
        basic_string\textbool> mark;
31
32
        bool augment(int i) {
33
34
             if (!mark[i]) {
35
                 mark[i] = true;
                 for (vectorkint>::const_iterator j = e[i].begin(); j != e[i].end(); ++j) {
36
                     if (my[*j] == -1 \mid | augment(my[*j])) {
37
                         mx[i] = *j;
38
                         my[*j] = i;
39
40
                         return true;
41
                     }
                 }
42
```

```
43
             return false;
44
        }
45
46
        int gao() {
47
48
             int ret = 0;
             fill(mx.begin(), mx.end(), -1);
49
             fill(my.begin(), my.end(), -1);
50
             for (int i = 0; i < nx; ++i) {
51
                 fill(mark.begin(), mark.end(), false);
52
53
                 if (augment(i)) {
54
                     ++ret;
55
             }
56
57
             return ret;
        }
58
    };
59
```

# 6.9 最小费用最大流

Listing 98: flow.cpp

```
#include <algorithm>
 1
    #include <cstdio>
 2
    #include <limits>
 3
    #include <queue>
 5
    #include <vector>
 7
    using namespace std;
 8
    template <int MAXN, typename T = int, typename S = T>
 9
10
    struct MinCostMaxFlow {
        struct NegativeCostCircuitExistsException {
11
12
        };
13
14
        struct Edge {
15
            int v;
16
            Tc;
            S w;
17
            int b;
18
            Edge(int v, T c, S w, int b) : v(v), c(c), w(w), b(b) {}
19
        };
20
21
        int n, source, sink;
22
        vector<Edge> e[MAXN];
23
24
        void init(int n, int source, int sink) {
25
            this\rightarrown = n;
26
            this—>source = source;
27
            this—>sink = sink;
28
            for (int i = 0; i < n; ++i) {
29
                 e[i].clear();
30
31
        }
32
33
        void addEdge(int a, int b, T c, S w) {
34
            e[a].push_back(Edge(b, c, w, e[b].size()));
35
            e[b].push_back(Edge(a, 0, -w, e[a].size() -1); // TODO
36
        }
37
38
        bool mark[MAXN];
39
        T maxc[MAXN];
40
41
        S minw[MAXN];
42
        int dist[MAXN];
```

```
Edge *prev[MAXN];
43
44
         bool spfa() {
45
46
              queu≪int> q;
              fill(mark, mark + n, false);
47
              fill(maxc, maxc + n, 0);
48
49
              fill(minw, minw + n, numeric_limits<S>::max());
50
              fill(dist, dist + n, 0);
              fill(prev, prev + n, (Edge *)NULL);
51
              mark[source] = true;
52
              maxc[source] = numeric_limits<S>::max();
53
              minw[source] = 0;
54
55
              q.push(source);
56
              while (!q.empty()) {
57
                  int cur = q.front();
58
                  mark[cur] = false;
59
60
                  q.pop();
                  for (typename vector<Edge>::iterator it = e[cur].begin(); it != e[cur].end(); ++it) {
61
                       T c = min(maxc[cur], it \rightarrow c);
62
                       if (c == 0) {
63
64
                           continue;
                       }
65
66
                       int v = it -> v;
67
                       S w = minw[cur] + it \rightarrow w;
68
                       if (\min w[v] > w \mid | (\min w[v] == w \&\& \max c[v] < c)) { // TODO}
69
                           maxc[v] = c;
70
                           minw[v] = w;
71
                           dist[v] = dist[cur] + 1;
72
                           if (dist[v] >= n) {
73
74
                                return false;
75
                           prev[v] = \delta*it;
76
                           if (!mark[v]) {
77
                                mark[v] = true;
78
79
                                q.push(v);
                           }
80
                       }
81
                  }
82
83
              return true;
84
         }
85
86
         pair<T, S> gao() {
87
              T sumc = 0;
88
              S sumw = 0;
89
              while (true) {
90
                  if (!_spfa()) {
91
                       throw NegativeCostCircuitExistsException();
92
                   } else if (maxc[sink] == 0) {
93
                       break;
94
                   } else {
95
                       T c = maxc[sink];
96
97
                       sumc += c;
                       sumw += c * minw[sink];
98
99
                       int cur = sink;
100
                       while (cur != source) {
101
102
                           Edge *e1 = prev[cur];
                           e1->c -= c;
103
                           Edge \stare2 = \deltae[e1\rightarrowv][e1\rightarrowb];
104
                           e2->c += c;
105
                           cur = e2 \rightarrow v;
106
```

## 6.10 AhoCorasick 自动机

#### Listing 99: ac-automata.cpp

```
#include <algorithm⊳
    #include <queue>
 2
 3
 4
    using namespace std;
 5
    struct AhoCorasick {
 6
        static const int NONE = 0;
 8
        static const int MAXN = 1024;
        static const int CHARSET = 26;
 9
10
        int end;
11
        int tag[MAXN];
12
        int fail[MAXN];
13
        int trie[MAXN][CHARSET];
14
15
        void init() {
16
            tag[0] = NONE;
17
            fill(trie[0], trie[0] + CHARSET, -1);
18
            end = 1;
19
        }
20
21
        int add(int m, const int *s) {
22
            int p = 0;
23
            for (int i = 0; i < m; ++i) {
24
25
                 if (trie[p][*s] == -1) {
26
                     tag[end] = NONE;
                     fill(trie[end], trie[end] + CHARSET, -1);
27
                     trie[p][*s] = end++;
28
                 }
29
30
                p = trie[p][*s];
31
                ++S;
32
33
            return p;
34
35
        void build(void) { // !!
36
            queu≪int> bfs;
37
            fail[0] = 0;
38
            for (int i = 0; i < CHARSET; ++i) {
39
                 if (trie[0][i] != -1) {
40
                     fail[trie[0][i]] = 0;
41
                     bfs.push(trie[0][i]);
42
43
                     trie[0][i] = 0;
44
                 }
45
46
            while (!bfs.empty()) {
47
                int p = bfs.front();
48
                tag[p] |= tag[fail[p]];
49
                bfs.pop();
50
                for (int i = 0; i < CHARSET; ++i) {
51
                     if (trie[p][i] != -1) {
52
53
                         fail[trie[p][i]] = trie[fail[p]][i];
```

#### 6.11 后缀数组

Listing 100: sa.cpp

```
#include <algorithm>
 1
    #include <utility>
 3
    #include <vector>
 4
    using namespace std;
 5
    struct SuffixArray {
 6
        vectorint> sa, rank, height;
 7
 8
        template <typename T>
 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) {
15
                assoc[i] = make_pair(a[i], i);
16
17
            sort(assoc.begin(), assoc.end());
18
            for (int i = 0; i < n; ++i) {
19
                sa[i] = assoc[i].second;
20
                if (i == 0 \mid | assoc[i].first != assoc[i - 1].first) {
21
                     rank[sa[i]] = i;
22
                } else {
23
                    rank[sa[i]] = rank[sa[i-1]];
24
                 }
25
            }
26
27
            vector<int> tmp(n), cnt(n);
28
            vector<pair<int, int>> suffix(n);
29
            for (int m = 1; m < n; m <<= 1) {
30
                // snd
31
                for (int i = 0; i < m; ++i) {
32
33
                    tmp[i] = n - m + i;
34
                for (int i = 0, j = m; i < n; ++i) {
35
                     if (sa[i] >= m) {
36
                         tmp[j++] = sa[i] - m;
37
38
39
                // fst
40
                fill(cnt.begin(), cnt.end(), 0);
41
                for (int i = 0; i < n; ++i) {
42
                     ++cnt[rank[i]];
43
44
                partial_sum(cnt.begin(), cnt.end(), cnt.begin());
45
                for (int i = n - 1; i \ge 0; —i) {
46
                     sa[—cnt[rank[tmp[i]]]] = tmp[i];
47
48
                //
49
                for (int i = 0; i < n; ++i) {
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) {
53
                     if (i == 0 || suffix[sa[i]] != suffix[sa[i - 1]]) {
54
                         rank[sa[i]] = i;
55
                     } else {
56
                         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) {
63
                 if (rank[i] == 0) {
64
                     height[0] = z = 0;
65
                 } else {
66
                     int x = i, y = sa[rank[i] - 1];
67
                     z = max(0, z - 1);
68
                     while (x + z < n \& y + z < n \& a[x + z] == a[y + z]) {
69
70
                         ++Z;
71
                     height[rank[i]] = z;
72
                }
73
            }
74
        }
75
    };
76
```

# 6.12 LU 分解

### Listing 101: lu.cpp

```
const int MAXN = 128;
 1
    const double EPS = 1e-10;
 2
 3
    void LU(int n, double a[MAXN][MAXN], int r[MAXN], int c[MAXN]) {
 4
 5
        for (int i = 0; i < n; ++i) {
 6
            r[i] = c[i] = i;
 7
 8
        for (int k = 0; k < n; ++k) {
 9
            int ii = k, jj = k;
            for (int i = k; i < n; ++i) {
10
                for (int j = k; j < n; ++j) {
11
                     if (fabs(a[i][j]) > fabs(a[ii][jj])) {
12
                         ii = i;
13
                         jj = j;
14
                     }
15
                }
16
            }
17
            swap(r[k], r[ii]);
18
            swap(c[k], c[jj]);
19
            for (int i = 0; i < n; ++i) {
20
                swap(a[i][k], a[i][jj]);
21
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) {
27
                continue;
28
29
            for (int i = k + 1; i < n; ++i) {
30
                a[i][k] = a[i][k] / a[k][k];
                for (int j = k + 1; j < n; ++j) {
31
32
                    a[i][j] = a[i][k] * a[k][j];
33
            }
34
```

```
}
35
    }
36
37
    void solve(int n, double a[MAXN][MAXN], int r[MAXN], int c[MAXN], double b[MAXN]) {
38
        static double x[MAXN];
39
        for (int i = 0; i < n; ++i) {
40
            x[i] = b[r[i]];
41
42
        for (int i = 0; i < n; ++i) {
43
            for (int j = 0; j < i; ++j) {
44
                x[i] = a[i][j] * x[j];
45
46
        }
47
        for (int i = n - 1; i \ge 0; —i) {
48
            for (int j = n - 1; j > i; —j) {
49
                x[i] = a[i][j] * x[j];
50
51
            if (fabs(a[i][i]) >= EPS) {
52
                x[i] /= a[i][i];
53
            } // else assert(fabs(x[i]) < EPS);</pre>
54
        }
55
        for (int i = 0; i < n; ++i) {
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