

Contents

1. basic	2
1.1. pbds.h	2
2. ds	3
2.1. dsu.h	3
2.2. segTree.h	4
2.3. twoDimPrfxSum.h	9
3. geo	11
3.1. Rotating_Calipers.h	11
4. graph	13
4.1. Flow	13
4.1.1. max_Flow.h	13
4.1.2. min_Cost.h	16
4.2. Tree	19
4.2.1. lca.h	19
5. math	20
5.1. number_theory	20
5.1.1. Comb.h	20
5.1.2. Euler_sieve.h	21
5.1.3. factor_pr.h	22
5.2. other	26
5.2.1. Frac.h	26
6. string	28
6.1. compress_print.h	28
6.2. get_occ.h	29
6.3. hash_print.h	30
6.4. KMP.h	32
6.5. trie_Tree.h	33

1. basic

1.1. pbds.h

```

1  #include <ext/pb_ds/assoc_container.hpp>
2  #include <ext/pb_ds/tree_policy.hpp>
3  using namespace __gnu_pbds;
4  using namespace std;
5  using ord_set = tree<int, null_type, less<int>, rb_tree_tag,
6  tree_order_statistics_node_update>;
7  using ord_mset = tree<int, null_type, less_equal<int>, rb_tree_tag,
8  tree_order_statistics_node_update>;
9  //find_by_order
10 //order_of_key

```

2. ds

2.1. dsu.h

```

1  class DSU {
2      std::vector<int> f, siz;
3  public:
4      DSU() {}
5      DSU(int n) {
6          init(n);
7      }
8
9      void init(int n) {
10         f.resize(n);
11         for(int i = 0; i < n; i++) f[i] = i;
12         siz.assign(n, 1);
13     }
14
15     int find(int x) {
16         while (x != f[x]) {
17             x = f[x] = f[f[x]];
18         }
19         return x;
20     }
21
22     bool same(int x, int y) {
23         return find(x) == find(y);
24     }
25
26     bool merge(int x, int y) {
27         x = find(x);
28         y = find(y);
29         if (x == y) {
30             return false;
31         }
32         siz[x] += siz[y];
33         f[y] = x;
34         return true;
35     }
36
37     int size(int x) {
38         return siz[find(x)];
39     }
40 };

```

2.2. segTree.h

```

1 // AC 带懒惰标记线段树
2 template <class TYPE_NAME>
3 class lazyTree
4 {
5     /*
6     * TYPE_NAME 需要支持: + += != 和自定义的合并运算符
7     * 实现了懒惰标记, 仅支持区间批量增加
8     * 区间批量减需要 TYPE_NAME 支持-, 且有 -a = 0 - a
9     * 额外处理了一个单点修改为对应值的函数, 非原生实现, 其复杂度为 4logn
10    * 不提供在线
11    * 不提供持久化
12    */
13 private:
14     vector<TYPE_NAME> d;
15     vector<TYPE_NAME> a;
16     vector<TYPE_NAME> b;
17     int n;
18     const TYPE_NAME INI = 0; // 不会影响合并运算的初始值, 如 max 取 INF, min 取 0,
19     mti 取 1
20     void subbuild(int s, int t, int p)
21     {
22         if (s == t)
23         {
24             d[p] = a[s];
25             return;
26         }
27         int m = s + ((t - s) >> 1); // (s+t)/2
28         subbuild(s, m, p * 2);
29         subbuild(m + 1, t, p * 2 + 1);
30         d[p] = d[p * 2] + d[p * 2 + 1];
31         // 合并运算符 ↑
32     }
33     TYPE_NAME subGetSum(int l, int r, int s, int t, int p)
34     {
35         if (l <= s && t <= r)
36             return d[p];
37         int m = s + ((t - s) >> 1);
38         if (b[p] != 0)
39         {
40             d[p * 2] += b[p] * (m - s + 1); // 合并运算符的高阶运算 此处运算为应
41             用懒惰标记
42             d[p * 2 + 1] += b[p] * (t - m); // 合并运算符的高阶运算 此处运算为应
43             用懒惰标记
44             b[p * 2] += b[p]; // 下传标记, 无需修改
45             b[p * 2 + 1] += b[p]; // 下传标记, 无需修改
46             b[p] = 0;
47         }
48         TYPE_NAME ans1 = INI;
49         TYPE_NAME ansr = INI;
50         if (l <= m)
51             ans1 = subGetSum(l, r, s, m, p * 2);
52         if (r > m)
53             ansr = subGetSum(l, r, m + 1, t, p * 2 + 1);

```

```

53         return ans1 + ansr;
54         // 合并运算符↑
55     }
56
57     void subUpdate(int l, int r, TYPE_NAME c, int s, int t, int p)
58     {
59         if (l <= s && t <= r)
60         {
61             d[p] += (t - s + 1) * c; // 合并运算符的高阶运算 此处运算为修改整匹配
62             b[p] += c;                // 记录懒惰标记, 无需修改
63             return;
64         }
65         int m = s + ((t - s) >> 1);
66         if (b[p] != 0 && s != t)
67         {
68             d[p * 2] += b[p] * (m - s + 1); // 合并运算符的高阶运算 此处运算为应
69             d[p * 2 + 1] += b[p] * (t - m); // 合并运算符的高阶运算 此处运算为应
70             b[p * 2] += b[p];                // 下传标记, 无需修改
71             b[p * 2 + 1] += b[p];            // 下传标记, 无需修改
72             b[p] = 0;
73         }
74         if (l <= m)
75             subUpdate(l, r, c, s, m, p * 2);
76         if (r > m)
77             subUpdate(l, r, c, m + 1, t, p * 2 + 1);
78         d[p] = d[p * 2] + d[p * 2 + 1];
79         // 合并运算符 ↑
80     }
81
82     public:
83         lazyTree(TYPE_NAME _n)
84         {
85             n = _n;
86             d.resize(4 * n + 5);
87             a.resize(4 * n + 5);
88             b.resize(4 * n + 5);
89         }
90
91         void build(vector<TYPE_NAME> _a)
92         {
93             a = _a;
94             subbuild(1, n, 1);
95         }
96
97         TYPE_NAME getsum(int l, int r)
98         {
99             return subGetSum(l, r, 1, n, 1);
100         }
101
102         void update(int l, int r, TYPE_NAME c) // 修改区间
103         {
104             subUpdate(l, r, c, 1, n, 1);
105         }
106

```

```

107     void update(int idx, TYPE_NAME tar)
108     { // 修改单点, 未测试
109         TYPE_NAME tmp = getsum(idx, idx);
110         tar -= tmp;
111         subUpdate(idx, idx, tar, 1, n, 1);
112     }
113 };
114
115 //AC MJ 的 MIN/MAX 树
116 template<class Info>
117 struct SegmentTree {
118     int n;
119     std::vector<Info> info;
120     SegmentTree() : n(0) {}
121     SegmentTree(int n_, Info v_ = Info()) {
122         init(n_, v_);
123     }
124     template<class T>
125     SegmentTree(std::vector<T> init_) {
126         init(init_);
127     }
128     void init(int n_, Info v_ = Info()) {
129         init(std::vector(n_, v_));
130     }
131     template<class T>
132     void init(std::vector<T> init_) {
133         n = init_.size();
134         info.assign(4 << std::__lg(n), Info());
135         std::function<void(int, int, int)> build = [&](int p, int l, int r) {
136             if (r - l == 1) {
137                 info[p] = init_[l];
138                 return;
139             }
140             int m = (l + r) / 2;
141             build(2 * p, l, m);
142             build(2 * p + 1, m, r);
143             pull(p);
144         };
145         build(1, 0, n);
146     }
147     void pull(int p) {
148         info[p] = info[2 * p] + info[2 * p + 1];
149     }
150     void modify(int p, int l, int r, int x, const Info &v) {
151         if (r - l == 1) {
152             info[p] = v;
153             return;
154         }
155         int m = (l + r) / 2;
156         if (x < m) {
157             modify(2 * p, l, m, x, v);
158         } else {
159             modify(2 * p + 1, m, r, x, v);
160         }
161         pull(p);
162     }
163     void modify(int p, const Info &v) {

```

```

164     modify(1, 0, n, p, v);
165 }
166 Info rangeQuery(int p, int l, int r, int x, int y) {
167     if (l >= y || r <= x) {
168         return Info();
169     }
170     if (l >= x && r <= y) {
171         return info[p];
172     }
173     int m = (l + r) / 2;
174     return rangeQuery(2 * p, l, m, x, y) + rangeQuery(2 * p + 1, m, r, x,
175 y);
176 }
177 Info rangeQuery(int l, int r) {
178     return rangeQuery(1, 0, n, l, r);
179 }
180 template<class F>
181 int findFirst(int p, int l, int r, int x, int y, F &&pred) {
182     if (l >= y || r <= x) {
183         return -1;
184     }
185     if (l >= x && r <= y && !pred(info[p])) {
186         return -1;
187     }
188     if (r - l == 1) {
189         return l;
190     }
191     int m = (l + r) / 2;
192     int res = findFirst(2 * p, l, m, x, y, pred);
193     if (res == -1) {
194         res = findFirst(2 * p + 1, m, r, x, y, pred);
195     }
196     return res;
197 }
198 template<class F>
199 int findFirst(int l, int r, F &&pred) {
200     return findFirst(1, 0, n, l, r, pred);
201 }
202 template<class F>
203 int findLast(int p, int l, int r, int x, int y, F &&pred) {
204     if (l >= y || r <= x) {
205         return -1;
206     }
207     if (l >= x && r <= y && !pred(info[p])) {
208         return -1;
209     }
210     if (r - l == 1) {
211         return l;
212     }
213     int m = (l + r) / 2;
214     int res = findLast(2 * p + 1, m, r, x, y, pred);
215     if (res == -1) {
216         res = findLast(2 * p, l, m, x, y, pred);
217     }
218     return res;
219 }
220 template<class F>

```

```

220     int findLast(int l, int r, F &&pred) {
221         return findLast(1, 0, n, l, r, pred);
222     }
223 };
224 const int inf = 1E9;
225 struct Info
226 {
227     int mn {inf}, mnId, mx {-inf}, mxId;
228 } ;
229 Info operator+(Info a, Info b)
230 {
231     if (a.mn > b.mn)
232         a.mn = b.mn, a.mnId = b.mnId;
233     if (a.mx < b.mx)
234         a.mx = b.mx, a.mxId = b.mxId;
235     return a;
236 }

```


2.3. twoDimPrfxSum.h

```

1  class prfx_2{
2  public:
3      vector<vector<int>> mtx;
4      int n,m;
5      public:
6      prfx_2(vector<vector<int>> _mtx){init(_mtx);};
7      prfx_2() { };
8      void init(vector<vector<int>> _mtx)
9      {
10         n = _mtx.size();
11         m = _mtx[0].size();
12         mtx.resize(n+1);
13         for(auto &x:mtx) x.resize(m+1);
14         lop(i,1,n+1)
15             lop(j,1,m+1)
16                 _mtx[i][j] = mtx[i-1][j] + mtx[i][j-1] - mtx[i-1][j-1] +
17             }
18
19         int getsum(int x1,int y1,int x2,int y2)
20         {
21             x1 ++,x2 ++,y1 ++,y2 ++;
22             return mtx[x2][y2] - mtx[x1-1][y2] - mtx[x2][y1-1] + mtx[x1-1][y1-1];
23         }
24
25         int getsum(pii d1,pii d2)
26         {
27             auto [x1,y1] = d1;
28             auto [x2,y2] = d2;
29             x1 ++,x2 ++,y1 ++,y2 ++;
30             return mtx[x2][y2] - mtx[x1-1][y2] - mtx[x2][y1-1] + mtx[x1-1][y1-1];
31         }
32
33         vector<int> & operator[](std::size_t i)
34         {
35             return mtx[i];
36         }
37     };
38
39
40     class conj_diff_2{
41         vector<vector<int>> mtx;
42         vector<vector<int>> prf;
43         int n,m;
44         public:
45
46         conj_diff_2(int _n,int _m)
47         {
48             n = _n+1,m = _m+1;
49             vector<vector<int>> initmp(n,vector<int> (m,0));
50             init(initmp);
51         }
52
53         conj_diff_2(vector<vector<int>> _mtx)
54         {
55             this->init(_mtx);
56         }

```

```

57
58     conj_diff_2(){ }
59
60     void init(vector<vector<int>> _mtx)
61     {
62         n = _mtx.size();
63         m = _mtx[0].size();
64         mtx.resize(n+2);
65         for(auto &x:mtx) x.resize(m+2);
66         prf.resize(n+2);
67         for(auto &x:prf) x.resize(m+2);
68         lop(i,1,n+1)
69             lop(j,1,m+1)
70                 prf[i][j] = _mtx[i-1][j-1];
71     }
72
73     void modify(int x1,int y1,int x2,int y2,int k)
74     {
75         x1 ++,x2 ++,y1 ++,y2 ++;
76         mtx[x1][y1] += k;
77         mtx[x2+1][y1] -= k,mtx[x1][y2+1] -= k;
78         mtx[x2+1][y2+1] += k;
79     }
80
81     void modify(pii d1,pii d2,int k)
82     {
83         this->modify(d1.fi,d1.se,d2.fi,d2.se,k);
84     }
85
86     vector<vector<int>> cacu()
87     {
88         auto res = prfx_2(mtx);
89         vector<vector<int>> rst(n,vector<int>(m));
90         lop(i,1,n+1)
91             lop(j,1,m+1)
92                 rst[i-1][j-1] = prf[i][j] + res[i+1][j+1];
93         return rst;
94     }
95
96     vector<int> & operator[](std::size_t i)
97     {
98         return mtx[i];
99     }
100 };

```

3. geo

3.1. Rotating_Calipers.h

```

1 //Rotating_Calipers
2 template<typename VALUE_TYPE>
3 class Rotating_Calipers
4 {
5 public:
6     using vec_pv = vector<pair<VALUE_TYPE, VALUE_TYPE>>;
7     vec_pv p;
8
9     static VALUE_TYPE cross(pair<VALUE_TYPE, VALUE_TYPE> p1, pair<VALUE_TYPE,
10 VALUE_TYPE> p2, pair<VALUE_TYPE, VALUE_TYPE> p0)
11     {
12         pair<VALUE_TYPE, VALUE_TYPE>
13         t1 = {p1.fi - p0.fi, p1.se - p0.se},
14         t2 = {p2.fi - p0.fi, p2.se - p0.se};
15         return t1.fi * t2.se - t1.se * t2.fi;
16     }
17
18     static VALUE_TYPE dis(const pair<VALUE_TYPE, VALUE_TYPE> &p1, const
19 pair<VALUE_TYPE, VALUE_TYPE> &p2){
20         return (p1.fi - p2.fi) * (p1.fi - p2.fi) + (p1.se - p2.se) * (p1.se -
21 p2.se);
22     };
23 public:
24
25     Rotating_Calipers() {}
26
27     Rotating_Calipers(vec_pv _A) {
28         build(_A);
29     }
30
31     void build(const vec_pv & _A) {
32         p = ConvexHull(_A);
33     }
34
35     static vec_pv ConvexHull(vec_pv A, VALUE_TYPE flag = 1)
36     {
37         int n = A.size();
38         if (n <= 2) return A;
39         vec_pv ans(n * 2);
40         sort(A.begin(), A.end(),
41             [](pair<VALUE_TYPE,VALUE_TYPE> a,pair<VALUE_TYPE,VALUE_TYPE> b) ->
42             bool {
43                 if(fabs(a.fi - b.fi) < 1e-10)
44                     return a.se < b.se;
45                 else return a.fi < b.fi;});
46         int now = -1;
47         for (int i = 0; i < n; i++)
48         { // 维护下凸包
49             while (now > 0 && cross(A[i], ans[now], ans[now - 1]) < flag)
50                 ans[+now] = A[i];
51             int pre = now;

```

```

50     for (int i = n - 2; i >= 0; i--)
51     { // 维护上凸包
52         while (now > pre && cross(A[i], ans[now], ans[now - 1]) < flag)
53             now--;
54         ans[++now] = A[i];
55     }
56     ans.resize(now);
57     return ans;
58 }
59 VALUE_TYPE getDiameter() {
60     int j = 1;
61     VALUE_TYPE ans = 0;
62     int m = p.size();
63     p.push_back(p[0]);
64     for(int i = 0; i < m; i++)
65     {
66         while( cross(p[i+1], p[j], p[i]) > cross(p[i+1], p[j+1], p[i]) ) j =
67             (j+1)%m;
68         ans = max(ans, max( dis(p[i], p[j]) , dis(p[i+1], p[j]) ) );
69     }
70     p.pop_back();
71     return ans;
72 }
73 VALUE_TYPE getPerimeter() {
74     VALUE_TYPE sum = 0;
75     p.pb(p[0]);
76     for(int i = 0; i < (int)p.size() - 1; i++)
77     {
78         sum += sqrtl(dis(p[i], p[i+1]));
79     }
80     p.pop_back();
81     return sum;
82 }
83
84 };

```

4. graph

4.1. Flow

4.1.1. max_Flow.h

```

1  class maxFlow//AC
2  {
3  private:
4      class edge
5      {
6      public:
7          ll int nxt,           // 出度
8              cap,             // 容量
9              flow;            // 流量
10         pair<int, int> revNodeIdx; // 反向边
11     public:
12         edge(int _nxt, int _cap)
13         {
14             nxt = _nxt;
15             cap = _cap;
16             flow = 0;
17         }
18         void setRevIdx(int _i, int _j)
19         {
20             revNodeIdx.first = _i;
21             revNodeIdx.second = _j;
22         }
23     };
24     vector<vector<edge>> edgeList; // 节点列表
25     vector<int> dep;              // 深度
26     vector<int> fir;              // 节点最近合法边
27     ll int maxFlowAns;
28
29     int T, S;
30
31     public:
32     maxFlow(int _n)
33     {
34         maxFlowAns = 0;
35         S = 1;
36         T = _n;
37         edgeList.resize(_n + 1);
38         dep.resize(_n + 1);
39         fir.resize(_n+1);
40     }
41
42     void resetTS(int _T, int _S)
43     {
44         T = _T;
45         S = _S;
46     }
47
48     void addedge(int _u, int _v, int _w)
49     {
50         edgeList[_u].push_back(edge(_v, _w));
51         edgeList[_v].push_back(edge(_u, 0)); // 反向建边

```

```

52     edgeList[_u][edgeList[_u].size() - 1].setRevIdx(_v,
edgeList[_v].size() - 1);
53     edgeList[_v][edgeList[_v].size() - 1].setRevIdx(_u,
edgeList[_u].size() - 1);
54 }
55
56 bool bfs() // 统计深度
57 {
58     queue<int> que;
59     for (auto x = dep.begin(); x != dep.end(); x++)
60         (*x) = 0;
61
62     dep[S] = 1;
63     que.push(S);
64     while (que.size())
65     {
66         ll int at = que.front();
67         que.pop();
68         for (int i = 0; i < edgeList[at].size(); i++)
69         {
70             auto tar = edgeList[at][i];
71             if ((!dep[tar.nxt]) && (tar.flow < tar.cap))
72             {
73                 dep[tar.nxt] = dep[at] + 1;
74                 que.push(tar.nxt);
75             }
76         }
77     }
78     return dep[T];
79 }
80
81 ll int dfs(int at, ll int flow)
82 {
83     if ((at == T) || (!flow))
84         return flow; // 到了或者没了
85     ll int ret = 0; // 本节点最大流
86     for (int &i = fir[at]; i < edgeList[at].size(); i++)
87     {
88         auto tar = edgeList[at][i]; // 目前遍历的边
89         int t1Flow = 0; // 目前边的最大流
90         if (dep[at] == dep[tar.nxt] - 1) // 遍历到的边为合法边
91         {
92             t1Flow = dfs(tar.nxt, min((ll)tar.cap - tar.flow, flow -
ret));
93             if (!t1Flow)
94                 continue; // 若最大流
// 为 0, 什么都不做
95             ret += t1Flow; // 本节点最大流
// 累加
96             edgeList[at][i].flow += t1Flow; // 本节点实时流量
97             edgeList[tar.revNodeIdx.first][tar.revNodeIdx.second].flow -= t1Flow; // 反向边流量
98             if (ret == flow)

```

```

99         return ret; // 充满了就不继续扫了
100     }
101 }
102 return ret;
103 }
104
105 11 int dinic()
106 {
107     if (maxFlowAns)
108         return maxFlowAns;
109     while (bfs())
110     {
111         for(auto x = fir.begin(); x != fir.end(); x++) (*x) = 0;
112         maxFlowAns += dfs(S, INT_MAX);
113     }
114     return maxFlowAns;
115 }
116 };

```

4.1.2. min_Cost.h

```

1  const int INF = 0x3f3f3f3f
2
3  class PD//AC
4  {
5  public:
6      class edge
7      {
8      public:
9          int v, f, c, next;
10         edge(int _v,int _f,int _c,int _next)
11         {
12             v = _v;
13             f = _f;
14             c = _c;
15             next = _next;
16         }
17         edge() { }
18     } ;
19
20     void vecset(int value,vector<int> &arr)
21     {
22         for(int i = 0;i < arr.size();i ++) arr[i] = value;
23         return;
24     }
25
26     class node
27     {
28     public:
29         int v, e;
30     } ;
31
32     class mypair
33     {
34     public:
35         int dis, id;
36
37         bool operator<(const mypair &a) const { return dis > a.dis; }
38
39         mypair(int d, int x)
40         {
41             dis = d;
42             id = x;
43         }
44     };
45
46     vector<int> head;
47     vector<int> dis;
48     vector<int> vis;
49     vector<int> h;
50     vector<edge> e;
51     vector<node> p;
52     int n, m, s, t, cnt = 1, maxf, minc;
53
54     PD(int _n,int _m,int _s,int _t)
55     {
56         n = _n;
57         m = _m;
58         s = _s;

```



```

59     t = _t;
60     maxf = 0;
61     minc = 0;
62     head.resize(n+2);
63     dis.resize(n+2);
64     vis.resize(n+2);
65     e.resize(2);
66     h.resize(n+2);
67     p.resize(m+2);
68 }
69
70 void addedge(int u, int v, int f, int c)
71 {
72     e.push_back(edge(v,f,c,head[u]));
73     head[u] = e.size()-1;
74     e.push_back(edge(u,0,-c,head[v]));
75     head[v] = e.size()-1;
76 }
77
78 bool dijkstra()
79 {
80     priority_queue<mypair> q;
81     vecset(INF,dis);
82     vecset(0,vis);
83     dis[s] = 0;
84     q.push(mypair(0, s));
85     while (!q.empty())
86     {
87         int u = q.top().id;
88         q.pop();
89         if (vis[u])
90             continue;
91         vis[u] = 1;
92         for (int i = head[u]; i; i = e[i].next)
93         {
94             int v = e[i].v, nc = e[i].c + h[u] - h[v];
95             if (e[i].f && dis[v] > dis[u] + nc)
96             {
97                 dis[v] = dis[u] + nc;
98                 p[v].v = u;
99                 p[v].e = i;
100                 if (!vis[v])
101                     q.push(mypair(dis[v], v));
102             }
103         }
104     }
105     return dis[t] != INF;
106 }
107
108 void spfa()
109 {
110     queue<int> q;
111     vecset(63,h);
112     h[s] = 0, vis[s] = 1;
113     q.push(s);
114     while (!q.empty())
115     {
116         int u = q.front();

```

```

117         q.pop();
118         vis[u] = 0;
119         for (int i = head[u]; i; i = e[i].next)
120         {
121             int v = e[i].v;
122             if (e[i].f && h[v] > h[u] + e[i].c)
123             {
124                 h[v] = h[u] + e[i].c;
125                 if (!vis[v])
126                 {
127                     vis[v] = 1;
128                     q.push(v);
129                 }
130             }
131         }
132     }
133 }
134
135 int pd()
136 {
137     spfa();
138     while (dijkstra())
139     {
140         int minf = INF;
141         for (int i = 1; i <= n; i++)
142             h[i] += dis[i];
143         for (int i = t; i != s; i = p[i].v)
144             minf = min(minf, e[p[i].e].f);
145         for (int i = t; i != s; i = p[i].v)
146         {
147             e[p[i].e].f -= minf;
148             e[p[i].e ^ 1].f += minf;
149         }
150         maxf += minf;
151         minc += minf * h[t];
152     }
153     return 0;
154 }
155
156 void printAns()
157 {
158     cout << maxf << " " << minc << "\n";
159 }
160 };

```

4.2. Tree

4.2.1. lca.h

```

1  class LCA{
2  public:
3      vector<vector<int>> cnj;
4      vector<int> lg,dep;
5      vector<array<int,32>> fa;
6      int n;
7
8      LCA(int _n) {
9          n = _n;
10         cnj.resize(n+1);
11         lg.resize(n+1),fa.resize(n+1),dep.resize(n+1);
12         for(int i = 1; i <= n; i++)
13             lg[i] = lg[i-1] + (1 << lg[i-1] == i);
14     }
15
16     void addEdge(int u,int v) {
17         cnj[u].push_back(v);
18         cnj[v].push_back(u);
19     }
20
21     void build(int rt = 1) {
22         using itf = function<void(int,int)>;
23         itf dfs = [&](int p,int f) -> void {
24             fa[p][0] = f,dep[p] = dep[f] + 1;
25             for(int i = 1;i <= lg[dep[p]];i++) fa[p][i] = fa[fa[p][i-1]]
26 [i-1];
27             for(auto x:cnj[p]) if(x == f) continue;
28             else dfs(x,p);
29         };
30         dfs(rt,0);
31     }
32
33     int get(int x,int y) {
34         if(dep[x] < dep[y]) swap(x,y);
35         while(dep[x] > dep[y]) x = fa[x][lg[dep[x]] - dep[y] - 1];
36         if(x == y) return x;
37         for(int k = lg[dep[x]]-1;k >= 0;k--) if(fa[x][k] != fa[y][k]) x =
38 fa[x][k],y = fa[y][k];
39         return fa[x][0];
40     }
41 };

```

5. math

5.1. number_theory

5.1.1. Comb.h

```

1  #include <template_overAll.h>
2
3  const int N = 1e6;
4  const int mod = 1e9+7;
5
6  int binpow(int x, int y)
7  {
8      int ans = 1;
9      while (y)
10     {
11         if (y & 1) ans = ans * x % mod;
12         x = x * x % mod;
13         y >>= 1;
14     }
15     return ans;
16 }
17
18 vector<int> fac(N), inv(N);
19
20 void init()
21 {
22     fac[0] = inv[0] = 1;
23     for (int i = 1; i < N; i++) fac[i] = fac[i - 1] * i % mod;
24     inv[N - 1] = binpow(fac[N - 1], mod - 2);
25     for (int i = N - 2; i >= 1; i--)
26     {
27         inv[i] = inv[i + 1] * (i + 1) % mod;
28     }
29 }
30
31 auto C = [&](int x, int y) -> int
32 {
33     return (fac[x] * inv[y] % mod) * inv[x - y] % mod;
34 };

```

5.1.2. Euler_sieve.h

```

1  #ifndef _IN_TEMPLATE_
2  #include <template_overAll.h>
3  #endif
4
5  vector<int> init(int n)
6  {
7      vector<int> pri;
8      vector<bool> vis(n, 0);
9      for (int i = 2; i <= n; i++)
10     {
11         if (!vis[i])
12             pri.push_back(i);
13         for (int j = 0; j < pri.size(); j++)
14         {
15             if (i * pri[j] > n)
16                 break;
17             vis[pri[j] * i] = 1;
18             if (i % pri[j] == 0)
19                 break;
20         }
21     }
22     return pri;
23 }
```

5.1.3. factor_pr.h

```

1  #include <template_overAll.h>
2  #define int long long
3  #define pii pair<int, int>
4  const int INF = 1145141919810LL;
5  using namespace std;
6
7  class Pollard_Rho
8  {
9  private:
10
11     vector<int> B;
12
13     int mul(int a, int b, int m)
14     {
15         int r = a * b - m * (int)(1.L / m * a * b);
16         return r - m * (r >= m) + m * (r < 0);
17     }
18
19     int mypow(int a, int b, int m)
20     {
21         int res = 1 % m;
22         for (; b; b >>= 1, a = mul(a, a, m))
23         {
24             if (b & 1)
25             {
26                 res = mul(res, a, m);
27             }
28         }
29         return res;
30     }
31
32     bool MR(int n)
33     {
34         if (n <= 1)
35             return 0;
36         for (int p : B)
37         {
38             if (n == p)
39                 return 1;
40             if (n % p == 0)
41                 return 0;
42         }
43         int m = (n - 1) >> __builtin_ctz(n - 1);
44         for (int p : B)
45         {
46             int t = m, a = mypow(p, m, n);
47             while (t != n - 1 && a != 1 && a != n - 1)
48             {
49                 a = mul(a, a, n);
50                 t *= 2;
51             }
52             if (a != n - 1 && t % 2 == 0)
53                 return 0;
54         }
55         return 1;
56     }
57 }

```

```

58     inline const int getfecsum(int _n)
59     {
60         int sum = 0;
61         while (_n)
62         {
63             sum += _n % 10;
64             _n /= 10;
65         }
66         return sum;
67     };
68
69     int PR(int n)
70     {
71         for (int p : B)
72         {
73             if (n % p == 0)
74                 return p;
75         }
76         auto f = [&](int x) -> int
77         {
78             x = mul(x, x, n) + 1;
79             return x >= n ? x - n : x;
80         };
81         int x = 0, y = 0, tot = 0, p = 1, q, g;
82         for (int i = 0; (i & 255) || (g = gcd(p, n)) == 1; i++, x = f(x), y =
f(f(y)))
83         {
84             if (x == y)
85             {
86                 x = tot++;
87                 y = f(x);
88             }
89             q = mul(p, abs(x - y), n);
90             if (q)
91                 p = q;
92         }
93         return g;
94     }
95
96     vector<int> fac(int n)
97     {
98         // if(n == 0)
99         // #define pb emplace_back
100         if (n <= 1)
101             return {};
102         if (MR(n))
103             return {n};
104         int d = PR(n);
105         auto v1 = fac(d), v2 = fac(n / d);
106         auto i1 = v1.begin(), i2 = v2.begin();
107         vector<int> ans;
108         while (i1 != v1.end() || i2 != v2.end())
109         {
110             if (i1 == v1.end())
111             {
112                 ans.pb(*i2++);
113             }
114             else if (i2 == v2.end())

```

```

115         {
116             ans.pb(*i1++);
117         }
118         else
119         {
120             if (*i1 < *i2)
121             {
122                 ans.pb(*i1++);
123             }
124             else
125             {
126                 ans.pb(*i2++);
127             }
128         }
129     }
130     return ans;
131 }
132
133 public:
134
135 Pollard_Rho(){
136     B = {2, 3, 5, 7, 11, 13, 17, 19, 23};
137 }
138
139 vector<pii> fac_Comp(int n)
140 {
141     auto srt = fac(n);
142     map<int, int> cnt;
143     for (auto x : srt)
144         cnt[x]++;
145     vector<pii> rt;
146     for (auto x : cnt)
147         rt.push_back(x);
148     return rt;
149 }
150
151 vector<int> fac_pri(int n)
152 {
153     return fac(n);
154 }
155
156 vector<int> fac_all(int n)
157 {
158     vector<pii> rt = fac_Comp(n);
159     vector<int> v;
160     function<void(int, int)> dfs = [&](int id, int x)
161     {
162         if (id == rt.size())
163         {
164             v.push_back(x);
165             return;
166         }
167         for(int i = 0; i <= rt[id].se; i++)
168         {
169             dfs(id + 1, x * (mypow(rt[id].fi, i, INF)));
170         }
171     };
172     dfs(0, 1);

```



```
173         return v;  
174     }  
175 };
```

5.2. other

5.2.1. Frac.h

```

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

```

```

55     friend bool operator==(const Frac &lhs, const Frac &rhs) {
56         return lhs.num * rhs.den == rhs.num * lhs.den;
57     }
58     friend bool operator!=(const Frac &lhs, const Frac &rhs) {
59         return lhs.num * rhs.den != rhs.num * lhs.den;
60     }
61     friend bool operator<(const Frac &lhs, const Frac &rhs) {
62         return lhs.num * rhs.den < rhs.num * lhs.den;
63     }
64     friend bool operator>(const Frac &lhs, const Frac &rhs) {
65         return lhs.num * rhs.den > rhs.num * lhs.den;
66     }
67     friend bool operator<=(const Frac &lhs, const Frac &rhs) {
68         return lhs.num * rhs.den <= rhs.num * lhs.den;
69     }
70     friend bool operator>=(const Frac &lhs, const Frac &rhs) {
71         return lhs.num * rhs.den >= rhs.num * lhs.den;
72     }
73     friend std::ostream &operator<<(std::ostream &os, Frac x) {
74         T g = std::gcd(x.num, x.den);
75         if (x.den == g) {
76             return os << x.num / g;
77         } else {
78             return os << x.num / g << "/" << x.den / g;
79         }
80     }
81 };
82
83 using F = Frac<int>;
    
```

6. string

6.1. compress_print.h

```

1  const int N = 1 << 21;
2  static const int mod1 = 1E9 + 7, base1 = 127;
3  static const int mod2 = 1E9 + 9, base2 = 131;
4  vector<int> val1;
5  vector<int> val2;
6  void init(int n = N)
7  {
8      val1.resize(n + 1), val2.resize(n + 2);
9      val1[0] = 1, val2[0] = 1;
10     for (int i = 1; i <= n; i++)
11     {
12         val1[i] = val1[i - 1] * base1 % mod1;
13         val2[i] = val2[i - 1] * base2 % mod2;
14     }
15 }
16
17 string compress(vector<string> in)
18 { // 前后缀压缩
19     vector<int> h1{1};
20     vector<int> h2{1};
21     string ans = "#";
22     for (auto s : in)
23     {
24         s = "#" + s;
25         int st = 0;
26         int chk1 = 0;
27         int chk2 = 0;
28         for (int j = 1; j < s.size() && j < ans.size(); j++)
29         {
30             chk1 = (chk1 * base1 % mod1 + s[j]) % mod1;
31             chk2 = (chk2 * base2 % mod2 + s[j]) % mod2;
32             if ((h1.back() == (h1[ans.size() - 1 - j] * val1[j] % mod1 +
33 % mod1) &&
34                 (h2.back() == (h2[ans.size() - 1 - j] * val2[j] % mod2 +
35 % mod2)    )
36                 st = j;
37         }
38         for (int j = st + 1; j < s.size(); j++)
39         {
40             ans += s[j];
41             h1.push_back((h1.back() * base1 % mod1 + s[j]) % mod1);
42             h2.push_back((h2.back() * base2 % mod2 + s[j]) % mod2);
43         }
44     }
45     return ans.substr(1);
46 }

```

6.2. get_occrr.h

```

1  #include <template_overAll.h>
2
3  /*
4   * 找到某一堆短字符串在长字符串中的出现位置
5   *  dira=1 为最早出现的后端点下标  dira=0 为最晚出现的前端点下标
6   *  源字符串 s 长度为|s|, 查找字符串列表中所有字符串长度和为|_s|
7   *  则时间复杂度为 O(max(|_s|log(|_s|),|s|))
8   */
9  class get_occrr
10 {
11 private:
12     string s;
13 public:
14     get_occrr(string _s) { s = _s; }
15     vector<int> locate(vector<string> _s, bool dira = 1)
16     {
17         int n = _s.size();
18         vector<int> occr(n, -1);
19         map<char, vector<pair<int, int>>> gncing;
20         if(dira == 1)
21         {
22             for(int i = 0; i < n; i++)
23                 gncing[_s[i][0]].push_back({i, 0});
24             for(int i = 0; i < s.size(); i++)
25             {
26                 vector<pair<int, int>> gnctmp = gncing[s[i]];
27                 gncing[s[i]].clear();
28                 for(int j = 0; j < gnctmp.size(); j++)
29                 {
30                     if(gnctmp[j].se+1 < _s[gnctmp[j].fi].size())
31                         gncing[_s[gnctmp[j].fi]]
32 [gnctmp[j].se+1].push_back({gnctmp[j].fi, gnctmp[j].se+1});
33                     else occr[gnctmp[j].fi] = i;
34                 }
35             } else {
36                 for(int i = 0; i < n; i++) gncing[_s[i]
37 [_s[i].size()-1]].push_back({i, _s[i].size()-1});
38                 for(int i = s.size()-1; i >= 0; i--)
39                 {
40                     vector<pair<int, int>> gnctmp = gncing[s[i]];
41                     gncing[s[i]].clear();
42                     for(int j = 0; j < gnctmp.size(); j++)
43                     {
44                         if(gnctmp[j].se - 1 >= 0)
45                             gncing[_s[gnctmp[j].fi]]
46 [gnctmp[j].se-1].push_back({gnctmp[j].fi, gnctmp[j].se-1});
47                         else occr[gnctmp[j].fi] = i;
48                     }
49                 }
50             }
51         }
52     }
53     return occr;
54 }
55 };

```

6.3. hash_print.h

```

1  #define int long long
2  const int N = 1 << 21;
3  static const int mod1 = 1E9 + 7, base1 = 127;
4  static const int mod2 = 1E9 + 9, base2 = 131;
5  vector<int> val1;
6  vector<int> val2;
7  using puv = pair<int,int>;
8  void init(int n = N)
9  {
10     val1.resize(n + 1), val2.resize(n + 2);
11     val1[0] = 1, val2[0] = 1;
12     for (int i = 1; i <= n; i++)
13     {
14         val1[i] = val1[i - 1] * base1 % mod1;
15         val2[i] = val2[i - 1] * base2 % mod2;
16     }
17 }
18 class hstring
19 {
20 public:
21     vector<int> h1;
22     vector<int> h2;
23     string s;
24
25     hstring(string s_) : s(s_), h1{1}, h2{1}
26     {
27         build();
28     }
29
30     void build()
31     {
32         for (auto it : s)
33         {
34             h1.push_back((h1.back() * base1 % mod1 + it) % mod1);
35             h2.push_back((h2.back() * base2 % mod2 + it) % mod2);
36         }
37     }
38
39     puv get()
40     { // 输出整串的哈希值
41         return {h1.back(), h2.back()};
42     }
43
44     puv substring(int l, int r)
45     { // 输出子串的哈希值
46         if (l > r) swap(l, r);
47         int ans1 = (mod + h1[r + 1] - h1[l] * val1[r - l + 1] % mod1) % mod1;
48         int ans2 = (mod + h2[r + 1] - h2[l] * val2[r - l + 1] % mod2) % mod2;
49         return {ans1, ans2};
50     }
51
52     puv modify(int idx, char x)
53     { // 修改 idx 位为 x
54         int n = s.size() - 1;
55         int ans1 = (h1.back() + val1[n - idx] * (x - s[idx]) % mod1) % mod1;
56         int ans2 = (h2.back() + val2[n - idx] * (x - s[idx]) % mod2) % mod2;
57         return {ans1, ans2};

```

```
58     }  
59 };
```

6.4. KMP.h

```

1  #include <template_overAll.h>
2
3  class KMP
4  {
5  private:
6      string s;
7      string inis;
8  public:
9      vector<int> pi;
10     KMP(string _s)
11     {
12         s = _s;
13         inis = s;
14     }
15     void prefix_function()
16     {
17         pi.clear();
18         int n = (int)s.length();
19         pi.resize(n);
20         for (int i = 1; i < n; i++)
21         {
22             int j = pi[i - 1];
23             while (j > 0 && s[i] != s[j])
24                 j = pi[j - 1];
25             if (s[i] == s[j])
26                 j++;
27             pi[i] = j;
28         }
29         return;
30     }
31     vector<int> find_occr(string p)
32     {
33         s = inis;
34         s = p + "#" + s;
35         prefix_function();
36         vector<int> v;
37         for (int i = p.size() + 1; i < s.size(); i++)
38             if (pi[i] == p.size())
39                 v.push_back(i - 2 * p.size());
40         return v;
41     }
42 };

```


6.5. trie_Tree.h

```

1  #include <template_overAll.h>
2
3  class Trie//AC
4  {
5  public:
6      vector<map<char, int>> t;
7      int root = 0;
8      Trie()
9      {
10         t.resize(1);
11     }
12     void addedge(string _s)
13     {
14         int pvidx = root;
15         _s.push_back('-');
16         for (int i = 0; i < _s.size(); i++)
17         {
18             if (t[pvidx].find(_s[i]) != t[pvidx].end())
19             {
20                 pvidx = t[pvidx][_s[i]];
21             }
22             else
23             {
24                 t[pvidx][_s[i]] = t.size();
25                 t.push_back(map<char, int>());
26                 pvidx = t[pvidx][_s[i]];
27             }
28         }
29     }
30     bool ifcmp(string &s)
31     {
32         int pvidx = root;
33         for(int i = 0;i < s.size();i ++)
34         {
35             if(t[pvidx].find(s[i]) != t[pvidx].end()) pvidx = t[pvidx][s[i]];
36             else return 0;
37         }
38         return t[pvidx].find('-') != t[pvidx].end();
39     }
40 };

```

