

BELJING NORMAL UNIVERSITY
SCHOOL OF MATHEMATICS

Template

appleDog

2024 年 11 月 23 日

目录

| | | |
|----------|-----------------------------------|-----------|
| 1 | hpp | 3 |
| 1.1 | heading | 3 |
| 1.2 | debug.h | 4 |
| 2 | shell scripts | 5 |
| 2.1 | linux version | 5 |
| 2.2 | windows version | 5 |
| 3 | data structure | 6 |
| 3.1 | stack | 6 |
| 3.2 | queue | 6 |
| 3.3 | DSU | 6 |
| 3.4 | spare table | 6 |
| 3.5 | Cartesian tree | 7 |
| 3.6 | segment tree | 7 |
| 3.7 | segment tree split | 9 |
| 3.8 | persistent segment tree | 10 |
| 3.9 | sweep line | 12 |
| 3.10 | treap | 13 |
| 3.11 | splay | 17 |
| 3.12 | link cut tree | 20 |
| 3.13 | Lichao tree | 20 |
| 3.14 | ODT | 21 |
| 4 | string | 21 |
| 4.1 | kmp | 21 |
| 4.2 | z function | 22 |
| 4.3 | manacher | 22 |
| 4.4 | AC automaton | 22 |
| 4.5 | PAM | 23 |
| 4.6 | Suffix Array | 24 |
| 4.7 | Cantor expansion | 25 |
| 4.8 | trie | 26 |
| 5 | math - number theory | 28 |

| | |
|--------------------------------|-----------|
| 目录 | 3 |
| 5.1 mod int | 28 |
| 5.2 Eculid | 29 |
| 5.3 inverse | 30 |
| 5.4 sieve | 31 |
| 5.5 powerful number | 33 |
| 5.6 block | 34 |
| 5.7 CRT & exCRT | 35 |
| 5.8 BSGS & exBSGS | 36 |
| 5.9 Miller Rabin | 36 |
| 5.10 Pollard Rho | 37 |
| 5.11 quadratic residu | 37 |
| 5.12 Lucas | 38 |
| 5.13 Wilson | 40 |
| 5.14 LTE | 40 |
| 5.15 Mobius inversion | 41 |
| 6 math - polynomial | 43 |
| 6.1 FTT | 43 |
| 6.2 FWT | 44 |
| 6.3 class polynomial | 46 |
| 6.4 wsy poly | 50 |
| 6.5 Lagrange interpolation | 55 |
| 7 math - game theory | 57 |
| 7.1 nim game | 57 |
| 7.2 anti - nim game | 57 |
| 8 math - linear algebra | 58 |
| 8.1 matrix | 58 |
| 8.2 linear basis | 59 |
| 8.3 linear programming | 60 |
| 8.4 bm | 60 |
| 9 complex number | 62 |
| 10 graph | 63 |
| 10.1 topology sort | 63 |
| 10.2 shortest path | 63 |

| | | |
|-----------|--|-----------|
| 10.3 | minimum spanning tree | 66 |
| 10.4 | SCC | 66 |
| 10.5 | DCC | 67 |
| 10.6 | 2-sat | 70 |
| 10.7 | minimum ring | 71 |
| 10.8 | tree - center of gravity | 71 |
| 10.9 | tree - DSU on tree | 71 |
| 10.10 | tree - AHU | 72 |
| 10.11 | tree - LCA | 73 |
| 10.12 | tree - heavy light decomposition | 73 |
| 10.13 | tree - virtual tree | 74 |
| 10.14 | tree - pseudo tree | 75 |
| 10.15 | tree - divide and conquer on tree | 76 |
| 10.16 | tree - matrix tree | 78 |
| 10.17 | Prefür sequence | 79 |
| 10.18 | network flow - maximal flow | 79 |
| 10.19 | network flow - minimum cost flow | 82 |
| 10.20 | network flow - minimal cut | 84 |
| 10.21 | network flow - upper / lower bound | 85 |
| 10.22 | network flow - other versions | 86 |
| 10.23 | matching - matching on bipartite graph | 88 |
| 10.24 | matching - matching on general graph | 90 |
| 11 | geometry | 91 |
| 11.1 | two demention | 91 |
| 11.2 | convex | 91 |
| 11.3 | half plane union | 93 |
| 11.4 | rotate | 93 |
| 11.5 | Simpson | 94 |
| 12 | offline algorithm | 95 |
| 12.1 | discretization | 95 |
| 12.2 | Mo algorithm | 95 |
| 12.3 | 回滚莫队 | 95 |
| 12.4 | CDQ | 96 |
| 12.5 | segment tree devide and conquer | 97 |

| | | |
|-----------|--|------------|
| 13 | Print All Cases | 99 |
| 13.1 | print all trees with n nodes | 99 |
| 13.1.1 | 有根树 | 99 |
| 14 | Magic | 101 |
| 14.1 | magic heap | 101 |
| 14.2 | operator queue | 101 |
| 14.3 | Fast GCD | 102 |
| 14.4 | $q \equiv \frac{a}{b} \bmod mod$ | 102 |

1 hpp

1.1 heading

```

1  #include <bits/stdc++.h>
2
3  // using namespace std;
4
5  using LL = long long;
6  using i128 = __int128;
7  using PII = std::pair<int, int>;
8  /*
9  using UI = unsigned int;
10 using ULL = unsigned long long;
11 using ULL = unsigned long long;
12 using PII = std::pair<int, LL>;
13 using PLI = std::pair<LL, int>;
14 using PLL = std::pair<LL, LL>;
15 using vi = std::vector<int>;
16 using vvi = std::vector<vi>;
17 using vl = std::vector<LL>;
18 using vvl = std::vector<vl>;
19 using vpi = std::vector<PII>;
20 */
21
22 #define ff first
23 #define ss second
24 #define all(v) v.begin(), v.end()
25 #define rall(v) v.rbegin(), v.rend()
26
27 #ifdef LOCAL
28 #include "debug.h"
29 #else
30 #define debug(...) \
31     do { \
32     } while (false)
33 #endif
34
35 constexpr int inf = 0x3f3f3f3f;
36 constexpr LL INF = 1e18;
37 constexpr int lowbit(int x) { return x & -x; }
38 /*
39 constexpr int add(int x, int y) { return x + y < mod ? x + y : x - mod + y; }
40 constexpr int sub(int x, int y) { return x < y ? mod + x - y : x - y; }
41 constexpr int mul(LL x, int y) { return x * y % mod; }
42 constexpr void Add(int& x, int y) { x = add(x, y); }
43 constexpr void Sub(int& x, int y) { x = sub(x, y); }
44 constexpr void Mul(int& x, int y) { x = mul(x, y); }
45 constexpr int pow(int x, int y, int z = 1) {
46     for (; y; y /= 2) {
47         if (y & 1) Mul(z, x);
48         Mul(x, x);
49     }
50     return z;
51 }
52 temps constexpr int add(Ts... x) {
53     int y = 0;
54     (... , Add(y, x));
55     return y;
56 }
57 temps constexpr int mul(Ts... x) {
58     int y = 1;
59     (... , Mul(y, x));
60     return y;
61 }
62 */
63 tandu bool Max(T& x, const U& y) { return x < y ? x = y, true : false; }
64 tandu bool Min(T& x, const U& y) { return x > y ? x = y, true : false; }
65
66 void solut() {
67     ;
68 }
69
70 int main() {
71     std::ios::sync_with_stdio(false);
72     std::cin.tie(0);
73     int t = 1;
74     std::cin >> t;
75     while (t--) {
76         solut();
77     }
78     return 0;
79 }

```

1.2 debug.h

```

1  template <typename T, typename U>
2  std::ostream& operator<<(std::ostream& os, const std::pair<T, U>& p) {
3      return os << '<' << p.first << ',' << p.second << '>';
4  }
5
6  template <
7      typename T, typename = decltype(std::begin(std::declval<T>())),
8      typename = std::enable_if_t<!std::is_same_v<T, std::string>>>
9  std::ostream& operator<<(std::ostream& os, const T& c) {
10     auto it = std::begin(c);
11     if (it == std::end(c)) return os << "{}";
12     for (os << '{' << *it; ++it != std::end(c); os << ',' << *it);
13     return os << '}';
14 }
15
16 #define debug(arg...) \
17     do { \
18         std::cerr << "[" #arg "]" :"; \
19         dbg(arg); \
20     } while (false)
21
22 template <typename... Ts>
23 void dbg(Ts... args) {
24     (... , (std::cerr << ' ' << args));
25     std::cerr << std::endl;
26 }

```

2 shell scripts

2.1 linux version

```
1  #!/bin/bash
2
3  cd "$1"
4
5  g++ -o main -O2 -std=c++17 -DLOCAL main.cpp -ftrapv -fsanitize=address,undefined
6
7  for input in *.in; do
8      output=${input%.*}.out
9      answer=${input%.*}.ans
10
11      ./main < $input > $output
12
13      echo "case ${input%.*}: "
14      echo "My: "
15      cat $output
16      echo "Answer: "
17      cat $answer
18
19  done
```

2.2 windows version

```
1  @echo off
2
3  cd %1
4
5  del .\main.exe
6
7  g++ -o main.exe main.cpp -DLOCAL -std=c++17 -ftrapv
8
9  for %%i in (*.in) do (
10     main.exe < %%i > %%~ni.out
11     echo case %%~ni:
12     echo My:
13     type %%~ni.out
14     echo Answer:
15     type %%~ni.ans
16 )
17
18 cd ../shell
```


3 data structure

3.1 stack

```

1 vi stk;
2 for (int i = 1; i <= n; i++){
3     while (!stk.empty() and stk.back() > a[i]) {
4         stk.pop_back();
5     }
6     stk.push_back(a[i]);
7 }

```

3.2 queue

```

1 std::deque<int> q;
2 for (int i = 1; i <= n; i++) {
3     while (!q.empty() and a[q.back()] >= a[i]) q.pop_back();
4     if (!q.empty() and i - q.front() >= k) q.pop_front();
5     q.push_back(i);
6 }

```

3.3 DSU

```

1 /* DSU */
2 vi fa(n + 1);
3 std::iota(all(fa), 0);
4 std::function<int(int)> find = [&] (int x) -> int{
5     return x == fa[x] ? x : fa[x] = find(fa[x]);
6 };
7 auto merge = [&] (int x, int y) -> void{
8     x = find(x), y = find(y);
9     if (x == y) return;
10    // operations //
11    fa[y] = x;
12 };

```

3.4 spare table

一维

```

1 /* spare table */
2 int B = 30;
3 vvi f(n + 1, vi(B));
4 vi Log2(n + 1);
5 auto init = [&]() -> void {
6     for (int i = 1; i <= n; i++) {
7         f[i][0] = a[i];
8         if (i > 1) Log2[i] = Log2[i / 2] + 1;
9     }
10    int t = Log2[n];
11    for (int j = 1; j <= t; j++) {
12        for (int i = 1; i <= n - (1 << j) + 1; i++) {
13            f[i][j] = std::max(f[i][j - 1], f[i + (1 << (j - 1))][j - 1]);
14        }
15    }
16 };
17 init();
18 auto query = [&](int l, int r) -> int {
19     int t = Log2[r - l + 1];
20     return std::max(f[l][t], f[r - (1 << t) + 1][t]);
21 };

```

二维

```

1  /* spare table */
2  intB = 30;
3  std::vector f(n + 1, std::vector<std::array<std::array<int, B>, B>>(m + 1));
4  vi Log2(n + 1);
5  auto init = [&]() -> void {
6      for (int i = 2; i <= std::max(n, m); i++) {
7          Log2[i] = Log2[i / 2] + 1;
8      }
9      for (int i = 2; i <= n; i++) {
10         for (int j = 2; j <= m; j++) {
11             f[i][j][0][0] = a[i][j];
12         }
13     }
14     for (int ki = 0; ki <= Log2[n]; ki++) {
15         for (int kj = 0; kj <= Log2[n]; kj++) {
16             if (!ki && !kj) continue;
17             for (int i = 1; i <= n - (1 << ki) + 1; i++) {
18                 for (int j = 1; j <= m - (1 << kj) + 1; j++) {
19                     if (ki) {
20                         f[i][j][ki][kj] =
21                             std::max(f[i][j][ki - 1][kj], f[i + (1 << (ki - 1))][j][ki - 1][kj]);
22                     } else {
23                         f[i][j][ki][kj] =
24                             std::max(f[i][j][ki][kj - 1], f[i][j + (1 << (kj - 1))][ki][kj - 1]);
25                     }
26                 }
27             }
28         }
29     }
30 };
31 init();
32 auto query = [&](int x1, int y1, int x2, int y2) -> int {
33     int ki = Log2[x2 - x1 + 1], kj = Log2[y2 - y1 + 1];
34     int t1 = f[x1][y1][ki][kj];
35     int t2 = f[x2 - (1 << ki) + 1][y1][ki][kj];
36     int t3 = f[x1][y2 - (1 << kj) + 1][ki][kj];
37     int t4 = f[x2 - (1 << ki) + 1][y2 - (1 << kj) + 1][ki][kj];
38     return std::max({t1, t2, t3, t4});
39 };

```

3.5 Cartesian tree

一种特殊的平衡树, 用元素的值作为平衡点节点的 *val*, 元素的下标作为 *key*.

```

1  /* cartesian tree */
2  vi ls(n + 1), rs(n + 1), stk(n + 1);
3  int top = 1;
4  for (int i = 1; i <= n; i++) {
5      int k = top;
6      while (k and a[stk[k]] > a[i]) k--;
7      if (k) rs[stk[k]] = i;
8      if (k < top) ls[i] = stk[k + 1];
9      stk[++k] = i;
10     top = k;
11 }

```

3.6 segment tree

```

1  /* segment tree @ czr */
2  const int N = 100010;
3  struct node {
4      int l, r;
5      ll sum, maxn, add, set;
6      bool addflag, setflag;
7  } tr[N << 2];
8
9  void push_up(int u) {
10     tr[u].sum = tr[u << 1].sum + tr[u << 1 | 1].sum;
11     tr[u].maxn = max(tr[u << 1].maxn, tr[u << 1 | 1].maxn);
12 }
13 // 0 4 0 0 0
14 // 2 6 2 2 2
15 // 2 6 2 4 4
16
17 void push_down(int u) {
18     auto& root = tr[u], &left = tr[u << 1], &right = tr[u << 1 | 1];
19     if (root.setflag) {

```

```

20     assert(!root.addflag);
21     left.add = 0, left.set = root.set, left.addflag = false, left.setflag = true;
22     right.add = 0, right.set = root.set, right.addflag = false, right.setflag = true;
23     left.sum = root.set * (left.r - left.l + 1);
24     right.sum = root.set * (right.r - right.l + 1);
25     left.maxn = root.set;
26     right.maxn = root.set;
27     root.set = 0, root.setflag = false;
28 }
29 if (root.addflag) {
30     assert(!root.setflag);
31     if (left.setflag) left.set += root.add;
32     else left.add += root.add, left.addflag = true;
33     if (right.setflag) right.set += root.add;
34     else right.add += root.add, right.addflag = true;
35
36     left.sum += root.add * (left.r - left.l + 1);
37     right.sum += root.add * (right.r - right.l + 1);
38     left.maxn += root.add;
39     right.maxn += root.add;
40     root.add = 0, root.addflag = false;
41 }
42 assert(root.add == 0);
43 }
44
45 void build(int u, int l, int r, vector<ll>& a) {
46     if (l == r) {
47         tr[u].l = tr[u].r = l, tr[u].sum = tr[u].maxn = a[l];
48         tr[u].add = 0, tr[u].set = 0;
49         tr[u].addflag = tr[u].setflag = false;
50     } else {
51         tr[u].l = l, tr[u].r = r, tr[u].add = 0, tr[u].set = 0;
52         tr[u].addflag = tr[u].setflag = false;
53         int mid = l + r >> 1;
54         build(u << 1, l, mid, a);
55         build(u << 1 | 1, mid + 1, r, a);
56         push_up(u);
57     }
58 }
59
60 // 区间加
61 void modify(int u, int l, int r, ll d) {
62     if (l > r) return;
63     if (tr[u].l >= l && tr[u].r <= r) {
64         if (tr[u].setflag) tr[u].set += d;
65         else tr[u].add += d, tr[u].addflag = true;
66         tr[u].sum += d * (tr[u].r - tr[u].l + 1);
67         tr[u].maxn += d;
68     } else {
69         push_down(u);
70         int mid = tr[u].l + tr[u].r >> 1;
71         if (l <= mid) modify(u << 1, l, r, d);
72         if (r > mid) modify(u << 1 | 1, l, r, d);
73         push_up(u);
74     }
75 }
76
77 // 区间赋值
78 void update(int u, int l, int r, ll x) {
79     if (l > r) return;
80     if (tr[u].l >= l && tr[u].r <= r) {
81         tr[u].set = x, tr[u].setflag = true;
82         tr[u].add = 0, tr[u].addflag = false;
83         tr[u].sum = x * (tr[u].r - tr[u].l + 1);
84         tr[u].maxn = x;
85     } else {
86         push_down(u);
87         int mid = tr[u].l + tr[u].r >> 1;
88         if (l <= mid) update(u << 1, l, r, x);
89         if (r > mid) update(u << 1 | 1, l, r, x);
90         push_up(u);
91     }
92 }
93
94 ll query_sum(int u, int l, int r) {
95     if (l > r) return 0;
96     if (tr[u].l >= l && tr[u].r <= r) return tr[u].sum;
97     else {
98         ll res = 0;
99         push_down(u);
100         int mid = tr[u].l + tr[u].r >> 1;
101         if (l <= mid) res += query_sum(u << 1, l, r);
102         if (r > mid) res += query_sum(u << 1 | 1, l, r);
103         return res;
104     }
105 }
106

```

```

107 ll query_maxn(int u, int l, int r) {
108     if (l > r) return -1e18;
109     if (tr[u].l >= l && tr[u].r <= r) return tr[u].maxn;
110     else {
111         ll res = -1e18;
112         push_down(u);
113         int mid = tr[u].l + tr[u].r >> 1;
114         if (l <= mid) res = max(res, query_maxn(u << 1, l, r));
115         if (r > mid) res = max(res, query_maxn(u << 1 | 1, l, r));
116         return res;
117     }
118 }
119
120 // 找到最小 i 使得 sum(l, i) >= k
121 ll find_presum_idx(int u, int l, int r, int x) {
122     if (tr[u].l == tr[u].r) return tr[u].l;
123     else {
124         push_down(u);
125         int mid = tr[u].l + tr[u].r >> 1;
126         if (r <= mid) {
127             return find_presum_idx(u << 1, l, r, x);
128         } else if (l > mid) {
129             return find_presum_idx(u << 1 | 1, l, r, x);
130         } else {
131             ll lsum = query_sum(u << 1, l, r);
132             if (lsum >= x) return find_presum_idx(u << 1, l, mid, x);
133             else return find_presum_idx(u << 1 | 1, mid + 1, r, x - lsum);
134         }
135     }
136 }

```

3.7 segment tree split

```

1  /* segment tree split @ wrb */
2  #include<bits/stdc++.h>
3  using namespace std;
4  namespace Acc{
5      using i64=int64_t;
6      enum{N=200009,M=10000000};
7      i64 v[M];
8      int lc[M],rc[M],tot,a[N],r[N];
9      auto up=[](int o){
10         v[o]=v[lc[o]]+v[rc[o]];
11     };
12     void bd(int&o,int l,int r){
13         if(o==++tot,l==r)return cin>>v[o],void();
14         int md=l+r>>1;
15         bd(lc[o],l,md),bd(rc[o],md+1,r),up(o);
16     }
17     void spl(int&o,int&x,int l,int r,int L,int R){
18         if(l<=L&&R<=r)return o=x,x=0,void();
19         int md=L+R>>1;
20         o=++tot;
21         if(l<=md)spl(lc[o],lc[x],l,r,L,md);
22         if(r>md)spl(rc[o],rc[x],l,r,md+1,R);
23         up(o),up(x);
24     }
25     void mg(int&o,int x,int l,int r){
26         if(!o||!x)return o|=x,void();
27         if(l==r)return v[o]+=v[x],void();
28         int md=l+r>>1;
29         mg(lc[o],lc[x],l,md);
30         mg(rc[o],rc[x],md+1,r);
31         up(o);
32     }
33     void ins(int&o,int l,int r,int x,int k){
34         if(!o)o=++tot;
35         if(v[o]==k,l==r)return;
36         int md=l+r>>1;
37         x<=md?ins(lc[o],l,md,x,k):ins(rc[o],md+1,r,x,k);
38     }
39     i64 qry(int o,int l,int r,int L,int R){
40         if(!o)return 0;
41         if(l<=L&&R<=r)return v[o];
42         int md=L+R>>1;i64 z=0;
43         if(l<=md)z=qry(lc[o],l,r,L,md);
44         if(r>md)z+=qry(rc[o],l,r,md+1,R);
45         return z;
46     }
47     int kth(int o,int l,int r,int k){
48         if(l==r)return l;
49         if(k>v[o])return -1;

```

```

50     int md=l+r>>1;
51     if(k<=v[lc[o]])return kth(lc[o],1,md,k);
52     else return kth(rc[o],md+1,r,k-v[lc[o]]);
53 }
54 auto work=[]() {
55     int n,m,i,x,y,o=1;
56     for(cin>>n>>m,bd(r[1],1,n);m--;)switch(cin>>i,i){
57         case 0:cin>>i>>x>>y,spl(r[++o],r[i],x,y,1,n);break;
58         case 1:cin>>x>>y,mg(r[x],r[y],1,n);break;
59         case 2:cin>>i>>x>>y,ins(r[i],1,n,y,x);break;
60         case 3:cin>>i>>x>>y,cout<<qry(r[i],x,y,1,n)<<'\\n';break;
61         case 4:cin>>i>>x,cout<<kth(r[i],1,n,x)<<'\\n';break;
62     }
63 };
64 }
65 int main(){
66     ios::sync_with_stdio(0);
67     cin.tie(0),Acc::work();
68 }

```

3.8 persistent segment tree

单点修改，版本拷贝

n 个数， m 次操作，操作分别为

1. v_i 1 loc_i $value_i$: 将第 v_i 个版本的 $a[loc_i]$ 修改为 $value_i$,
2. v_i 2 loc_i : 拷贝第 v_i 个版本，并查询该版本的 $a[loc_i]$.

```

1 // 洛谷 P3919 【模板】可持久化线段树 1 (可持久化数组)
2
3 struct node {
4     int l, r, key;
5 };
6
7 int main() {
8     std::ios::sync_with_stdio(false);
9     std::cin.tie(0);
10    std::cout.tie(0);
11
12    int n, m;
13    std::cin >> n >> m;
14    vi a(n + 1);
15    for (int i = 1; i <= n; i++) {
16        std::cin >> a[i];
17    }
18
19    /* hjt segment tree */
20    int idx = 0;
21    vi root(m + 1);
22    std::vector<node> tr(n * 25);
23
24    std::function<int(int, int)> build = [&](int l, int r) -> int {
25        int p = ++idx;
26        if (l == r) {
27            tr[p].key = a[l];
28            return p;
29        }
30        int mid = (l + r) >> 1;
31        tr[p].l = build(l, mid);
32        tr[p].r = build(mid + 1, r);
33        return p;
34    };
35
36    std::function<int(int, int, int, int, int)> modify = [&](int p, int l, int r, int k,
37                                                            int x) -> int {
38        int q = ++idx;
39        tr[q].l = tr[p].l, tr[q].r = tr[p].r;
40        if (tr[q].l == tr[q].r) {
41            tr[q].key = x;
42            return q;
43        }
44        int mid = (l + r) >> 1;
45        if (k <= mid) {
46            tr[q].l = modify(tr[q].l, l, mid, k, x);
47        } else {

```

```

48         tr[q].r = modify(tr[q].r, mid + 1, r, k, x);
49     }
50     return q;
51 };
52
53 std::function<int(int, int, int, int)> query = [&](int p, int l, int r, int k) -> int {
54     if (tr[p].l == tr[p].r) {
55         return tr[p].key;
56     }
57     int mid = (l + r) >> 1;
58     if (k <= mid) {
59         return query(tr[p].l, l, mid, k);
60     } else {
61         return query(tr[p].r, mid + 1, r, k);
62     }
63 };
64
65 root[0] = build(1, n);
66
67 for (int i = 1; i <= m; i++) {
68     int op, ver, k, x;
69     std::cin >> ver >> op;
70     if (op == 1) {
71         std::cin >> k >> x;
72         root[i] = modify(root[ver], 1, n, k, x);
73     } else {
74         std::cin >> k;
75         root[i] = root[ver];
76         std::cout << query(root[ver], 1, n, k) << '\n';
77     }
78 }
79
80 return 0;
81 }

```

区间第 k 小

长度为 n 的序列 a , m 次查询, 每次查询 $[l, r]$ 中的第 k 小值.

```

1 // 洛谷P3834 【模板】可持久化线段树 2
2
3 struct node {
4     int l, r, cnt;
5 };
6
7 int main() {
8     std::ios::sync_with_stdio(false);
9     std::cin.tie(0);
10    std::cout.tie(0);
11
12    int n, m;
13    std::cin >> n >> m;
14    vi a(n + 1), v;
15    for (int i = 1; i <= n; i++) {
16        std::cin >> a[i];
17        v.push_back(a[i]);
18    }
19    std::sort(all(v));
20    v.erase(unique(all(v)), v.end());
21    auto find = [&](int x) -> int { return std::lower_bound(all(v), x) - v.begin() + 1; };
22
23    /* hjt segment tree */
24    std::vector<node>(n * 25);
25    vi root(n + 1);
26    int idx = 0;
27
28    std::function<int(int, int)> build = [&](int l, int r) -> int {
29        int p = ++idx;
30        if (l == r) return p;
31        int mid = (l + r) >> 1;
32        tr[p].l = build(l, mid), tr[p].r = build(mid + 1, r);
33        return p;
34    };
35
36    std::function<int(int, int, int, int)> modify = [&](int p, int l, int r, int x) -> int {
37        int q = ++idx;
38        tr[q] = tr[p];
39        if (tr[q].l == tr[q].r) {
40            tr[q].cnt++;
41            return q;
42        }
43        int mid = (l + r) >> 1;

```

```

44     if (x <= mid) {
45         tr[q].l = modify(tr[q].l, l, mid, x);
46     } else {
47         tr[q].r = modify(tr[q].r, mid + 1, r, x);
48     }
49     tr[q].cnt = tr[tr[q].l].cnt + tr[tr[q].r].cnt;
50     return q;
51 };
52
53 std::function<int(int, int, int, int, int)> query = [&](int p, int q, int l, int r,
54                                                     int x) -> int {
55     if (l == r) return l;
56     int cnt = tr[tr[p].l].cnt - tr[tr[q].l].cnt;
57     int mid = (l + r) >> 1;
58     if (x <= cnt) {
59         return query(tr[p].l, tr[q].l, l, mid, x);
60     } else {
61         return query(tr[p].r, tr[q].r, mid + 1, r, x - cnt);
62     }
63 };
64
65 root[0] = build(1, v.size());
66
67
68 for (int i = 1; i <= n; i++) {
69     root[i] = modify(root[i - 1], 1, v.size(), find(a[i]));
70 }
71 for (int i = 1; i <= m; i++) {
72     int l, r, k;
73     std::cin >> l >> r >> k;
74     std::cout << v[query(root[r], root[l - 1], 1, v.size(), k) - 1] << '\n';
75 }
76
77 return 0;
78 }

```

3.9 sweep line

```

1  /* sweep line @ czr */
2  struct Node {
3      int l, r;
4      ll sum, length, res;
5  } tr[N << 2];
6
7  void push_up(int u) {
8      tr[u].res = (tr[u << 1].res + tr[u << 1 | 1].res) % Mod;
9  }
10
11 void update_length(int u) {
12     if (tr[u].sum) {
13         tr[u].length = tr[u].res;
14     } else {
15         if (tr[u].l == tr[u].r) tr[u].length = 0;
16         else tr[u].length = (tr[u << 1].length + tr[u << 1 | 1].length) % Mod;
17     }
18 }
19
20 void build(int u, int l, int r) {
21     if (l == r) tr[u] = {l, r, 0, 0, 0};
22     else {
23         tr[u] = {l, r, 0, 0, 0};
24         int mid = l + r >> 1;
25         build(u << 1, l, mid);
26         build(u << 1 | 1, mid + 1, r);
27         push_up(u);
28     }
29 }
30
31 void modify(int u, int l, int r, int op) {
32     if (tr[u].l >= l && tr[u].r <= r) {
33         tr[u].sum += op;
34         update_length(u);
35     } else {
36         int mid = tr[u].l + tr[u].r >> 1;
37         if (l <= mid) modify(u << 1, l, r, op);
38         if (r > mid) modify(u << 1 | 1, l, r, op);
39         push_up(u);
40         update_length(u);
41     }
42 }
43
44 void change(int u, int x, ll d) {

```

```

45 |     if (tr[u].l == tr[u].r) {
46 |         tr[u].res = (tr[u].res + d) % Mod;
47 |         update_length(u);
48 |     } else {
49 |         int mid = tr[u].l + tr[u].r >> 1;
50 |         if (x <= mid) change(u << 1, x, d);
51 |         else change(u << 1 | 1, x, d);
52 |         push_up(u);
53 |         update_length(u);
54 |     }
55 | }

```

```

1 | /* sweep line @ wrb */
2 | #define int long long
3 | const int N = 2e5+10;
4 | int b[N<<1], n, len, ans;
5 | struct node{
6 |     int y1, y2, x, k;
7 | }a[N<<1];
8 | struct Seg{
9 | #define lc (o<<1)
10 | #define rc (o<<1|1)
11 |     static const int N = 5e6+10;
12 |     int sum[N], cnt[N], tag[N];
13 |     void push_up(int o, int l, int r){
14 |         if(sum[o]) cnt[o] = b[r+1] - b[l];
15 |         else cnt[o] = cnt[lc] + cnt[rc];
16 |     }
17 |     void add(int o, int l, int r, int L, int R, int k){
18 |         if(r < L || l > R) return;
19 |         if(l == L && r == R) return (void)(sum[o] += k, push_up(o, l, r));
20 |         int mid = L + R >> 1;
21 |         if(r <= mid) add(lc, l, r, L, mid, k);
22 |         else if(l > mid) add(rc, l, r, mid+1, R, k);
23 |         else add(lc, l, mid, L, mid, k), add(rc, mid+1, r, mid+1, R, k);
24 |         push_up(o, L, R);
25 |     }
26 | #undef lc
27 | #undef rc
28 | }t;
29 | void work(){
30 |     cin >> n;
31 |     for(int i=1, x1, y1, x2, y2; i<=n; i++){
32 |         cin >> x1 >> y1 >> x2 >> y2;
33 |         b[i*2-1] = y1, b[i*2] = y2, a[i*2-1] = {y1, y2, x1, 1}, a[i*2] = {y1, y2, x2, -1};
34 |     }
35 |     n <<= 1;
36 |     sort(b+1, b+n+1), len = unique(b+1, b+n+1) - b - 1;
37 |     for(int i=1; i<=n; i++) a[i].y1 = lower_bound(b+1, b+len+1, a[i].y1) - b, a[i].y2 = lower_bound(b+1, b+len+1, a[i].y2) - b;
38 |     sort(a+1, a+n+1, [](node a, node b) -> bool {return a.x < b.x;});
39 |     for(int i=1; i<=n; i++){
40 |         t.add(1, a[i].y1, a[i].y2-1, 1, len-1, a[i].k);
41 |         ans += t.cnt[1] * (a[i+1].x - a[i].x);
42 |     }
43 |     cout << ans;
44 | }
45 | #undef int

```

3.10 treap

fhq treap

n 次操作, 操作分为如下 6 种:

1. 插入数 x ;
2. 删除数 x (若有多个相同的数, 只删除一个);
3. 查询数 x 的排名 (排名定义为小于 x 的数的个数 + 1);
4. 查询排名为 x 的数;
5. 求 x 的前驱 (前驱定义为小于 x 的最大数);

6. 求 x 的后继 (后继定义为大于 x 的最小数).

```

1 struct node {
2     node *ch[2];
3     int key, val;
4     int cnt, size;
5
6     node(int _key) : key(_key), cnt(1), size(1) {
7         ch[0] = ch[1] = nullptr;
8         val = rand();
9     }
10
11     // node(node *_node) {
12     //     key = _node->key, val = _node->val, cnt = _node->cnt, size = _node->size;
13     // }
14
15     inline void push_up() {
16         size = cnt;
17         if (ch[0] != nullptr) size += ch[0]->size;
18         if (ch[1] != nullptr) size += ch[1]->size;
19     }
20 };
21
22 struct treap {
23     #define _2 second.first
24     #define _3 second.second
25
26     node *root;
27
28     pair<node *, node *> split(node *p, int key) {
29         if (p == nullptr) return {nullptr, nullptr};
30         if (p->key <= key) {
31             auto temp = split(p->ch[1], key);
32             p->ch[1] = temp.first;
33             p->push_up();
34             return {p, temp.second};
35         } else {
36             auto temp = split(p->ch[0], key);
37             p->ch[0] = temp.second;
38             p->push_up();
39             return {temp.first, p};
40         }
41     }
42
43     pair<node *, pair<node *, node *> > split_by_rank(node *p, int rank) {
44         if (p == nullptr) return {nullptr, {nullptr, nullptr}};
45         int ls_size = p->ch[0] == nullptr ? 0 : p->ch[0]->size;
46         if (rank <= ls_size) {
47             auto temp = split_by_rank(p->ch[0], rank);
48             p->ch[0] = temp._3;
49             p->push_up();
50             return {temp.first, {temp._2, p}};
51         } else if (rank <= ls_size + p->cnt) {
52             node *lt = p->ch[0];
53             node *rt = p->ch[1];
54             p->ch[0] = p->ch[1] = nullptr;
55             return {lt, {p, rt}};
56         } else {
57             auto temp = split_by_rank(p->ch[1], rank - ls_size - p->cnt);
58             p->ch[1] = temp.first;
59             p->push_up();
60             return {p, {temp._2, temp._3}};
61         }
62     }
63
64     node *merge(node *u, node *v) {
65         if (u == nullptr && v == nullptr) return nullptr;
66         if (u != nullptr && v == nullptr) return u;
67         if (v != nullptr && u == nullptr) return v;
68         if (u->val < v->val) {
69             u->ch[1] = merge(u->ch[1], v);
70             u->push_up();
71             return u;
72         } else {
73             v->ch[0] = merge(u, v->ch[0]);
74             v->push_up();
75             return v;
76         }
77     }
78
79     void insert(int key) {
80         auto temp = split(root, key);
81         auto l_tr = split(temp.first, key - 1);
82         node *new_node;
83         if (l_tr.second == nullptr) {
84             new_node = new node(key);

```

```

85     } else {
86         l_tr.second->cnt++;
87         l_tr.second->push_up();
88     }
89     node *l_tr_combined = merge(l_tr.first, l_tr.second == nullptr ? new_node : l_tr.second);
90     root = merge(l_tr_combined, temp.second);
91 }
92
93 void remove(int key) {
94     auto temp = split(root, key);
95     auto l_tr = split(temp.first, key - 1);
96     if (l_tr.second->cnt > 1) {
97         l_tr.second->cnt--;
98         l_tr.second->push_up();
99         l_tr.first = merge(l_tr.first, l_tr.second);
100    } else {
101        if (temp.first == l_tr.second) temp.first = nullptr;
102        delete l_tr.second;
103        l_tr.second = nullptr;
104    }
105    root = merge(l_tr.first, temp.second);
106 }
107
108 int get_rank_by_key(node *p, int key) {
109     auto temp = split(p, key - 1);
110     int ret = (temp.first == nullptr ? 0 : temp.first->size) + 1;
111     root = merge(temp.first, temp.second);
112     return ret;
113 }
114
115 int get_key_by_rank(node *p, int rank) {
116     auto temp = split_by_rank(p, rank);
117     int ret = temp._2->key;
118     root = merge(temp.first, merge(temp._2, temp._3));
119     return ret;
120 }
121
122 int get_prev(int key) {
123     auto temp = split(root, key - 1);
124     int ret = get_key_by_rank(temp.first, temp.first->size);
125     root = merge(temp.first, temp.second);
126     return ret;
127 }
128
129 int get_nex(int key) {
130     auto temp = split(root, key);
131     int ret = get_key_by_rank(temp.second, 1);
132     root = merge(temp.first, temp.second);
133     return ret;
134 }
135 };
136
137 treap tr;
138
139 int main() {
140     ios::sync_with_stdio(false);
141     cin.tie(0);
142     cout.tie(0);
143
144     srand(time(0));
145
146     int n;
147     cin >> n;
148     while (n-- > 0) {
149         int op, x;
150         cin >> op >> x;
151         if (op == 1) {
152             tr.insert(x);
153         } else if (op == 2) {
154             tr.remove(x);
155         } else if (op == 3) {
156             cout << tr.get_rank_by_key(tr.root, x) << '\n';
157         } else if (op == 4) {
158             cout << tr.get_key_by_rank(tr.root, x) << '\n';
159         } else if (op == 5) {
160             cout << tr.get_prev(x) << '\n';
161         } else {
162             cout << tr.get_nex(x) << '\n';
163         }
164     }
165     return 0;
166 }

```

用 01 trie 实现的一种方式

同样的题目, 注意使用 01 trie 只能存在非负数. 速度能快不少, 但只能单点操作, 而且有点费空间.

```

1 // 洛谷 P3369 【模板】普通平衡树
2
3 struct Treap {
4     int id = 1, maxlog = 25;
5     int ch[N * 25][2], siz[N * 25];
6
7     int newnode() {
8         id++;
9         ch[id][0] = ch[id][1] = siz[id] = 0;
10        return id;
11    }
12
13    void merge(int key, int cnt) {
14        int u = 1;
15        for (int i = maxlog - 1; i >= 0; i--) {
16            int v = (key >> i) & 1;
17            if (!ch[u][v]) ch[u][v] = newnode();
18            u = ch[u][v];
19            siz[u] += cnt;
20        }
21    }
22
23    int get_key_by_rank(int rank) {
24        int u = 1, key = 0;
25        for (int i = maxlog - 1; i >= 0; i--) {
26            if (siz[ch[u][0]] >= rank) {
27                u = ch[u][0];
28            } else {
29                key |= (1 << i);
30                rank -= siz[ch[u][0]];
31                u = ch[u][1];
32            }
33        }
34        return key;
35    }
36
37    int get_rank_by_key(int rank) {
38        int key = 0;
39        int u = 1;
40        for (int i = maxlog - 1; i >= 0; i--) {
41            if ((rank >> i) & 1) {
42                key += siz[ch[u][0]];
43                u = ch[u][1];
44            } else {
45                u = ch[u][0];
46            }
47            if (!u) break;
48        }
49        return key;
50    }
51
52    int get_prev(int x) { return get_key_by_rank(get_rank_by_key(x)); }
53    int get_next(int x) { return get_key_by_rank(get_rank_by_key(x + 1) + 1); }
54 } treap;
55
56 const int num = 1e7;
57 int n, op, x;
58
59 int main() {
60     std::ios::sync_with_stdio(false);
61     std::cin.tie(0);
62     std::cout.tie(0);
63
64     std::cin >> n;
65     for (int i = 1; i <= n; i++) {
66         std::cin >> op >> x;
67         if (op == 1) {
68             treap.merge(x + num, 1);
69         } else if (op == 2) {
70             treap.merge(x + num, -1);
71         } else if (op == 3) {
72             std::cout << treap.get_rank_by_key(x + num) + 1 << '\n';
73         } else if (op == 4) {
74             std::cout << treap.get_key_by_rank(x) - num << '\n';
75         } else if (op == 5) {
76             std::cout << treap.get_prev(x + num) - num << '\n';
77         } else if (op == 6) {
78             std::cout << treap.get_next(x + num) - num << '\n';
79         }
80     }
81     return 0;

```

82 | }

3.11 splay

文艺平衡树

初始为 1 到 n 的序列, m 次操作, 每次将序列下标为 $[l \sim r]$ 的区间翻转.

```

1 // 洛谷 P3391 【模板】文艺平衡树
2
3 struct node {
4     int ch[2], fa, key;
5     int siz, flag;
6
7     void init(int _fa, int _key) { fa = _fa, key = _key, siz = 1; }
8 };
9
10 struct splay {
11     node tr[N];
12     int n, root, idx;
13
14     bool get(int u) { return u == tr[tr[u].fa].ch[1]; }
15
16     void pushup(int u) { tr[u].siz = tr[tr[u].ch[0]].siz + tr[tr[u].ch[1]].siz + 1; }
17
18     void pushdown(int u) {
19         if (tr[u].flag) {
20             std::swap(tr[u].ch[0], tr[u].ch[1]);
21             tr[tr[u].ch[0]].flag ^= 1, tr[tr[u].ch[1]].flag ^= 1;
22             tr[u].flag = 0;
23         }
24     }
25
26     void rotate(int x) {
27         int y = tr[x].fa, z = tr[y].fa;
28         int op = get(x);
29         tr[y].ch[op] = tr[x].ch[op ^ 1];
30         if (tr[x].ch[op ^ 1]) tr[tr[x].ch[op ^ 1]].fa = y;
31         tr[x].ch[op ^ 1] = y;
32         tr[y].fa = x, tr[x].fa = z;
33         if (z) tr[z].ch[y == tr[z].ch[1]] = x;
34         pushup(y), pushup(x);
35     }
36
37     void opt(int u, int k) {
38         for (int f = tr[u].fa; f = tr[u].fa, f != k; rotate(u)) {
39             if (tr[f].fa != k) rotate(get(u) == get(f) ? f : u);
40         }
41         if (k == 0) root = u;
42     }
43
44     void output(int u) {
45         pushdown(u);
46         if (tr[u].ch[0]) output(tr[u].ch[0]);
47         if (tr[u].key >= 1 && tr[u].key <= n) {
48             std::cout << tr[u].key << ' ';
49         }
50         if (tr[u].ch[1]) output(tr[u].ch[1]);
51     }
52
53     void insert(int key) {
54         idx++;
55         tr[idx].ch[0] = root;
56         tr[idx].init(0, key);
57         tr[root].fa = idx;
58         root = idx;
59         pushup(idx);
60     }
61
62     int kth(int k) {
63         int u = root;
64         while (1) {
65             pushdown(u);
66             if (tr[u].ch[0] && k <= tr[tr[u].ch[0]].siz) {
67                 u = tr[u].ch[0];
68             } else {
69                 k -= tr[tr[u].ch[0]].siz + 1;
70                 if (k <= 0) {
71                     opt(u, 0);
72                     return u;
73                 } else {

```

```

74         u = tr[u].ch[1];
75     }
76 }
77 }
78 }
79
80 } splay;
81
82 int n, m, l, r;
83
84 int main() {
85     std::ios::sync_with_stdio(false);
86     std::cin.tie(0);
87     std::cout.tie(0);
88
89     std::cin >> n >> m;
90     splay.n = n;
91     splay.insert(-inf);
92     rep(i, 1, n) splay.insert(i);
93     splay.insert(inf);
94     rep(i, 1, m) {
95         std::cin >> l >> r;
96         l = splay.kth(l), r = splay.kth(r + 2);
97         splay.opt(l, 0), splay.opt(r, 1);
98         splay.tr[splay.tr[r].ch[0]].flag ^= 1;
99     }
100     splay.output(splay.root);
101
102     return 0;
103 }

```

普通平衡树

```

1 // 洛谷 P3369 【模板】普通平衡树
2
3 struct node {
4     int ch[2], fa, key, siz, cnt;
5
6     void init(int _fa, int _key) { fa = _fa, key = _key, siz = cnt = 1; }
7
8     void clear() { ch[0] = ch[1] = fa = key = siz = cnt = 0; }
9 };
10
11 struct splay {
12     node tr[N];
13     int n, root, idx;
14
15     bool get(int u) { return u == tr[tr[u].fa].ch[1]; }
16
17     void pushup(int u) { tr[u].siz = tr[tr[u].ch[0]].siz + tr[tr[u].ch[1]].siz + tr[u].cnt; }
18
19     void rotate(int x) {
20         int y = tr[x].fa, z = tr[y].fa;
21         int op = get(x);
22         tr[y].ch[op] = tr[x].ch[op ^ 1];
23         if (tr[x].ch[op ^ 1]) tr[tr[x].ch[op ^ 1]].fa = y;
24         tr[x].ch[op ^ 1] = y;
25         tr[y].fa = x, tr[x].fa = z;
26         if (z) tr[z].ch[y == tr[z].ch[1]] = x;
27         pushup(y), pushup(x);
28     }
29
30     void opt(int u, int k) {
31         for (int f = tr[u].fa; f = tr[u].fa, f != k; rotate(u)) {
32             if (tr[f].fa != k) {
33                 rotate(get(u) == get(f) ? f : u);
34             }
35         }
36         if (k == 0) root = u;
37     }
38
39     void insert(int key) {
40         if (!root) {
41             idx++;
42             tr[idx].init(0, key);
43             root = idx;
44             return;
45         }
46         int u = root, f = 0;
47         while (1) {
48             if (tr[u].key == key) {
49                 tr[u].cnt++;
50                 pushup(u), pushup(f);

```

```

51         opt(u, 0);
52         break;
53     }
54     f = u, u = tr[u].ch[tr[u].key < key];
55     if (!u) {
56         idx++;
57         tr[idx].init(f, key);
58         tr[f].ch[tr[f].key < key] = idx;
59         pushup(idx), pushup(f);
60         opt(idx, 0);
61         break;
62     }
63 }
64 }
65
66 // 返回节点编号 //
67 int kth(int rank) {
68     int u = root;
69     while (1) {
70         if (tr[u].ch[0] && rank <= tr[tr[u].ch[0]].siz) {
71             u = tr[u].ch[0];
72         } else {
73             rank -= tr[tr[u].ch[0]].siz + tr[u].cnt;
74             if (rank <= 0) {
75                 opt(u, 0);
76                 return u;
77             } else {
78                 u = tr[u].ch[1];
79             }
80         }
81     }
82 }
83
84 // 返回排名 //
85 int nlt(int key) {
86     int rank = 0, u = root;
87     while (1) {
88         if (tr[u].key > key) {
89             u = tr[u].ch[0];
90         } else {
91             rank += tr[tr[u].ch[0]].siz;
92             if (tr[u].key == key) {
93                 opt(u, 0);
94                 return rank + 1;
95             }
96             rank += tr[u].cnt;
97             if (tr[u].ch[1]) {
98                 u = tr[u].ch[1];
99             } else {
100                 return rank + 1;
101             }
102         }
103     }
104 }
105
106 int get_prev(int key) { return kth(nlt(key) - 1); }
107
108 int get_next(int key) { return kth(nlt(key) + 1); }
109
110 void remove(int key) {
111     nlt(key);
112     if (tr[root].cnt > 1) {
113         tr[root].cnt--;
114         pushup(root);
115         return;
116     }
117     int u = root, l = get_prev(key);
118     tr[tr[u].ch[1]].fa = l;
119     tr[l].ch[1] = tr[u].ch[1];
120     tr[u].clear();
121     pushup(root);
122 }
123
124 void output(int u) {
125     if (tr[u].ch[0]) output(tr[u].ch[0]);
126     std::cout << tr[u].key << ' ';
127     if (tr[u].ch[1]) output(tr[u].ch[1]);
128 }
129
130 } splay;
131
132 int n, op, x;
133
134 int main() {
135     std::ios::sync_with_stdio(false);
136     std::cin.tie(0);
137     std::cout.tie(0);

```

```

138
139 splay.insert(-inf), splay.insert(inf);
140
141 std::cin >> n;
142 for (int i = 1; i <= n; i++) {
143     std::cin >> op >> x;
144     if (op == 1) {
145         splay.insert(x);
146     } else if (op == 2) {
147         splay.remove(x);
148     } else if (op == 3) {
149         std::cout << splay.nlt(x) - 1 << endl;
150     } else if (op == 4) {
151         std::cout << splay.tr[splay.kth(x + 1)].key << endl;
152     } else if (op == 5) {
153         std::cout << splay.tr[splay.get_prev(x)].key << endl;
154     } else if (op == 6) {
155         std::cout << splay.tr[splay.get_next(x)].key << endl;
156     }
157 }
158
159 return 0;
160 }

```

3.12 link cut tree

```

1  /* link cut tree @ wrb */
2  struct LCT{
3      int v[N], r[N], f[N], s[N][2], st[N], tp;
4      void pu(int x){v[x]=a[x]^v[s[x][0]]^v[s[x][1]};}
5      void flp(int x){r[x]^=1, std::swap(s[x][0], s[x][1]);}
6      void pd(int x){if(r[x])flp(s[x][0]), flp(s[x][1]), r[x]=0;}
7      bool isrt(int x){return s[f[x]][0]!=x&&s[f[x]][1]!=x;}
8      void rtt(int x){
9          int y=f[x], z=f[y], k=(s[y][1]==x); if(!isrt(y)) s[z][y==s[z][1]]=x;
10         f[x]=z, f[y]=x, f[s[x][k^1]]=y, s[y][k]=s[x][k^1], s[x][k^1]=y, pu(y), pu(x);
11     }
12     void spl(int x){
13         st[tp++]=x; for(int i=x; !isrt(i); i=f[i]) st[tp++]=f[i];
14         while(tp)pd(st[--tp]);
15         while(!isrt(x)){
16             if(!isrt(f[x]))rtt((s[f[x]][0]==x)^(s[f[f[x]]][0]==f[x])?x:f[x]);
17             rtt(x);
18         }
19         pu(x);
20     }
21     void acc(int x){for(int y=0; x=y, x=f[x]) spl(x), s[x][1]=y, pu(x);}
22     void mkrt(int x){acc(x), spl(x), flp(x);}
23     int fdrt(int x){acc(x), spl(x); while(s[x][0])x=s[x][0]; spl(x); return x;}
24     void cut(int x, int y){mkrt(x); if(x==fdrt(y)&&f[y]==x&&!s[y][0])s[x][1]=f[y]=0, pu(x);}
25     void lk(int x, int y){mkrt(x); if(x!=fdrt(y))f[x]=y;}
26 }t;

```

3.13 Lichao tree

```

1  /* Lichao tree @ wrb */
2  #include<bits/stdc++.h>
3  using namespace std;
4  namespace Acc{
5      #define lc (o<<1)
6      #define rc (o<<1|1)
7      const int N = 4e5+10;
8      int v[N], n, l, r, z;
9      double k[N], b[N];
10     inline void r1(int&x){x=(x+z-1)%39989+1;}
11     inline void r2(int&x){x=(x+z-1)%1000000000+1;}
12     double f(int o, int x){
13         return k[o]*x+b[o];
14     }
15     int beat(int x, int a, int b){
16         double u=f(a, x), v=f(b, x);
17         return fabs(u-v)<=1e-8?a<b:u>v;
18     }
19     void add(int o, int L, int R, int x){
20         int md=L+R>>1;
21         if(L<=L&&R<=R){
22             if(!v[o])return (void)(v[o]=x);
23             if(beat(L, v[o], x) && beat(R, v[o], x))return;
24             if(beat(L, x, v[o]) && beat(R, x, v[o]))return (void)(v[o]=x);
25             if(beat(md, x, v[o]))swap(x, v[o]);

```

```

26         if(beat(L,x,v[o]))add(lc,L,md,x);
27         else add(rc,md+1,R,x);
28         return;
29     }
30     if(r>md)add(rc,md+1,R,x);
31     if(l<=md)add(lc,L,md,x);
32 }
33 int ask(int o,int L,int R){
34     if(L==R)return v[o];
35     int md=L+R>>1,h=l<=md?ask(lc,L,md):ask(rc,md+1,R);
36     return beat(l,h,v[o])?h:v[o];
37 }
38 void work(){
39     cin>>n;
40     for(int op,y1,y2,c=0;n--;){
41         cin>>op;
42         if(op){
43             cin>>l>>y1>>r>>y2,++c,r1(l),r2(y1),r1(r),r2(y2);
44             if(l==r)k[c]=0,b[c]=max(y1,y2);
45             else {
46                 if(l>r)swap(l,r),swap(y1,y2);
47                 k[c]=(y2-y1+0.)/(r-l),b[c]=y1-k[c]*l;
48             }
49             add(1,1,4e4+10,c);
50         }else cin>>l,r1(l),cout<<(z=ask(1,1,4e4+10))<<'\\n';
51     }
52 }
53 }
54 int main(){
55     return Acc::work(),0;
56 }

```

3.14 ODT

```

1  /* ODT @ wrb */
2  struct T{
3      int l,r,v;
4      T(int a,int b=-1,int c=-1):l(a),r(b),v(c){}
5      bool operator<(const T&_)const{return l<_.l;}
6  };
7  set<T>s;
8  auto spl(int p){
9      auto it=s.lower_bound(p);
10     if(it!=end(s) && it->l==p)return it;
11     --it;
12     int l=it->l,r=it->r,v=it->v;
13     s.erase(it),s.insert(T(l,p-1,v));
14     return s.insert(T(p,r,v)).first;
15 }
16 void asgn(int l,int r,int v){
17     auto ed=spl(r+1),bg=spl(l);
18     s.erase(bg,ed);
19     auto i=s.insert(T(l,r,v)).first,j=prev(i);
20     if(i!=begin(s)&&j->v==v)l=j->l,s.erase(j);
21     if((j=next(i))!=end(s)&&j->v==v)r=j->r,s.erase(j);
22     s.erase(i),s.insert(T(l,r,v));
23 }

```

4 string

4.1 kmp

```

1  /* kmp */
2  auto kmp = [&](const std::string& s) -> vi {
3      int n = s.length();
4      vi next(n);
5      for (int i = 1; i < n; i++) {
6          int j = next[i - 1];
7          while (j > 0 and s[i] != s[j]) j = next[j - 1];
8          if (s[i] == s[j]) j++;
9          next[i] = j;
10     }
11     return next;
12 };

```


4.2 z function

```

1  /* exkmp */
2  auto exkmp = [&](const std::string& s) -> vi {
3      int n = s.size();
4      vi z(n);
5      for (int i = 1, l = 0, r = 0; i < n; i++) {
6          if (i <= r and z[i - l] < r - i + 1) {
7              z[i] = z[i - l];
8          } else {
9              z[i] = std::max(0, r - i + 1);
10             while (z[i] + i < n and s[z[i]] == s[z[i] + i]) z[i]++;
11         }
12         if (z[i] + i - 1 > r) {
13             l = i;
14             r = z[i] + i - 1;
15         }
16     }
17     return z;
18 };

```

4.3 manacher

```

1  /* manacher @ wrb */
2  auto Manacher = [&](const std::string& t) {
3      std::string s = "#";
4      for (char c : t) s += c, s += '#';
5      int i, o = 0, r = 0, n = s.size();
6      std::vector<int> p(n, 1), q(n);
7      for (i = 0; i < n; ++i) {
8          if (i <= r) p[i] = std::min(r - i + 1, p[2 * o - i]);
9          for (; p[i] <= i && s[i + p[i]] == s[i - p[i]]; ++p[i]);
10         if (i + p[i] - 1 > r) r = i + p[i] - 1, o = i;
11     }
12     return p;
13 };

```

4.4 AC automaton

```

1  /* AC auto */
2  int cnt = 0;
3  const int N = 2e5 + 10;
4  static std::array<std::array<int, 26>, N> tr;
5  static std::array<int, N> exist, fail, ans, point;
6  vi order;
7
8  auto insert = [&](const auto& s) {
9      int p = 0;
10     for (const auto& ch : s) {
11         int c = ch - 'a';
12         if (!tr[p][c]) tr[p][c] = ++cnt;
13         p = tr[p][c];
14     }
15     exist[p]++;
16     return p;
17 };
18
19 auto build = [&]() {
20     std::queue<int> q;
21     for (int i = 0; i < 26; i++) {
22         if (tr[0][i]) q.push(tr[0][i]);
23     }
24     while (!q.empty()) {
25         auto u = q.front();
26         q.pop();
27         order.push_back(u);
28         for (int i = 0; i < 26; i++) {
29             if (tr[u][i]) {
30                 fail[tr[u][i]] = tr[fail[u]][i];
31                 q.push(tr[u][i]);
32             } else {
33                 tr[u][i] = tr[fail[u]][i];
34             }
35         }
36     }
37 };

```

```

38 |
39 | auto query = [&](const auto& s) {
40 |     int p = 0;
41 |     for (const auto ch : s) {
42 |         p = tr[p][ch - 'a'];
43 |         ans[p]++;
44 |     }
45 |     return;
46 | };
47 |
48 | void solve () {
49 |     for (int i = 0; i < n; i++) {
50 |         point[i] = insert(t);
51 |     }
52 |     build();
53 |     query(s);
54 |     /* fail 树上子树求和 */
55 |     reverse(all(order));
56 |     for (const auto& i : order) ans[fail[i]] += ans[i];
57 | }

```

4.5 PAM

```

1  | /* PAM @ ddl */
2  | std::vector<node> tr;
3  | std::vector<int> stk;
4  | auto newnode = [&](int len) {
5  |     tr.emplace_back();
6  |     tr.back().len = len;
7  |     return (int) tr.size() - 1;
8  | };
9  | auto PAMinit = [&]() {
10 |     newnode(0), tr.back().fail = 1;
11 |     newnode(-1), tr.back().fail = 0;
12 |     stk.push_back(-1);
13 | };
14 | PAMinit();
15 | auto getfail = [&](int v) {
16 |     while (stk.end()[-2 - tr[v].len] != stk.back()) {
17 |         v = tr[v].fail;
18 |     }
19 |     return v;
20 | };
21 | auto insert = [&](int last, int c, int cnt) {
22 |     stk.emplace_back(c);
23 |     int x = getfail(last);
24 |     if (!tr[x].ch[c]) {
25 |         int u = newnode(tr[x].len + 2);
26 |         tr[u].fail = tr[getfail(tr[x].fail)].ch[c];
27 |         tr[x].ch[c] = u;
28 |         /* tr[u].size = tr[tr[u].fail].size + 1; */
29 |         /* Can be used to count the number of types of palindromic strings ending at the current
30 |          * position */
31 |     }
32 |     tr[tr[x].ch[c]].size += cnt;
33 |     return tr[x].ch[c];
34 | };
35 | auto build = [&]() { /* DP on fail tree */
36 |     int ans = 0;
37 |     for (int i = (int) tr.size() - 1; i > 1; i--) {
38 |         tr[tr[i].fail].size += tr[i].size;
39 |         /* options */
40 |     }
41 |     return ans;
42 | };
43 | /* PAM */
44 | int ans = 0, last = 0;
45 | for (int i = 0; i < n; i++) {
46 |     last = insert(last, s[i] - 'a', 1);
47 | }

```

```

1  | /* PAM @ wrb */
2  | template<const int M = 26>
3  | struct PAM {
4  |     struct T {
5  |         int len, d, fa, ch[M];
6  |         T() : len(), d(), fa(), ch() {}
7  |     };
8  |     int las;
9  |     string s;
10 |     vector<T> t;

```

```

11     vector<int> bl;
12     size_t count() const {
13         return t.size() - 2;
14     }
15     const T& operator[](const size_t& p) const {
16         return t[p];
17     }
18     const T& ask(const size_t& p) const {
19         return t[bl[p]];
20     }
21     int gf(int o, int p) {
22         while (p - t[o].len - 1 < 0 || s[p - t[o].len - 1] != s[p]) o = t[o].fa;
23         return o;
24     }
25     void append(int c) {
26         int p = s.size(), o;
27         s += c, o = gf(las, p);
28         if (t[o].ch[c] == 0) {
29             t.emplace_back();
30             t.back().len = t[o].len + 2;
31             t.back().fa = t[gf(t[o].fa, p)].ch[c];
32             t.back().d = t[t.back().fa].d + 1;
33             t[o].ch[c] = t.size() - 1;
34         }
35         bl.emplace_back(las = t[o].ch[c]);
36     }
37     PAM() : las(), s(), t(2) {
38         t[0].fa = t[1].fa = 1, t[1].len = -1;
39     }
40     PAM(const string& str, int h) : las(), s(), t(2) {
41         t[0].fa = t[1].fa = 1, t[1].len = -1;
42         for (char c : str) append(c - h);
43     }
44 };

```

4.6 Suffix Array

```

1  /* suffix array and ST table @ jiangly */
2  auto suffixArray = [&](const std::string& s) {
3      int n = s.length();
4      vi sa(n), rk(n);
5      std::iota(all(sa), 0);
6      std::sort(all(sa), [&](int a, int b) { return s[a] < s[b]; });
7      rk[sa[0]] = 0;
8      for (int i = 1; i < n; ++i) {
9          rk[sa[i]] = rk[sa[i - 1]] + (s[sa[i]] != s[sa[i - 1]]);
10     }
11     int k = 1;
12     vi tmp(n), cnt(n);
13     tmp.reserve(n);
14     while (rk[sa[n - 1]] < n - 1) {
15         tmp.clear();
16         for (int i = 0; i < k; ++i) tmp.push_back(n - k + i);
17         for (const auto& i : sa) {
18             if (i >= k) tmp.push_back(i - k);
19         }
20         std::fill(all(cnt), 0);
21         for (int i = 0; i < n; i++) cnt[rk[i]]++;
22         for (int i = 1; i < n; i++) cnt[i] += cnt[i - 1];
23         for (int i = n - 1; i >= 0; i--) sa[--cnt[rk[tmp[i]]]] = tmp[i];
24         std::swap(rk, tmp);
25         rk[sa[0]] = 0;
26         for (int i = 1; i < n; i++) {
27             rk[sa[i]] = rk[sa[i - 1]] + (tmp[sa[i - 1]] < tmp[sa[i]] or sa[i - 1] + k == n or
28                                     tmp[sa[i - 1] + k] < tmp[sa[i] + k]);
29         }
30         k *= 2;
31     }
32     vi height(n);
33     for (int i = 0, j = 0; i < n; ++i) {
34         if (rk[i] == 0) continue;
35         if (j) --j;
36         while (s[i + j] == s[sa[rk[i] - 1] + j]) ++j;
37         height[rk[i]] = j;
38     }
39     return std::make_tuple(sa, rk, height);
40 };
41 auto [sa, rk, height] = suffixArray(s);
42 vvi f(n, vi(30, inf));
43 vi Log2(n);
44 auto init = [&]() -> void {
45     for (int i = 0; i < n; i++) {

```

```

46     f[i][0] = height[i];
47     if (i > 1) Log2[i] = Log2[i / 2] + 1;
48 };
49 int t = Log2.back();
50 for (int j = 1; j <= t; j++) {
51     for (int i = 0; i <= n - (1 << j); i++) {
52         f[i][j] = std::min(f[i][j - 1], f[i + (1 << (j - 1))][j - 1]);
53     }
54 }
55 };
56 init();
57 auto query = [&](int l, int r) -> int {
58     int t = Log2[r - l + 1];
59     return std::min(f[l][t], f[r - (1 << t) + 1][t]);
60 };
61 auto lcp = [&](int i, int j) {
62     i = rk[i], j = rk[j];
63     if (i > j) std::swap(i, j);
64     return query(i + 1, j);
65 };

```

```

1  /* suffix array @ wrb */
2  auto SA = [](std::string s) {
3      int n = s.size(), m = 128, i, j, l;
4      std::vector<int> ct(m), sa(n), rk(n), h(n), a(n);
5      for (i = 0; i < n; ++i) ++ct[rk[i] = s[i]];
6      for (i = 1; i < m; ++i) ct[i] += ct[i - 1];
7      for (i = n - 1; ~i; --i) sa[--ct[rk[i]]] = i;
8      for (l = 1; l < n; l *= 2) {
9          for (j = 0, i = n - 1; i >= n - l; --i) a[j++] = i;
10         for (i = 0; i < n; ++i) if (sa[i] >= l) a[j++] = sa[i] - l;
11         ct = std::vector<int>(m);
12         for (i = 0; i < n; ++i) ++ct[rk[a[i]]];
13         for (i = 1; i < m; ++i) ct[i] += ct[i - 1];
14         for (i = n - 1; ~i; --i) sa[--ct[rk[a[i]]]] = a[i];
15         std::swap(rk, a), rk[sa[0]] = 0;
16         for (i = 1; i < n; ++i) {
17             rk[sa[i]] = rk[sa[i - 1]] + (a[sa[i]] != a[sa[i - 1]] || a[(sa[i] + l) % n] != a[(sa[i - 1] +
18                 l) % n]);
19         }
20         if ((m = rk[sa[n - 1]] + 1) == n) break;
21     }
22     for (i = j = 0; i + 1 < n; h[rk[i++]] = j) {
23         for (j ? --j : 0; s[i + j] == s[sa[rk[i] - 1] + j]; ++j);
24     }
25     sa.erase(sa.begin());
26     rk.erase(rk.begin());
27     h.erase(h.begin());
28     return make_tuple(sa, rk, h);
29 };
30 //h[i] : LCP(rk[i], rk[i - 1])

```

4.7 Cantor expansion

```

1  /* Cantor expression @ wrb */
2  std::cin >> n, fac[0] = 1;
3  for (int i = 1; i <= n; ++i) {
4      std::cin >> a[i];
5      fac[i] = 1ll * fac[i - 1] * i % P;
6  }
7  auto ins = [&](int x) {
8      for (; x <= n; x += x & -x) ++t[x];
9  };
10 auto ask = [&](int x) {
11     int z = 0;
12     for (; x; x ^= x & -x) z += t[x];
13     return z;
14 };
15 int z = 0;
16 for (int i = n; i; --i) {
17     z = (z + 1ll * fac[n - i] * ask(a[i])) % P;
18     ins(a[i]);
19 }
20 std::cout << ++z << '\n';

```

4.8 trie

普通字典树 (单词匹配)

```

1  /* trie */
2  int cnt;
3  std::vector<std::array<int, 26>> trie(n + 1);
4  vi exist(n + 1);
5  auto insert = [&](const std::string& s) -> void {
6      int p = 0;
7      for (const auto ch : s) {
8          int c = ch - 'a';
9          if (!trie[p][c]) trie[p][c] = ++cnt;
10         p = trie[p][c];
11     }
12     exist[p] = true;
13 };
14 auto find = [&](const string& s) -> bool {
15     int p = 0;
16     for (const auto ch : s) {
17         int c = ch - 'a';
18         if (!trie[p][c]) return false;
19         p = trie[p][c];
20     }
21     return exist[p];
22 };

```

01 字典树 (求最大异或值)

给定 n 个数, 取两个数进行异或运算, 求最大异或值.

```

1  /* trie */
2  int cnt = 0;
3  std::vector<std::array<int, 2>> trie(N);
4  auto insert = [&](int x) -> void {
5      int p = 0;
6      for (int i = 30; i >= 0; i--) {
7          int c = (x >> i) & 1;
8          if (!trie[p][c]) trie[p][c] = ++cnt;
9          p = trie[p][c];
10     }
11 };
12 auto find = [&](int x) -> int {
13     int sum = 0, p = 0;
14     for (int i = 30; i >= 0; i--) {
15         int c = (x >> i) & 1;
16         if (trie[p][c ^ 1]) {
17             p = trie[p][c ^ 1];
18             sum += (1 << i);
19         } else {
20             p = trie[p][c];
21         }
22     }
23     return sum;
24 };

```

字典树合并

来自浙大城市学院 2023 校赛 E 题.

给定一棵根为 1 的树, 每个点的点权为 w_i . 一共 q 次询问, 每次给出一对 u, v , 询问以 v 为根的子树上的点与 u 的权值最大异或值.

```

1  int main() {
2      std::ios::sync_with_stdio(false);
3      std::cin.tie(0);
4
5      int n, m;
6      std::cin >> n;
7      vi w(n + 1);
8      for (int i = 1; i <= n; i++) std::cin >> w[i];
9      vvi e(n + 1);
10     for (int i = 1, u, v; i < n; i++) {

```

```

11     std::cin >> u >> v;
12     e[u].push_back(v);
13     e[v].push_back(u);
14 }
15
16 // 离线询问 //
17 std::cin >> m;
18 std::vector<vpi> q(n + 1);
19 vi ans(m + 1);
20 for (int i = 1; i <= m; i++) {
21     int u, v;
22     std::cin >> u >> v;
23     q[v].emplace_back(u, i);
24 }
25
26 // 01 trie //
27 std::vector<std::array<int, 2>> tr(1);
28 auto new_node = [&]() -> int {
29     tr.emplace_back();
30     return tr.size() - 1;
31 };
32 vi id(n + 1);
33 auto insert = [&](int root, int x) {
34     int p = root;
35     for (int i = 29; i >= 0; i--) {
36         int c = x >> i & 1;
37         if (!tr[p][c]) tr[p][c] = new_node();
38         p = tr[p][c];
39     }
40 };
41 auto query = [&](int root, int x) -> int {
42     int ans = 0, p = root;
43     for (int i = 29; i >= 0; i--) {
44         int c = x >> i & 1;
45         if (tr[p][c ^ 1]) {
46             p = tr[p][c ^ 1];
47             ans += (1 << i);
48         } else {
49             p = tr[p][c];
50         }
51     }
52     return ans;
53 };
54 std::function<int(int, int)> merge = [&](int a, int b) -> int {
55     // b 的信息挪到 a 上 //
56     if (!a) return b;
57     if (!b) return a;
58     tr[a][0] = merge(tr[a][0], tr[b][0]);
59     tr[a][1] = merge(tr[a][1], tr[b][1]);
60     return a;
61 };
62 std::function<void(int, int)> dfs = [&](int u, int fa) {
63     id[u] = new_node();
64     insert(id[u], w[u]);
65     for (auto v : e[u]) {
66         if (v == fa) continue;
67         dfs(v, u);
68         id[u] = merge(id[u], id[v]);
69     }
70     for (auto [v, i] : q[u]) {
71         ans[i] = query(id[u], w[v]);
72     }
73 };
74 dfs(1, 0);
75 for (int i = 1; i <= m; i++) std::cout << ans[i] << endl;
76 return 0;
77 }

```

5 math - number theory

5.1 mod int

```

1  template <int P>
2  struct Mint {
3      int v = 0;
4
5      // reflection //
6      template <typet = int>
7      constexpr operator T() const {
8          return v;
9      }
10
11     // constructor //
12     constexpr Mint() = default;
13     template <typet>
14     constexpr Mint(T x) : v(x % P) {}
15     constexpr int val() const { return v; }
16     constexpr int mod() { return P; }
17
18     // io //
19     friend std::istream& operator>>(std::istream& is, Mint& x) {
20         LL y;
21         is >> y;
22         x = y;
23         return is;
24     }
25     friend std::ostream& operator<<(std::ostream& os, Mint x) { return os << x.v; }
26
27     // comparision //
28     friend constexpr bool operator==(const Mint& lhs, const Mint& rhs) { return lhs.v == rhs.v; }
29     friend constexpr bool operator!=(const Mint& lhs, const Mint& rhs) { return lhs.v != rhs.v; }
30     friend constexpr bool operator<(const Mint& lhs, const Mint& rhs) { return lhs.v < rhs.v; }
31     friend constexpr bool operator<=(const Mint& lhs, const Mint& rhs) { return lhs.v <= rhs.v; }
32     friend constexpr bool operator>(const Mint& lhs, const Mint& rhs) { return lhs.v > rhs.v; }
33     friend constexpr bool operator>=(const Mint& lhs, const Mint& rhs) { return lhs.v >= rhs.v; }
34
35     // arithmetic //
36     template <typet>
37     friend constexpr Mint power(Mint a, T n) {
38         Mint ans = 1;
39         while (n) {
40             if (n & 1) ans *= a;
41             a *= a;
42             n >>= 1;
43         }
44         return ans;
45     }
46     friend constexpr Mint inv(const Mint& rhs) { return power(rhs, P - 2); }
47     friend constexpr Mint operator+(const Mint& lhs, const Mint& rhs) {
48         return lhs.val() + rhs.val() < P ? lhs.val() + rhs.val() : lhs.val() - P + rhs.val();
49     }
50     friend constexpr Mint operator-(const Mint& lhs, const Mint& rhs) {
51         return lhs.val() < rhs.val() ? lhs.val() + P - rhs.val() : lhs.val() - rhs.val();
52     }
53     friend constexpr Mint operator*(const Mint& lhs, const Mint& rhs) {
54         return static_cast<LL>(lhs.val()) * rhs.val() % P;
55     }
56     friend constexpr Mint operator/(const Mint& lhs, const Mint& rhs) { return lhs * inv(rhs); }
57     Mint operator+() const { return *this; }
58     Mint operator-() const { return Mint() - *this; }
59     constexpr Mint& operator++() {
60         v++;
61         if (v == P) v = 0;
62         return *this;
63     }
64     constexpr Mint& operator--() {
65         if (v == 0) v = P;
66         v--;
67         return *this;
68     }
69     constexpr Mint& operator++(int) {
70         Mint ans = *this;
71         ++*this;
72         return ans;
73     }
74     constexpr Mint& operator--(int) {
75         Mint ans = *this;
76         --*this;
77         return ans;
78     }
79     constexpr Mint& operator+=(const Mint& rhs) {

```

```

80     v = v + rhs;
81     return *this;
82 }
83 constexpr Mint& operator--(const Mint& rhs) {
84     v = v - rhs;
85     return *this;
86 }
87 constexpr Mint& operator*=(const Mint& rhs) {
88     v = v * rhs;
89     return *this;
90 }
91 constexpr Mint& operator/=(const Mint& rhs) {
92     v = v / rhs;
93     return *this;
94 }
95 };
96 using Z = Mint<998244353>;

```

5.2 Eculid

欧几里得算法

```
1 std::gcd(a, b)
```

扩展欧几里得算法

```

1  /* exgcd */
2  auto exgcd = [&](LL a, LL b, LL& x, LL& y) {
3      LL x1 = 1, x2 = 0, x3 = 0, x4 = 1;
4      while (b != 0) {
5          LL c = a / b;
6          std::tie(x1, x2, x3, x4, a, b) =
7              std::make_tuple(x3, x4, x1 - x3 * c, x2 - x4 * c, b, a - b * c);
8      }
9      x = x1, y = x2;
10 };
11 auto exgcd = [&](auto&& self, LL a, LL b, LL& x, LL& y) {
12     if (!b) {
13         x = 1, y = 0;
14         return a;
15     }
16     LL d = self(self, b, a % b, y, x);
17     y -= a / b * x;
18     return d;
19 };

```

```

1  auto exgcd = [&](auto&& self, LL a, LL b, LL& x, LL& y) {
2      if (!b) {
3          x = 1, y = 0;
4          return;
5      }
6      self(self, b, a % b, y, x);
7      y -= a / b * x;
8  };

```

```

1  auto exgcd = [&](auto&& self, LL a, LL b, LL& x, LL& y) {
2      if (!b) {
3          x = 1, y = 0;
4          return a;
5      }
6      LL d = self(self, b, a % b, y, x);
7      y -= a / b * x;
8      return d;
9  };

```

类欧几里得算法

一般形式: 求 $f(a, b, c, n) = \sum_{i=0}^n \lfloor \frac{ai+b}{c} \rfloor$

$$f(a, b, c, n) = nm - f(c, c - b - 1, a, m - 1)$$

```

1 LL f(LL a, LL b, LL c, LL n) {
2     if (a == 0) return ((b / c) * (n + 1));
3     if (a >= c || b >= c)
4         return f(a % c, b % c, c, n) + (a / c) * n * (n + 1) / 2 + (b / c) * (n + 1);
5     LL m = (a * n + b) / c;
6     LL v = f(c, c - b - 1, a, m - 1);
7     return n * m - v;
8 }

```

更进一步, 求: $g(a, b, c, n) = \sum_{i=0}^n i \lfloor \frac{ai+b}{c} \rfloor$ 以及 $h(a, b, c, n) = \sum_{i=0}^n \lfloor \frac{ai+b}{c} \rfloor^2$

$$g(a, b, c, n) = \lfloor \frac{mn(n+1) - f(c, c-b-1, a, m-1) - h(c, c-b-1, a, m-1)}{2} \rfloor$$

$$h(a, b, c, n) = nm(m+1) - 2f(c, c-b-1, a, m-1) - 2g(c, c-b-1, a, m-1) - f(a, b, c, n)$$

```

1 const int inv2 = 499122177;
2 const int inv6 = 166374059;
3
4 LL f(LL a, LL b, LL c, LL n);
5 LL g(LL a, LL b, LL c, LL n);
6 LL h(LL a, LL b, LL c, LL n);
7
8 struct data {
9     LL f, g, h;
10 };
11
12 data calc(LL a, LL b, LL c, LL n) {
13     LL ac = a / c, bc = b / c, m = (a * n + b) / c, n1 = n + 1, n21 = n * 2 + 1;
14     data d;
15     if (a == 0) {
16         d.f = bc * n1 % mod;
17         d.g = bc * n % mod * n1 % mod * inv2 % mod;
18         d.h = bc * bc % mod * n1 % mod;
19         return d;
20     }
21     if (a >= c || b >= c) {
22         d.f = n * n1 % mod * inv2 % mod * ac % mod + bc * n1 % mod;
23         d.g =
24             ac * n % mod * n1 % mod * n21 % mod * inv6 % mod + bc * n % mod * n1 % mod * inv2 % mod;
25         d.h = ac * ac % mod * n % mod * n1 % mod * n21 % mod * inv6 % mod +
26             bc * bc % mod * n1 % mod + ac * bc % mod * n % mod * n1 % mod;
27         d.f %= mod, d.g %= mod, d.h %= mod;
28         data e = calc(a % c, b % c, c, n);
29         d.h += e.h + 2 * bc % mod * e.f % mod + 2 * ac % mod * e.g % mod;
30         d.g += e.g, d.f += e.f;
31         d.f %= mod, d.g %= mod, d.h %= mod;
32         return d;
33     }
34     data e = calc(c, c - b - 1, a, m - 1);
35     d.f = n * m % mod - e.f, d.f = (d.f % mod + mod) % mod;
36     d.g = m * n % mod * n1 % mod - e.h - e.f, d.g = (d.g * inv2 % mod + mod) % mod;
37     d.h = n * m % mod * (m + 1) % mod - 2 * e.g - 2 * e.f - d.f;
38     d.h = (d.h % mod + mod) % mod;
39     return d;
40 }

```

5.3 inverse

线性递推

$$a^{-1} \equiv -\lfloor \frac{p}{a} \rfloor \times (p \% a)^{-1}$$

```

1 /* inverse */
2 vi inv(n + 1);
3 auto sieve_inv = [&](int n) {
4     inv[1] = 1;
5     for (int i = 2; i <= n; i++) {
6         inv[i] = 1ll * (p - p / i) * inv[p % i] % p;
7     }
8 };

```

求 n 个数的逆元

```

1  /* inverse */
2  auto inverse = [&](const vi& a) {
3      int n = a.size();
4      vi b(n), f(n), ivf(n);
5      f[0] = a[0];
6      for (int i = 1; i < n; i++) {
7          f[i] = 1ll * f[i - 1] * a[i] % p;
8      }
9      ivf.back() = quick_power(f.back(), p - 2, p);
10     for (int i = n - 1; i; i--) {
11         ivf[i - 1] = 1ll * ivf[i] * a[i] % p;
12     }
13     b[0] = ivf[0];
14     for (int i = 1; i < n; i++) {
15         b[i] = 1ll * ivf[i] * f[i - 1] % p;
16     }
17     return b;
18 };

```

5.4 sieve

素数

```

1  vi prime, is_prime(n + 1, 1);
2  auto Euler_sieve = [&](int n){
3      for (int i = 2; i <= n; i++) {
4          if (is_prime[i]) prime.push_back(i);
5          for (auto p : prime) {
6              if (i * p > n) break;
7              is_prime[i * p] = 0;
8              if (i % p == 0) break;
9          }
10     }
11 };

```

欧拉函数

```

1  vi phi(n + 1), prime;
2  vi is_prime(n + 1, 1);
3  auto get_phi = [&](int n) {
4      int cnt = 0;
5      phi[1] = 1;
6      for (int i = 2; i <= n; i++) {
7          if (is_prime[i]) {
8              prime.push_back(i);
9              phi[i] = i - 1;
10         }
11         for (auto p : prime) {
12             if (i * p > n) break;
13             is_prime[i * p] = 0;
14             if (i % p) {
15                 phi[i * p] = phi[i] * phi[p];
16             } else {
17                 phi[i * p] = phi[i] * p;
18                 break;
19             }
20         }
21     }
22 };

```

约数和

```

1  vi g(n + 1), d(n + 1), prime;
2  vi is_prime(n + 1, 1);
3  auto get_d = [&](int n) {
4      int tot = 0;
5      g[1] = d[1] = 1;
6      for (int i = 2; i <= n; i++) {
7          if (is_prime[i]) {

```

```

8         prime.push_back(i);
9         d[i] = g[i] = i + 1;
10    }
11    for (auto p : prime) {
12        if (i * p > n) break;
13        is_prime[i * p] = 0;
14        if (i % p == 0) {
15            g[i * p] = g[i] * p + 1;
16            d[i * p] = d[i] / g[i] * g[i * p];
17            break;
18        } else {
19            d[i * p] = d[i] * d[p];
20            g[i * p] = 1 + p;
21        }
22    }
23 }
24 };

```

莫比乌斯函数

```

1  vi mu(n + 1), prime;
2  vi is_prime(n + 1, 1);
3  auto get_mu = [&](int n) {
4      mu[1] = 1;
5      for (int i = 2; i <= n; i++) {
6          if (is_prime[i]) {
7              prime.push_back(i);
8              mu[i] = -1;
9          }
10         for (auto p : prime) {
11             if (i * p > n) break;
12             is_prime[i * p] = 0;
13             if (i % p == 0) {
14                 mu[i * p] = 0;
15                 break;
16             }
17             mu[i * p] = -mu[i];
18         }
19     }
20 };

```

杜教筛

```

1  const int N = 1e7;
2  vi mu(N + 1), phi(N + 1), prime;
3  vl sum_phi(N + 1), sum_mu(N + 1);
4  vi is_prime(N + 1, 1);
5  std::map<LL, LL> mp_mu;
6
7  /* 计算 1 ~ 10^7 的 mu */
8  auto get_mu = [&](int n) {
9      phi[1] = mu[1] = 1;
10     for (int i = 2; i <= n; i++) {
11         if (is_prime[i]) {
12             prime.push_back(i);
13             phi[i] = i - 1;
14             mu[i] = -1;
15         }
16         for (auto p : prime) {
17             if (i * p > n) break;
18             is_prime[i * p] = 0;
19             if (i % p == 0) {
20                 phi[i * p] = phi[i] * p;
21                 mu[i * p] = 0;
22                 break;
23             }
24             phi[i * p] = phi[i] * phi[p];
25             mu[i * p] = -mu[i];
26         }
27     }
28 };
29 get_mu(N);
30 for (int i = 1; i <= N; i++) {
31     sum_phi[i] = sum_phi[i - 1] + phi[i];
32     sum_mu[i] = sum_mu[i - 1] + mu[i];
33 }
34
35 /* 杜教筛：求 mu 的前缀和 */

```

```

36 std::function<LL(LL)> S_mu = [&](LL x) -> LL {
37     if (x <= N) return sum_mu[x];
38     auto it = mp_mu.find(x);
39     if (it != mp_mu.end()) return mp_mu[x];
40     LL ans = 1;
41     for (LL i = 2, j; i <= x; i = j + 1) {
42         j = x / (x / i);
43         ans -= S_mu(x / i) * (j - i + 1);
44     }
45     return mp_mu[x] = ans;
46 };
47
48 /* 杜教筛: 求 phi 的前缀和 */
49 auto S_phi = [&](LL x) -> LL {
50     if (x <= N) return sum_phi[x];
51     LL ans = 0;
52     for (LL i = 1, j; i <= x; i = j + 1) {
53         j = x / (x / i);
54         ans += 1ll * (S_mu(j) - S_mu(i - 1)) * (x / i) * (x / i);
55     }
56     return (ans - 1) / 2 + 1;
57 };

```

5.5 powerful number

目标: 求积性函数 $f(n)$ 的前缀和. 做法如下:

1. 构造积性函数 $g(n)$, 满足其易求前缀和且素数处函数值等于 f 的函数值.
2. 构造 $h = f/g$, 即 $f = h * g$ (狄利克雷卷积), 容易知道 $h(1) = 1$. 容易计算出 h 在非 powerful number 处函数值均为 0.
3. 根据

$$\begin{aligned}
 F(n) &= \sum_{i=1}^n f(i) \\
 &= \sum_{i=1}^n \sum_{d|i} g(i)h(i/d) \\
 &= \sum_{d=1}^n \sum_{i=1}^{\lfloor \frac{n}{d} \rfloor} h(d)g(i) \\
 &= \sum_{d=1}^n h(d)G\left(\left\lfloor \frac{n}{d} \right\rfloor\right) \\
 &= \sum_{d=1,2,\dots,n, d \text{ is powerful number}} h(d)G\left(\left\lfloor \frac{n}{d} \right\rfloor\right)
 \end{aligned}$$

发现只需要计算 h 在 powerful number 处的函数值即可, 可以边搜索边计算.

给定 $f(p^k) = p^k(p^k - 1)$ 为积性函数, 计算其前缀和.

```

1 auto powerfulNumber = [&](LL n) {
2     /* 1. construct g, and compute G */
3     /* int m = sqrt(n); // maybe TLE // */
4     int m = 2e6;
5     std::vector<Z> Gs(m + 1);
6     vi prime, is_prime(m + 1, 1);
7     Gs[1] = 1;
8     for (int i = 2; i <= m; i++) {
9         if (is_prime[i]) {
10             prime.push_back(i);
11             Gs[i] = i - 1;
12         }
13         for (auto p : prime) {
14             if (i * p > m) break;
15             is_prime[i * p] = 0;
16             if (i % p) {
17                 Gs[i * p] = Gs[i] * Gs[p];
18             } else {

```

```

19         Gs[i * p] = Gs[i] * p;
20         break;
21     }
22 }
23 }
24 for (int i = 2; i <= m; i++) {
25     Gs[i] = Gs[i - 1] + Z(i) * Gs[i];
26 }
27 std::map<LL, Z> mp;
28 auto G = [&](auto&& self, LL n) {
29     if (n <= m) return Gs[n];
30     if (mp.find(n) != mp.end()) return mp[n];
31     Z ans = Z(n) * Z(n + 1) * Z(n * 2 + 1) * inv6;
32     for (LL l = 2, r, k; l <= n; l = r + 1) {
33         k = n / l, r = n / (n / l);
34         ans -= (Z(r) * Z(r + 1) - Z(l - 1) * Z(l)) * inv2 * self(self, k);
35     }
36     return mp[n] = ans;
37 };
38 /* 2. compute h(p^c) */
39 vvl ps(prime.size());
40 std::vector<std::vector<Z>> hs(prime.size());
41 int len = 0;
42 for (int i = 0; i < prime.size(); i++) {
43     LL p = prime[i], now = p * p, c = 2;
44     ps[i] = {1, p}, hs[i] = {1, 0};
45     while (now <= n) {
46         ps[i].push_back(now);
47         Z ans = Z(ps[i][c]) * (Z(ps[i][c]) - 1);
48         for (int j = 1; j <= c; j++) {
49             ans -= Z(ps[i][j]) * Z(ps[i][j - 1]) * Z(p - 1) * hs[i][c - j];
50         }
51         hs[i].push_back(ans);
52         now *= p, c += 1;
53     }
54     len += ps[i].size();
55 }
56 debug(len);
57 /* 3. search powerful number */
58 Z ans = 0;
59 auto dfs = [&](auto&& self, int id, LL now, Z hd) -> void {
60     ans += hd * G(G, n / now);
61     for (int i = id; i < prime.size(); i++) {
62         int p = prime[i], c = 2;
63         if (now > n / p / p) break;
64         for (LL x = now * p * p; x <= n; x *= p, c++) {
65             if (hs[i][c]) self(self, i + 1, x, hd * hs[i][c]);
66         }
67     }
68 };
69 dfs(dfs, 0, 1, 1);
70 return ans;
71 };

```

5.6 block

分块的逻辑

下取整 $\lfloor \frac{n}{g} \rfloor = k$ 的分块 ($g \leq n$)

```

1 for(int l = 1, r, k; l <= n; l = r + 1){
2     k = n / l;
3     r = n / (n / l);
4     debug(l, r, k);
5 }

```

$k = \lfloor \frac{n}{g} \rfloor$ 从大到小遍历 $\lfloor \frac{n}{g} \rfloor$ 的所有取值, $[l, r]$ 对应的是 g 取值的区间.

```

1 n = 11
2 [l, r, k] : 1 1 11
3 [l, r, k] : 2 2 5
4 [l, r, k] : 3 3 3
5 [l, r, k] : 4 5 2
6 [l, r, k] : 6 11 1

```

上取整 $\lceil \frac{n}{g} \rceil = k$ 的分块 ($g < n$)

```

1 for(int l = 1, r, k; l < n; l = r + 1){
2     k = (n + l - 1) / l;
3     r = (n + k - 2) / (k - 1) - 1;
4     debug(l, r, k);
5 }

```

$k = \lceil \frac{n}{g} \rceil$ 从大到小遍历 $\lceil \frac{n}{g} \rceil$ 的所有取值, $[l, r]$ 对应的是 g 取值的区间.

```

1 n = 11
2 [l, r, k] : 1 1 11
3 [l, r, k] : 2 2 6
4 [l, r, k] : 3 3 4
5 [l, r, k] : 4 5 3
6 [l, r, k] : 6 10 2

```

一般形式

计算 $\sum_{i=1}^n f(i) \lfloor \frac{n}{i} \rfloor$, 设 $s(i)$ 为 $f(i)$ 的前缀和。

```

1 for (int l = 1, r; l <= n; l = r + 1) {
2     r = n / (n / l);
3     ans += (s[r] - s[l - 1]) * (n / l);
4 }

```

$$\sum_{i=1}^n f(i) \lfloor \frac{a}{i} \rfloor \lfloor \frac{b}{i} \rfloor$$

```

1 for (int l = 1, r, r1, r2; l <= n; l = r + 1) {
2     if (a / l) {
3         r1 = a / (a / l);
4     } else {
5         r1 = n;
6     }
7     if (b / l) {
8         r2 = b / (b / l);
9     } else {
10        r2 = n;
11    }
12    r = min(min(r1, r2), n);
13    ans += (s[r] - s[l - 1]) * (a / l) * (b / l);
14 }

```

5.7 CRT & exCRT

求解

$$\begin{cases} N \equiv a_1 \pmod{m_1} \\ N \equiv a_2 \pmod{m_2} \\ \dots \\ N \equiv a_n \pmod{m_n} \end{cases}$$

$$\text{有 } N \equiv \sum_{i=1}^k a_i \times \text{inv}\left(\frac{M}{m_i}, m_i\right) \times \left(\frac{M}{m_i}\right) \pmod{M}$$

```

1 /* CRT */
2 auto crt = [&](int n, const vi& a, const vi& m) -> LL{
3     LL ans = 0, M = 1;
4     for(int i = 1; i <= n; i++) M *= m[i];
5     for(int i = 1; i <= n; i++){
6         ans = (ans + a[i] * inv(M / m[i], m[i]) * (M / m[i])) % M;
7     }
8     return (ans % M + M) % M;
9 };

```

扩展中国剩余定理

```

1 /* exCRT */
2 auto excrt = [&](int n, const vi& a, const vi& m) -> LL{
3     LL A = a[1], M = m[1];

```

```

4   for (int i = 2; i <= n; i++) {
5       LL x, y, d = std::gcd(M, m[i]);
6       exgcd(M, m[i], x, y);
7       LL mod = M / d * m[i];
8       x = x * (a[i] - A) / d % (m[i] / d);
9       A = ((M * x + A) % mod + mod) % mod;
10      M = mod;
11  }
12  return A;
13 };

```

5.8 BSGS & exBSGS

求解满足 $a^x \equiv b \pmod p$ 的 x

```

1  /* BSGS */
2  /* return value = -1e18 means no solution */
3  auto BSGS = [&](LL a, LL b, LL p) {
4      if (1 % p == b % p) return 0ll;
5      LL k = std::sqrt(p) + 1;
6      std::unordered_map<LL, LL> hash;
7      for (LL i = 0, j = b % p; i < k; i++) {
8          hash[j] = i;
9          j = j * a % p;
10     }
11     LL ak = 1;
12     for (int i = 1; i <= k; i++) ak = ak * a % p;
13     for (int i = 1, j = ak; i <= k; i++) {
14         if (hash.count(j)) return 1ll * i * k - hash[j];
15         j = 1ll * j * ak % p;
16     }
17     return -INF;
18 };

```

$(a, p) \neq 1$ 的情形

```

1  /* exBSGS */
2  /* return value < 0 means no solution */
3  auto exBSGS = [&](auto&& self, LL a, LL b, LL p) {
4      b = (b % p + p) % p;
5      if (1ll % p == b % p) return 0ll;
6      LL x, y, d = std::gcd(a, p);
7      exgcd(exgcd, a, p, x, y);
8      if (d > 1) {
9          if (b % d != 0) return -INF;
10         exgcd(exgcd, a / d, p / d, x, y);
11         return self(self, a, b / d * x % (p / d), p / d) + 1;
12     }
13     return BSGS(a, b, p);
14 };

```

5.9 Miller Rabin

```

1  /* Miller Rabin */
2  vl vv = {2, 325, 9375, 28178, 450775, 9780504, 1795265022};
3  auto quick_power = [&](LL a, LL n, LL mod) {
4      LL ans = 1;
5      while (n) {
6          if (n & 1) ans = (i128) ans * a % mod;
7          a = (i128) a * a % mod;
8          n >>= 1;
9      }
10     return ans;
11 };
12
13 auto millerRabin = [&](LL n) {
14     if (n < 2 || n % 6 % 4 != 1) return (n | 1) == 3;
15     int s = __builtin_ctzll(n - 1);
16     LL d = n >> s;
17     for (auto a : vv) {
18         LL p = quick_power(a % n, d, n);
19         int i = s;
20         while (p != 1 and p != n - 1 and a % n and i--) p = (i128) p * p % n;
21         if (p != n - 1 and i != s) return false;
22     }
23     return true;

```

```
24 };
```

5.10 Pollard Rho

能在 $O(n^{\frac{1}{4}})$ 的时间复杂度随机出一个 n 的非平凡因数.

```
1  /* pollard rho */
2  auto pollard_rho = [&](LL x) -> LL{
3      LL s = 0, t = 0, val = 1;
4      LL c = rand() % (x - 1) + 1;
5      for(int goal = 1;; goal <= 1, s = t, val = 1){
6          for(int step = 1; step <= goal; step++){
7              t = ((i128) t * t + c) % x;
8              val = (i128) val * abs(t - s) % x;
9              if(step % 127 == 0){
10                 LL d = std::gcd(val, x);
11                 if(d > 1) return d;
12             }
13         }
14         LL d = std::gcd(val, x);
15         if(d > 1) return d;
16     }
17 };
```

利用 Miller Rabin 和 Pollard Rho 进行素因数分解

```
1  auto factorize = [&](LL a) -> vl{
2      vl ans, stk;
3      for (auto p : prime) {
4          if (p > 1000) break;
5          while (a % p == 0) {
6              ans.push_back(p);
7              a /= p;
8          }
9          if (a == 1) return ans;
10     }
11     stk.push_back(a);
12     while (!stk.empty()) {
13         LL b = stk.back();
14         stk.pop_back();
15         if (miller_rabin(b)) {
16             ans.push_back(b);
17             continue;
18         }
19         LL c = b;
20         while (c >= b) c = pollard_rho(b);
21         stk.push_back(c);
22         stk.push_back(b / c);
23     }
24     return ans;
25 };
```

5.11 quadratic residu

```
1  /* cipolla */
2  auto cipolla = [&](int x) {
3      std::srand(time(0));
4      auto check = [&](int x) -> bool { return pow(x, (mod - 1) / 2) == 1; };
5      if (!x) return 0;
6      if (!check(x)) return -1;
7      int a, b;
8      while (1) {
9          a = rand() % mod;
10         b = sub(mul(a, a), x);
11         if (!check(b)) break;
12     }
13     PII t = {a, 1};
14     PII ans = {1, 0};
15     auto mulp = [&](PII x, PII y) -> PII {
16         auto [x1, x2] = x;
17         auto [y1, y2] = y;
18         int c = add(mul(x1, y1), mul(x2, y2, b));
19         int d = add(mul(x1, y2), mul(x2, y1));
20         return {c, d};
21     };
```



```

22     for (int i = (mod + 1) / 2; i; i >>= 1) {
23         if (i & 1) ans = mulp(ans, t);
24         t = mulp(t, t);
25     }
26     return std::min(ans.ff, mod - ans.ff);
27 }

```

5.12 Lucas

卢卡斯定理

用于求大组合数，并且模数是一个不大的素数。

$$\binom{n}{m} \bmod p = \binom{\lfloor n/p \rfloor}{\lfloor m/p \rfloor} \cdot \binom{n \bmod p}{m \bmod p} \bmod p$$

$\binom{n \bmod p}{m \bmod p}$ 可以直接计算， $\binom{\lfloor n/p \rfloor}{\lfloor m/p \rfloor}$ 可以继续使用卢卡斯计算。

递归至 $m = 0$ 的时候，返回 1。

p 不太大，一般在 10^5 左右。

```

1  auto C = [&](LL n, LL m, LL p) -> LL {
2      if (n < m) return 0;
3      if (m == 0) return 1;
4      return fac[n] * inv_fac[m] % p * inv_fac[n - m] % p;
5  };
6  /* lucas */
7  auto lucas = [&](auto&& self, LL n, LL m, LL p) -> LL {
8      if (n < m) return 0;
9      if (m == 0) return 1;
10     return C(n % p, m % p, p) * self(self, n / p, m / p, p) % p;
11 }

```

素数在组合数中的次数

Legengre 给出一种 $n!$ 中素数 p 的幂次的计算方式为：

$$\sum_{1 \leq j} \left\lfloor \frac{n}{p^j} \right\rfloor.$$

另一种计算方式利用 p 进制下各位数字和：

$$v_p(n!) = \frac{n - S_p(n)}{p - 1}.$$

则有

$$v_p(C_m^n) = \frac{S_p(n) + S_p(m - n) - S_p(m)}{p - 1}.$$

扩展卢卡斯定理

计算

$$\binom{n}{m} \bmod p,$$

p 可能为合数。

第一部分：CRT.

原问题变成求

$$\left\{ \begin{array}{l} \binom{n}{m} \equiv a_1 \pmod{p_1^{\alpha_1}} \\ \binom{n}{m} \equiv a_2 \pmod{p_2^{\alpha_2}} \\ \dots \\ \binom{n}{m} \equiv a_k \pmod{p_k^{\alpha_k}} \end{array} \right.$$

在求出 a_i 之后就可以利用 CRT 求出答案.

第二部分: 移除分子分母中的素数

问题转换成求解

$$\binom{n}{m} \pmod{q^k}.$$

等价于

$$\frac{\frac{n!}{q^x}}{\frac{m!}{q^y} \frac{(n-m)!}{q^z}} q^{x-y-z} \pmod{q^k},$$

其中 x 表示 $n!$ 中 q 的次数, y, z 同理.

第三部分: 威尔逊定理的推论

问题转换为求

$$\frac{n!}{q^x} \pmod{q^k}.$$

可以利用威尔逊定理的推论.

```

1  /* exlucas */
2  auto exLucas = [&](LL n, LL m, LL p) {
3      auto inv = [&](LL a, LL p) {
4          LL x, y;
5          exgcd(a, p, x, y);
6          return (x % p + p) % p;
7      };
8
9      auto func = [&](auto&& self, LL n, LL pi, LL pk) {
10         if (!n) return 1ll;
11         LL ans = 1;
12         for (LL i = 2; i <= pk; i++) {
13             if (i % pi) ans = ans * i % p;
14         }
15         ans = quick_power(ans, n / pk, pk);
16         for (LL i = 2; i <= n % pk; i++) {
17             if (i % pi) ans = ans * i % pk;
18         }
19         ans = ans * self(self, n / pi, pi, pk) % pk;
20         return ans;
21     };
22
23     auto multiLucas = [&](LL n, LL m, LL pi, LL pk) {
24         LL cnt = 0;
25         for (LL i = n; i; i /= pi) cnt += i / pi;
26         for (LL i = m; i; i /= pi) cnt -= i / pi;
27         for (LL i = n - m; i; i /= pi) cnt -= i / pi;
28         LL ans = quick_power(pi, cnt, pk) * func(func, n, pi, pk) % pk;
29         ans = ans * inv(func(func, m, pi, pk), pk) % pk;
30         ans = ans * inv(func(func, n - m, pi, pk), pk) % pk;
31         return ans;
32     };
33
34     auto crt = [&](const vl& a, const vl& m, int k) {
35         LL ans = 0;
36
37         for (int i = 0; i < k; i++) {
38             ans = (ans + a[i] * inv(p / m[i], m[i]) * (p / m[i])) % p;
39         }

```

```

40     return (ans % p + p) % p;
41 };
42
43 vl a, prime;
44 LL pp = p;
45 for (int i = 2; i * i <= pp; i++) {
46     if (pp % i) continue;
47     prime.push_back(1);
48     while (pp % i == 0) {
49         prime.back() *= i;
50         pp /= i;
51     }
52     a.push_back(multiLucas(n, m, i, prime.back()));
53 }
54 if (pp > 1) {
55     prime.push_back(pp);
56     a.push_back(multiLucas(n, m, pp, pp));
57 }
58 return crt(a, prime, a.size());
59 };

```

5.13 Wilson

简单结论

对于素数 p 有

$$(p-1)! \equiv -1 \pmod{p}.$$

推论

令 $(n!)_p$ 表示不大于 n 且不被 p 整除的正整数的乘积.

特殊情形: n 为素数 p 时即为上述结论.

一般结论: 对素数 p 和正整数 q 有

$$((p^q)!)_p \equiv \pm 1 \pmod{p^q}.$$

详细定义:

$$((p^q)!)_p = \begin{cases} 1 & \text{if } p = 2 \text{ and } q \geq 3, \\ -1 & \text{other wise.} \end{cases}$$

5.14 LTE

将素数 p 在整数 n 中的个数记为 $v_p(n)$.

$$(n, p) = 1$$

对所有素数 p 和满足 $(n, p) = 1$ 的整数 n , 有

1. 若 $p \mid x - y$, 则有

$$v_p(x^n - y^n) = v_p(x - y).$$

2. 若 $p \mid x - y$, 则对奇数 n 有

$$v_p(x^n + y^n) = v_p(x + y).$$

p 是奇素数

对所有奇素数 p 有

1. 若 $p \mid x - y$, 则有

$$v_p(x^n - y^n) = v_p(x - y) + v_p(n).$$

2. 若 $p \mid x - y$, 则对奇数 n 有

$$v_p(x^n + y^n) = v_p(x + y) + v_p(n).$$

$p = 2$

对 $p = 2$ 且 $p \mid x - y$ 有

1. 对奇数 n 有

$$v_2(x^n - y^n) = v_2(x - y).$$

2. 对偶数 n 有

$$v_2(x^n - y^n) = v_2(x - y) + v_2(x + y) + v_2(n) - 1.$$

除此之外, 对上述 x, y, n , 若 $4 \mid x - y$, 有

1. $v_2(x + y) = 1$.
2. $v_2(x^n - y^n) = v_2(x - y) + v_2(n)$.

5.15 Mobius inversion

莫比乌斯函数

$$\mu(n) = \begin{cases} 1 & n = 1, \\ 0 & n \text{ 含有平方因子}, \\ (-1)^k & k \text{ 为 } n \text{ 的本质不同素因子个数}. \end{cases}$$

性质

$$\sum_{d \mid n} \mu(d) = \begin{cases} 1 & n = 1 \\ 0 & n \neq 1 \end{cases}.$$

$$\varphi(n) = \sum_{d \mid n} d \cdot \mu\left(\frac{n}{d}\right).$$

反演结论

$$[gcd(i, j) = 1] = \sum_{d \mid gcd(i, j)} \mu(d).$$

$O(n \log n)$ 求莫比乌斯函数

```

1 mu[1] = 1;
2 for (int i = 1; i <= n; i++){
3     for (int j = i + i; j <= n; j += i){

```

```

4      mu[j] -= mu[i];
5      }
6  }
```

莫比乌斯变换

设 $f(n), F(n)$.

1. $F(n) = \sum_{d|n} f(d)$, 则 $f(n) = \sum_{d|n} \mu(d) F\left(\frac{n}{d}\right)$.
2. $F(n) = \sum_{n|d} f(d)$, 则 $f(n) = \sum_{n|d} \mu\left(\frac{d}{n}\right) F(d)$.

6 math - polynomial

6.1 FTT

FFT 与拆系数 FFT

```

1  const int sz = 1 << 23;
2  int rev[sz];
3  int rev_n;
4  void set_rev(int n) {
5      if (n == rev_n) return;
6      for (int i = 0; i < n; i++) rev[i] = (rev[i / 2] | (i & 1) * n) / 2;
7      rev_n = n;
8  }
9  template void butterfly(T* a, int n) {
10     set_rev(n);
11     for (int i = 0; i < n; i++) {
12         if (i < rev[i]) std::swap(a[i], a[rev[i]]);
13     }
14 }
15
16 namespace Comp {
17
18 long double pi = 3.141592653589793238;
19
20 template struct complex {
21     T x, y;
22     complex(T x = 0, T y = 0) : x(x), y(y) {}
23     complex operator+(const complex& b) const { return complex(x + b.x, y + b.y); }
24
25     complex operator-(const complex& b) const { return complex(x - b.x, y - b.y); }
26
27     complex operator*(const complex& b) const {
28         return complex<T>(x * b.x - y * b.y, x * b.y + y * b.x);
29     }
30     complex operator~() const { return complex(x, -y); }
31     static complex unit(long double rad) { return complex(std::cos(rad), std::sin(rad)); }
32 };
33
34 } // namespace Comp
35
36 struct fft_t {
37     typedef Comp::complex<double> complex;
38     complex wn[sz];
39
40     fft_t() {
41         for (int i = 0; i < sz / 2; i++) {
42             wn[sz / 2 + i] = complex::unit(2 * Comp::pi * i / sz);
43         }
44         for (int i = sz / 2 - 1; i; i--) wn[i] = wn[i * 2];
45     }
46
47     void operator()(complex* a, int n, int type) {
48         if (type == -1) std::reverse(a + 1, a + n);
49         butterfly(a, n);
50         for (int i = 1; i < n; i *= 2) {
51             const complex* w = wn + i;
52             for (complex* b = a, t; b != a + n; b += i + 1) {
53                 t = b[i];
54                 b[i] = *b - t;
55                 *b = *b + t;
56                 for (int j = 1; j < i; j++) {
57                     t = (++b)[i] * w[j];
58                     b[i] = *b - t;
59                     *b = *b + t;
60                 }
61             }
62         }
63         if (type == 1) return;
64         for (int i = 0; i < n * 2; i++) ((double*) a)[i] /= n;
65     }
66 } FFT;
67
68 typedef decltype(FFT)::complex complex;
69
70 vi fft(const vi& f, const vi& g) {
71     static complex ff[sz];
72     int n = f.size(), m = g.size();
73     vi h(n + m - 1);
74     if (std::min(n, m) <= 50) {
75         for (int i = 0; i < n; i++) {

```

```

76         for (int j = 0; j < m; ++j) {
77             h[i + j] += f[i] * g[j];
78         }
79     }
80     return h;
81 }
82 int c = 1;
83 while (c + 1 < n + m) c *= 2;
84 std::memset(ff, 0, sizeof(decltype(*(ff))) * (c));
85 for (int i = 0; i < n; i++) ff[i].x = f[i];
86 for (int i = 0; i < m; i++) ff[i].y = g[i];
87 FFT(ff, c, 1);
88 for (int i = 0; i < c; i++) ff[i] = ff[i] * ff[i];
89 FFT(ff, c, -1);
90 for (int i = 0; i + 1 < n + m; i++) h[i] = std::llround(ff[i].y / 2);
91 return h;
92 }
93
94 vi mtt(const vi& f, const vi& g) {
95     static complex ff[3][sz], gg[2][sz];
96     static int s[3] = {1, 31623, 31623 * 31623};
97     int n = f.size(), m = g.size();
98     vi h(n + m - 1);
99     if (std::min(n, m) <= 50) {
100         for (int i = 0; i < n; ++i) {
101             for (int j = 0; j < m; ++j) {
102                 Add(h[i + j], mul(f[i], g[j]));
103             }
104         }
105         return h;
106     }
107     int c = 1;
108     while (c + 1 < n + m) c *= 2;
109     for (int i = 0; i < 2; ++i) {
110         std::memset(ff[i], 0, sizeof(decltype(*(ff[i]))) * (c));
111         std::memset(gg[i], 0, sizeof(decltype(*(ff[i]))) * (c));
112         for (int j = 0; j < n; ++j) ff[i][j].x = f[j] / s[i] % s[1];
113         for (int j = 0; j < m; ++j) gg[i][j].x = g[j] / s[i] % s[1];
114         FFT(ff[i], c, 1);
115         FFT(gg[i], c, 1);
116     }
117     for (int i = 0; i < c; ++i) {
118         ff[2][i] = ff[1][i] * gg[1][i];
119         ff[1][i] = ff[1][i] * gg[0][i];
120         gg[1][i] = ff[0][i] * gg[1][i];
121         ff[0][i] = ff[0][i] * gg[0][i];
122     }
123     for (int i = 0; i < 3; ++i) {
124         FFT(ff[i], c, -1);
125         for (int j = 0; j + 1 < n + m; ++j) {
126             Add(h[j], mul(std::llround(ff[i][j].x) % mod, s[i]));
127         }
128     }
129     FFT(gg[1], c, -1);
130     for (int i = 0; i + 1 < n + m; ++i) {
131         Add(h[i], mul(std::llround(gg[1][i].x) % mod, s[1]));
132     }
133     return h;
134 }

```

6.2 FWT

各种分治过程: **and**:

$$\begin{aligned} \text{FWT}[A] &= \text{merge}(\text{FWT}[A_0] + \text{FWT}[A_1], \text{FWT}[A_1]), \\ \text{UFWT}[A'] &= \text{merge}(\text{UFWT}[A'_0] - \text{UFWT}[A'_1], \text{UFWT}[A'_1]). \end{aligned}$$

or:

$$\begin{aligned} \text{FWT}[A] &= \text{merge}(\text{FWT}[A_0], \text{FWT}[A_0] + \text{FWT}[A_1]), \\ \text{UFWT}[A'] &= \text{merge}(\text{UFWT}[A'_0], -\text{UFWT}[A'_0] + \text{UFWT}[A'_1]). \end{aligned}$$

xor:

$$\text{FWT}[A] = \text{merge}(\text{FWT}[A_0] + \text{FWT}[A_1], \text{FWT}[A_0] - \text{FWT}[A_1]),$$

$$\text{UFWT}[A'] = \text{merge} \left(\frac{\text{UFWT}[A'_0] + \text{UFWT}[A'_1]}{2}, \frac{\text{UFWT}[A'_0] - \text{UFWT}[A'_1]}{2} \right).$$

```

1  /* FWT */
2  auto FWT_and = [&](vi v, int type) -> vi {
3      int n = v.size();
4      for (int mid = 1; mid < n; mid <= 1) {
5          for (int block = mid << 1, j = 0; j < n; j += block) {
6              for (int i = j; i < j + mid; i++) {
7                  LL x = v[i], y = v[i + mid];
8                  if (type == 1) {
9                      v[i] = add(x, y);
10                 } else {
11                     v[i] = sub(x, y);
12                 }
13             }
14         }
15     }
16     return v;
17 };
18
19 auto FWT_or = [&](vi v, int type) -> vi {
20     int n = v.size();
21     for (int mid = 1; mid < n; mid <= 1) {
22         for (int block = mid << 1, j = 0; j < n; j += block) {
23             for (int i = j; i < j + mid; i++) {
24                 LL x = v[i], y = v[i + mid];
25                 if (type == 1) {
26                     v[i + mid] = add(x, y);
27                 } else {
28                     v[i + mid] = sub(y, x);
29                 }
30             }
31         }
32     }
33     return v;
34 };
35
36 auto FWT_xor = [&](vi v, int type) -> vi {
37     int n = v.size();
38     for (int mid = 1; mid < n; mid <= 1) {
39         for (int block = mid << 1, j = 0; j < n; j += block) {
40             for (int i = j; i < j + mid; i++) {
41                 LL x = v[i], y = v[i + mid];
42                 v[i] = add(x, y);
43                 v[i + mid] = sub(x, y);
44                 if (type == -1) {
45                     Mul(v[i], inv2);
46                     Mul(v[i + mid], inv2);
47                 }
48             }
49         }
50     }
51     return v;
52 };
53
54 a = FWT(a, 1), b = FWT(b, 1);
55 for (int i = 0; i < (1 << n); i++) {
56     c[i] = mul(a[i], b[i]);
57 }
58 c = FWT(c, -1);

```

```

1  /* FWT @ wrb */
2  void FMTor(int f[]) {
3      for (int i = 0; i < n; ++i)
4          for (int j = 0; j < m; ++j)
5              if (j >> i & 1) f[j] = (f[j] + f[j ^ 1 << i]) % P;
6  }
7  void FMToriv(int f[]) {
8      for (int i = 0; i < n; ++i)
9          for (int j = 0; j < m; ++j)
10             if (j >> i & 1) f[j] = (f[j] - f[j ^ 1 << i] + P) % P;
11 }
12 void FMTand(int f[]) {
13     for (int i = 0; i < n; ++i)
14         for (int j = 0; j < m; ++j)
15             if (~j >> i & 1) f[j] = (f[j] + f[j ^ 1 << i]) % P;
16 }
17 void FMTandiv(int f[]) {
18     for (int i = 0; i < n; ++i)
19         for (int j = 0; j < m; ++j)
20             if (~j >> i & 1) f[j] = (f[j] - f[j ^ 1 << i] + P) % P;
21 }
22 void FWT(int f[]) {

```



```

23     for (int len = 1; len < m; len *= 2) {
24         for (int i = 0; i < m; i += len * 2) {
25             for (int j = i; j < i + len; ++j) {
26                 int x = f[j], y = f[j + len];
27                 f[j] = (x + y) % P;
28                 f[j + len] = (x - y + P) % P;
29             }
30         }
31     }
32 }
33 void FWTiv(int f[]) {
34     for (int len = 1; len < m; len *= 2) {
35         for (int i = 0; i < m; i += len * 2) {
36             for (int j = i; j < i + len; ++j) {
37                 int x = f[j], y = f[j + len];
38                 f[j] = 111 * (x + y) * iv2 % P;
39                 f[j + len] = 111 * (x - y + P) * iv2 % P;
40             }
41         }
42     }
43 }

```

6.3 class polynomial

```

1  class polynomial : public vi {
2      public:
3          polynomial() = default;
4          polynomial(const vi& v) : vi(v) {}
5          polynomial(vi&& v) : vi(std::move(v)) {}
6
7          int degree() { return size() - 1; }
8
9          void clearzero() {
10             while (size() && !back()) pop_back();
11         }
12 };
13
14
15 polynomial& operator+=(polynomial& a, const polynomial& b) {
16     a.resize(std::max(a.size(), b.size()), 0);
17     for (int i = 0; i < b.size(); i++) {
18         Add(a[i], b[i]);
19     }
20     a.clearzero();
21     return a;
22 }
23
24 polynomial operator+(const polynomial& a, const polynomial& b) {
25     polynomial ans = a;
26     return ans += b;
27 }
28
29 polynomial& operator-=(polynomial& a, const polynomial& b) {
30     a.resize(std::max(a.size(), b.size()), 0);
31     for (int i = 0; i < b.size(); i++) {
32         Sub(a[i], b[i]);
33     }
34     a.clearzero();
35     return a;
36 }
37
38 polynomial operator-(const polynomial& a, const polynomial& b) {
39     polynomial ans = a;
40     return ans -= b;
41 }
42
43 class ntt_t {
44     public:
45         static const int maxbit = 22;
46         static const int sz = 1 << maxbit;
47         static const int mod = 998244353;
48         static const int g = 3;
49
50         std::array<int, sz + 10> w;
51         std::array<int, maxbit + 10> len_inv;
52
53         ntt_t() {
54             int wn = pow(g, (mod - 1) >> maxbit);
55             w[0] = 1;
56             for (int i = 1; i <= sz; i++) {
57                 w[i] = mul(w[i - 1], wn);
58             }
59             len_inv[maxbit] = pow(sz, mod - 2);

```

```

60     for (int i = maxbit - 1; ~i; i--) {
61         len_inv[i] = add(len_inv[i + 1], len_inv[i + 1]);
62     }
63 }
64
65 void operator()(vi& v, int& n, int type) {
66     int bit = 0;
67     while ((1 << bit) < n) bit++;
68     int tot = (1 << bit);
69     v.resize(tot, 0);
70     vi rev(tot);
71     n = tot;
72     for (int i = 0; i < tot; i++) {
73         rev[i] = rev[i >> 1] >> 1;
74         if (i & 1) {
75             rev[i] |= tot >> 1;
76         }
77     }
78     for (int i = 0; i < tot; i++) {
79         if (i < rev[i]) {
80             std::swap(v[i], v[rev[i]]);
81         }
82     }
83     for (int midd = 0; (1 << midd) < tot; midd++) {
84         int mid = 1 << midd;
85         int len = mid << 1;
86         for (int i = 0; i < tot; i += len) {
87             for (int j = 0; j < mid; j++) {
88                 int w0 = v[i + j];
89                 int w1 = mul(
90                     w[type == 1 ? (j << maxbit - midd - 1) : (len - j << maxbit - midd - 1)],
91                     v[i + j + mid]);
92                 v[i + j] = add(w0, w1);
93                 v[i + j + mid] = sub(w0, w1);
94             }
95         }
96     }
97     if (type == -1) {
98         for (int i = 0; i < tot; i++) {
99             v[i] = mul(v[i], len_inv[bit]);
100         }
101     }
102 }
103 } NTT;

```

乘法

```

1  polynomial& operator*=(polynomial& a, const polynomial& b) {
2      if (!a.size() || !b.size()) {
3          a.resize(0);
4          return a;
5      }
6      polynomial tmp = b;
7      int deg = a.size() + b.size() - 1;
8      int temp = deg;
9
10     // 项数较小直接硬算
11
12     if ((LL) a.size() * (LL) b.size() <= (LL) deg * 50LL) {
13         tmp.resize(0);
14         tmp.resize(deg, 0);
15         for (int i = 0; i < a.size(); i++) {
16             for (int j = 0; j < b.size(); j++) {
17                 tmp[i + j] = add(tmp[i + j], mul(a[i], b[j]));
18             }
19         }
20         a = tmp;
21         return a;
22     }
23
24     // 项数较多跑 NTT
25
26     NTT(a, deg, 1);
27     NTT(tmp, deg, 1);
28     for (int i = 0; i < deg; i++) {
29         Mul(a[i], tmp[i]);
30     }
31     NTT(a, deg, -1);
32     a.resize(temp);
33     return a;
34 }
35

```

```

36 polynomial operator*(const polynomial& a, const polynomial& b) {
37     polynomial ans = a;
38     return ans *= b;
39 }

```

逆

```

1  polynomial inverse(const polynomial& a) {
2      polynomial ans({pow(a[0], mod - 2)});
3      polynomial temp;
4      polynomial tempa;
5      int deg = a.size();
6      for (int i = 0; (1 << i) < deg; i++) {
7          tempa.resize(0);
8          tempa.resize(1 << i << 1, 0);
9          for (int j = 0; j != tempa.size() and j != deg; j++) {
10             tempa[j] = a[j];
11         }
12         temp = ans * (polynomial({2}) - tempa * ans);
13         if (temp.size() > (1 << i << 1)) {
14             temp.resize(1 << i << 1, 0);
15         }
16         temp.clearzero();
17         std::swap(temp, ans);
18     }
19     ans.resize(deg);
20     return ans;
21 }

```

对数

```

1  polynomial differential(const polynomial& a) {
2      if (!a.size()) {
3          return a;
4      }
5      polynomial ans(vi(a.size() - 1));
6      for (int i = 1; i < a.size(); i++) {
7          ans[i - 1] = mul(a[i], i);
8      }
9      return ans;
10 }
11
12 polynomial integral(const polynomial& a) {
13     polynomial ans(vi(a.size() + 1));
14     for (int i = 0; i < a.size(); i++) {
15         ans[i + 1] = mul(a[i], pow(i + 1, mod - 2));
16     }
17     return ans;
18 }
19
20 polynomial ln(const polynomial& a) {
21     int deg = a.size();
22     polynomial da = differential(a);
23     polynomial inva = inverse(a);
24     polynomial ans = integral(da * inva);
25     ans.resize(deg);
26     return ans;
27 }

```

指数

```

1  polynomial exp(const polynomial& a) {
2      polynomial ans({1});
3      polynomial temp;
4      polynomial tempa;
5      polynomial tempaa;
6      int deg = a.size();
7      for (int i = 0; (1 << i) < deg; i++) {
8          tempa.resize(0);
9          tempa.resize(1 << i << 1, 0);
10         for (int j = 0; j != tempa.size() and j != deg; j++) {
11             tempa[j] = a[j];
12         }
13         tempaa = ans;

```

```

14     tempaa.resize(1 << i << 1);
15     temp = ans * (tempa + polynomial({1}) - ln(tempaa));
16     if (temp.size() > (1 << i << 1)) {
17         temp.resize(1 << i << 1, 0);
18     }
19     temp.clearzero();
20     std::swap(temp, ans);
21 }
22 ans.resize(deg);
23 return ans;
24 }

```

根号

```

1 polynomial sqrt(polynomial& a) {
2     polynomial ans({cipolla(a[0])});
3     if (ans[0] == -1) return ans;
4     polynomial temp;
5     polynomial tempa;
6     polynomial tempaa;
7     int deg = a.size();
8     for (int i = 0; (1 << i) < deg; i++) {
9         tempa.resize(0);
10        tempa.resize(1 << i << 1, 0);
11        for (int j = 0; j != tempa.size() and j != deg; j++) {
12            tempa[j] = a[j];
13        }
14        tempaa = ans;
15        tempaa.resize(1 << i << 1);
16        temp = (tempa * inverse(tempaa) + ans) * inv2;
17        if (temp.size() > (1 << i << 1)) {
18            temp.resize(1 << i << 1, 0);
19        }
20        temp.clearzero();
21        std::swap(temp, ans);
22    }
23    ans.resize(deg);
24    return ans;
25 }
26
27 // 特判 //
28
29 int cnt = 0;
30 for (int i = 0; i < a.size(); i++) {
31     if (a[i] == 0) {
32         cnt++;
33     } else {
34         break;
35     }
36 }
37 if (cnt) {
38     if (cnt == n) {
39         for (int i = 0; i < n; i++) {
40             std::cout << "0 ";
41         }
42         std::cout << endl;
43         return 0;
44     }
45     if (cnt & 1) {
46         std::cout << "-1" << endl;
47         return 0;
48     }
49     polynomial b(vi(a.size() - cnt));
50     for (int i = cnt; i < a.size(); i++) {
51         b[i - cnt] = a[i];
52     }
53     a = b;
54 }
55 a.resize(n - cnt / 2);
56 a = sqrt(a);
57 if (a[0] == -1) {
58     std::cout << "-1" << endl;
59     return 0;
60 }
61 reverse(all(a));
62 a.resize(n);
63 reverse(all(a));

```

6.4 wsy poly

```

1  #include <bits/stdc++.h>
2
3  using ul = std::uint32_t;
4  using li = std::int32_t;
5  using ll = std::int64_t;
6  using ull = std::uint64_t;
7  using llf = long double;
8  using lf = double;
9  using vul = std::vector<ul>;
10 using vvul = std::vector<vul>;
11 using pulb = std::pair<ul, bool>;
12 using vpulb = std::vector<pulb>;
13 using vvpulb = std::vector<vpulb>;
14 using vb = std::vector<bool>;
15
16 const ul base = 998244353;
17
18 std::mt19937 rnd;
19
20 ul plus(ul a, ul b) { return a + b < base ? a + b : a + b - base; }
21
22 ul minus(ul a, ul b) { return a < b ? a + base - b : a - b; }
23
24 ul mul(ul a, ul b) { return ull(a) * ull(b) % base; }
25
26 void exgcd(li a, li b, li& x, li& y) {
27     if (b) {
28         exgcd(b, a % b, y, x);
29         y -= x * (a / b);
30     } else {
31         x = 1;
32         y = 0;
33     }
34 }
35
36 ul inverse(ul a) {
37     li x, y;
38     exgcd(a, base, x, y);
39     return x < 0 ? x + li(base) : x;
40 }
41
42 ul pow(ul a, ul b) {
43     ul ret = 1;
44     ul temp = a;
45     while (b) {
46         if (b & 1) {
47             ret = mul(ret, temp);
48         }
49         temp = mul(temp, temp);
50         b >>= 1;
51     }
52     return ret;
53 }
54
55
56 ul sqrt(ul x) {
57     ul a;
58     ul w2;
59     while (true) {
60         a = rnd() % base;
61         w2 = minus(mul(a, a), x);
62         if (pow(w2, base - 1 >> 1) == base - 1) {
63             break;
64         }
65     }
66     ul b = base + 1 >> 1;
67     ul rs = 1, rt = 0;
68     ul as = a, at = 1;
69     ul qs, qt;
70     while (b) {
71         if (b & 1) {
72             qs = plus(mul(rs, as), mul(mul(rt, at), w2));
73             qt = plus(mul(rs, at), mul(rt, as));
74             rs = qs;
75             rt = qt;
76         }
77         b >>= 1;
78         qs = plus(mul(as, as), mul(mul(at, at), w2));
79         qt = plus(mul(as, at), mul(as, at));
80         as = qs;
81         at = qt;
82     }
83     return rs + rs < base ? rs : base - rs;
84 }

```

```

85 |
86 | ul log(ul x, ul y, bool initd = false) {
87 |     static std::map<ul, ul> bs;
88 |     const ul d = std::round(std::sqrt(1f(base - 1)));
89 |     if (!initd) {
90 |         bs.clear();
91 |         for (ul i = 0, j = 1; i != d; ++i, j = mul(j, x)) {
92 |             bs[j] = i;
93 |         }
94 |     }
95 |     ul temp = inverse(pow(x, d));
96 |     for (ul i = 0, j = 1;; i += d, j = mul(j, temp)) {
97 |         auto it = bs.find(mul(y, j));
98 |         if (it != bs.end()) {
99 |             return it->second + i;
100 |         }
101 |     }
102 | }
103 |
104 | ul powroot(ul x, ul y, bool initd = false) {
105 |     const ul g = 3;
106 |     ul lgx = log(g, x, initd);
107 |     li s, t;
108 |     exgcd(y, base - 1, s, t);
109 |     if (s < 0) {
110 |         s += base - 1;
111 |     }
112 |     return pow(g, ull(s) * ull(lgx) % (base - 1));
113 | }
114 |
115 | class polynomial : public vul {
116 | public:
117 |     void clearzero() {
118 |         while (size() && !back()) {
119 |             pop_back();
120 |         }
121 |     }
122 |     polynomial() = default;
123 |     polynomial(const vul& a) : vul(a) {}
124 |     polynomial(vul&& a) : vul(std::move(a)) {}
125 |     ul degree() const { return size() - 1; }
126 |     ul operator()(ul x) const {
127 |         ul ret = 0;
128 |         for (ul i = size() - 1; ~i; --i) {
129 |             ret = mul(ret, x);
130 |             ret = plus(ret, vul::operator[](i));
131 |         }
132 |         return ret;
133 |     }
134 | };
135 |
136 | polynomial& operator+=(polynomial& a, const polynomial& b) {
137 |     a.resize(std::max(a.size(), b.size()), 0);
138 |     for (ul i = 0; i != b.size(); ++i) {
139 |         a[i] = plus(a[i], b[i]);
140 |     }
141 |     a.clearzero();
142 |     return a;
143 | }
144 |
145 | polynomial operator+(const polynomial& a, const polynomial& b) {
146 |     polynomial ret = a;
147 |     return ret += b;
148 | }
149 |
150 | polynomial& operator-=(polynomial& a, const polynomial& b) {
151 |     a.resize(std::max(a.size(), b.size()), 0);
152 |     for (ul i = 0; i != b.size(); ++i) {
153 |         a[i] = minus(a[i], b[i]);
154 |     }
155 |     a.clearzero();
156 |     return a;
157 | }
158 |
159 | polynomial operator-(const polynomial& a, const polynomial& b) {
160 |     polynomial ret = a;
161 |     return ret -= b;
162 | }
163 |
164 | class ntt_t {
165 | public:
166 |     static const ul lgysz = 20;
167 |     static const ul sz = 1 << lgysz;
168 |     static const ul g = 3;
169 |     ul w[sz + 1];
170 |     ul leninv[lgysz + 1];
171 |     ntt_t() {

```

```

172     ul wn = pow(g, (base - 1) >> lgysz);
173     w[0] = 1;
174     for (ul i = 1; i <= sz; ++i) {
175         w[i] = mul(w[i - 1], wn);
176     }
177     leninv[lgysz] = inverse(sz);
178     for (ul i = lgysz - 1; ~i; --i) {
179         leninv[i] = plus(leninv[i + 1], leninv[i + 1]);
180     }
181 }
182 void operator()(vul& v, ul& n, bool inv) {
183     ul lgn = 0;
184     while ((1 << lgn) < n) {
185         ++lgn;
186     }
187     n = 1 << lgn;
188     v.resize(n, 0);
189     for (ul i = 0, j = 0; i != n; ++i) {
190         if (i < j) {
191             std::swap(v[i], v[j]);
192         }
193         ul k = n >> 1;
194         while (k & j) {
195             j ^= ~k;
196             k >>= 1;
197         }
198         j |= k;
199     }
200     for (ul lgmid = 0; (1 << lgmid) != n; ++lgmid) {
201         ul mid = 1 << lgmid;
202         ul len = mid << 1;
203         for (ul i = 0; i != n; i += len) {
204             for (ul j = 0; j != mid; ++j) {
205                 ul t0 = v[i + j];
206                 ul t1 =
207                     mul(w[inv ? (len - j << lgysz - lgmid - 1) : (j << lgysz - lgmid - 1)],
208                         v[i + j + mid]);
209                 v[i + j] = plus(t0, t1);
210                 v[i + j + mid] = minus(t0, t1);
211             }
212         }
213     }
214     if (inv) {
215         for (ul i = 0; i != n; ++i) {
216             v[i] = mul(v[i], leninv[lgn]);
217         }
218     }
219 }
220 } ntt;
221
222 polynomial& operator*=(polynomial& a, const polynomial& b) {
223     if (!b.size() || !a.size()) {
224         a.resize(0);
225         return a;
226     }
227     polynomial temp = b;
228     ul npmp1 = a.size() + b.size() - 1;
229     if (ull(a.size()) * ull(b.size()) <= ull(npmp1) * ull(50)) {
230         temp.resize(0);
231         temp.resize(npmp1, 0);
232         for (ul i = 0; i != a.size(); ++i) {
233             for (ul j = 0; j != b.size(); ++j) {
234                 temp[i + j] = plus(temp[i + j], mul(a[i], b[j]));
235             }
236         }
237         a = temp;
238         a.clearzero();
239         return a;
240     }
241     ntt(a, npmp1, false);
242     ntt(temp, npmp1, false);
243     for (ul i = 0; i != npmp1; ++i) {
244         a[i] = mul(a[i], temp[i]);
245     }
246     ntt(a, npmp1, true);
247     a.clearzero();
248     return a;
249 }
250
251 polynomial operator*(const polynomial& a, const polynomial& b) {
252     polynomial ret = a;
253     return ret *= b;
254 }
255
256 polynomial& operator*=(polynomial& a, ul b) {
257     if (!b) {
258         a.resize(0);

```

```

259     return a;
260 }
261 for (ul i = 0; i != a.size(); ++i) {
262     a[i] = mul(a[i], b);
263 }
264 return a;
265 }
266
267 polynomial operator*(const polynomial& a, ul b) {
268     polynomial ret = a;
269     return ret *= b;
270 }
271
272 polynomial inverse(const polynomial& a, ul lgdeg) {
273     polynomial ret({inverse(a[0])});
274     polynomial temp;
275     polynomial tempa;
276     for (ul i = 0; i != lgdeg; ++i) {
277         tempa.resize(0);
278         tempa.resize(1 << i << 1, 0);
279         for (ul j = 0; j != tempa.size() && j != a.size(); ++j) {
280             tempa[j] = a[j];
281         }
282         temp = ret * (polynomial({2}) - tempa * ret);
283         if (temp.size() > (1 << i << 1)) {
284             temp.resize(1 << i << 1, 0);
285         }
286         temp.clearzero();
287         std::swap(temp, ret);
288     }
289     return ret;
290 }
291
292 void quotientremain(const polynomial& a, polynomial b, polynomial& q, polynomial& r) {
293     if (a.size() < b.size()) {
294         q = polynomial();
295         r = std::move(a);
296         return;
297     }
298     std::reverse(b.begin(), b.end());
299     auto ta = a;
300     std::reverse(ta.begin(), ta.end());
301     ul n = a.size() - 1;
302     ul m = b.size() - 1;
303     ta.resize(n - m + 1);
304     ul lgnmmp1 = 0;
305     while ((1 << lgnmmp1) < n - m + 1) {
306         ++lgnmmp1;
307     }
308     q = ta * inverse(b, lgnmmp1);
309     q.resize(n - m + 1);
310     std::reverse(b.begin(), b.end());
311     std::reverse(q.begin(), q.end());
312     r = a - b * q;
313 }
314
315 polynomial mod(const polynomial& a, const polynomial& b) {
316     polynomial q, r;
317     quotientremain(a, b, q, r);
318     return r;
319 }
320
321 polynomial quotient(const polynomial& a, const polynomial& b) {
322     polynomial q, r;
323     quotientremain(a, b, q, r);
324     return q;
325 }
326
327 polynomial sqrt(const polynomial& a, ul lgdeg) {
328     polynomial ret({sqrt(a[0])});
329     polynomial temp;
330     polynomial tempa;
331     for (ul i = 0; i != lgdeg; ++i) {
332         tempa.resize(0);
333         tempa.resize(1 << i << 1, 0);
334         for (ul j = 0; j != tempa.size() && j != a.size(); ++j) {
335             tempa[j] = a[j];
336         }
337         temp = (tempa * inverse(ret, i + 1) + ret) * (base + 1 >> 1);
338         if (temp.size() > (1 << i << 1)) {
339             temp.resize(1 << i << 1, 0);
340         }
341         temp.clearzero();
342         std::swap(temp, ret);
343     }
344     return ret;
345 }

```



```

346 polynomial differential(const polynomial& a) {
347     if (!a.size()) {
348         return a;
349     }
350     polynomial ret(vul(a.size() - 1, 0));
351     for (ul i = 1; i != a.size(); ++i) {
352         ret[i - 1] = mul(a[i], i);
353     }
354     return ret;
355 }
356
357 polynomial integral(const polynomial& a) {
358     polynomial ret(vul(a.size() + 1, 0));
359     for (ul i = 0; i != a.size(); ++i) {
360         ret[i + 1] = mul(a[i], inverse(i + 1));
361     }
362     return ret;
363 }
364
365 polynomial ln(const polynomial& a, ul lgdeg) {
366     polynomial da = differential(a);
367     polynomial inva = inverse(a, lgdeg);
368     polynomial ret = integral(da * inva);
369     if (ret.size() > (1 << lgdeg)) {
370         ret.resize(1 << lgdeg);
371         ret.clearzero();
372     }
373     return ret;
374 }
375
376 polynomial exp(const polynomial& a, ul lgdeg) {
377     polynomial ret({1});
378     polynomial temp;
379     polynomial tempa;
380     for (ul i = 0; i != lgdeg; ++i) {
381         tempa.resize(0);
382         tempa.resize(1 << i << 1, 0);
383         for (ul j = 0; j != tempa.size() && j != a.size(); ++j) {
384             tempa[j] = a[j];
385         }
386         temp = ret * (polynomial({1}) - ln(ret, i + 1) + tempa);
387         if (temp.size() > (1 << i << 1)) {
388             temp.resize(1 << i << 1, 0);
389         }
390         temp.clearzero();
391         std::swap(temp, ret);
392     }
393     return ret;
394 }
395
396 polynomial pow(const polynomial& a, ul k, ul lgdeg) { return exp(ln(a, lgdeg) * k, lgdeg); }
397
398 polynomial alpi[1 << 16][17];
399
400 polynomial getalpi(const ul x[], ul l, ul lgrml) {
401     if (lgrml == 0) {
402         return alpi[l][lgrml] = vul({minus(0, x[l]), 1});
403     }
404     return alpi[l][lgrml] = getalpi(x, l, lgrml - 1) * getalpi(x, l + (1 << lgrml - 1), lgrml - 1);
405 }
406
407 void multians(const polynomial& f, const ul x[], ul y[], ul l, ul lgrml) {
408     if (f.size() <= 700) {
409         for (ul i = l; i != l + (1 << lgrml); ++i) {
410             y[i] = f(x[i]);
411         }
412         return;
413     }
414     if (lgrml == 0) {
415         y[l] = f(x[l]);
416         return;
417     }
418     multians(mod(f, alpi[l][lgrml - 1]), x, y, l, lgrml - 1);
419     multians(mod(f, alpi[l + (1 << lgrml - 1)][lgrml - 1]), x, y, l + (1 << lgrml - 1), lgrml - 1);
420 }
421
422 ul sqrt(ul x) {
423     ul a;
424     ul w2;
425     while (true) {
426         a = rnd() % base;
427         w2 = minus(mul(a, a), x);
428         if (pow(w2, base - 1 >> 1) == base - 1) {
429             break;
430         }
431     }
432 }

```

```

433     ul b = base + 1 >> 1;
434     ul rs = 1, rt = 0;
435     ul as = a, at = 1;
436     ul qs, qt;
437     while (b) {
438         if (b & 1) {
439             qs = plus(mul(rs, as), mul(mul(rt, at), w2));
440             qt = plus(mul(rs, at), mul(rt, as));
441             rs = qs;
442             rt = qt;
443         }
444         b >>= 1;
445         qs = plus(mul(as, as), mul(mul(at, at), w2));
446         qt = plus(mul(as, at), mul(as, at));
447         as = qs;
448         at = qt;
449     }
450     return rs + rs < base ? rs : base - rs;
451 }
452
453 ul log(ul x, ul y, bool initied = false) {
454     static std::map<ul, ul> bs;
455     const ul d = std::round(std::sqrt(1f(base - 1)));
456     if (!initied) {
457         bs.clear();
458         for (ul i = 0, j = 1; i != d; ++i, j = mul(j, x)) {
459             bs[j] = i;
460         }
461     }
462     ul temp = inverse(pow(x, d));
463     for (ul i = 0, j = 1;; i += d, j = mul(j, temp)) {
464         auto it = bs.find(mul(y, j));
465         if (it != bs.end()) {
466             return it->second + i;
467         }
468     }
469 }
470
471 ul powroot(ul x, ul y, bool initied = false) {
472     const ul g = 3;
473     ul lgx = log(g, x, initied);
474     li s, t;
475     exgcd(y, base - 1, s, t);
476     if (s < 0) {
477         s += base - 1;
478     }
479     return pow(g, ull(s) * ull(lgx) % (base - 1));
480 }
481
482 ul n;
483
484 int main() {
485     std::scanf("%u", &n);
486     polynomial f;
487     for (ul i = 0; i <= n; ++i) {
488         ul t;
489         std::scanf("%u", &t);
490         f.push_back(t % base);
491     }
492     polynomial g = exp(ln(f * inverse(f[0]), 17) * inverse(3), 17) * powroot(f[0], 3);
493     while (g.size() <= n) {
494         g.push_back(0);
495     }
496     for (ul i = 0; i <= n; ++i) {
497         if (i) {
498             std::putchar(' ');
499         }
500         std::printf("%u", g[i]);
501     }
502     std::putchar('\n');
503     return 0;
504 }

```

6.5 Lagrange interpolation

一般的插值

给出一个多项式 $f(x)$ 上的 n 个点 (x_i, y_i) , 求 $f(k)$.

插值的结果是

$$f(x) = \sum_{i=1}^n y_i \prod_{j \neq i} \frac{x - x_j}{x_i - x_j},$$

直接带入 k 并且取模即可, 时间复杂度 $O(n^2)$.

```

1  /* interpolation */
2  auto lagrange = (const vi& x, const vi& y, int n, int k) {
3      for (int i = 1; i <= n; i++) {
4          LL s1 = y[i] % mod, s2 = 1ll;
5          for (int j = 1; j <= n; j++) {
6              if (i != j) {
7                  s1 = s1 * (k - x[j]) % mod;
8                  s2 = s2 * (x[i] - x[j]) % mod;
9              }
10         }
11         Add(ans, mul(s1, quick_power(s2, mod - 2, mod)));
12     }
13     return ans;
14 };

```

坐标连续的插值

给出的点是 $(0, y_0), \dots, (n, y_n)$.

```

1  /* interpolation */
2  auto lagrange(int n, std::vector<Z> y, int x) -> Z {
3      if (x <= n) return y[x];
4      Z s1 = 1, s2 = 0;
5      for (int i = 1; i <= n; i++) s1 *= (x - i);
6      for (int i = 1; i <= n; i++) {
7          Z res = ((n - i) & 1 ? -y[i] : y[i]);
8          res /= (x - i);
9          res *= fiv[i - 1];
10         res *= fiv[n - i];
11         s2 += res;
12     }
13     return s1 * s2;
14 }

```

7 math - game theory

7.1 nim game

若 nim 和为 0, 则先手必败.

暴力打表.

```

1 vi SG(21, -1); /* 记忆化 */
2 std::function<int(int, int)> sg = [&](int x) -> int {
3     if (/* 为最终态 */) return SG[x] = 0;
4     if (SG[x] != -1) return SG[x];
5     vi st;
6     for (/* 枚举所有可到达的状态 y */) {
7         st.push_back(sg(y));
8     }
9     std::sort(all(st));
10    for (int i = 0; i < st.size(); i++) {
11        if (st[i] != i) return SG[x] = i;
12    }
13    return SG[x] = st.size();
14 };

```

7.2 anti - nim game

若

1. 所有堆的石子均为一个, 且 nim 和不为 0,
2. 至少有一堆石子超过一个, 且 nim 和为 0,

则先手必败.

8 math - linear algebra

8.1 matrix

determinant mod 998244353

```

1  /* determinant */
2  auto det = [&](int n, vvi e) -> int {
3      int ans = 1;
4      for (int i = 1; i <= n; i++) {
5          if (a[i][i] == 0) {
6              for (int j = i + 1; j <= n; j++) {
7                  if (a[j][i] != 0) {
8                      for (int k = i; k <= n; k++) {
9                          std::swap(a[i][k], a[j][k]);
10                     }
11                     ans = sub(mod, ans);
12                     break;
13                 }
14             }
15         }
16         if (a[i][i] == 0) return 0;
17         Mul(ans, a[i][i]);
18         int x = pow(a[i][i], mod - 2);
19         for (int k = i; k <= n; k++) {
20             Mul(a[i][k], x);
21         }
22         for (int j = i + 1; j <= n; j++) {
23             int x = a[j][i];
24             for (int k = i; k <= n; k++) {
25                 Sub(a[j][k], mul(a[i][k], x));
26             }
27         }
28     }
29     return ans;
30 };

```

determinant mod non-prime

```

1  /* determinant @ wrb */
2  int a[609][609];
3  int main() {
4      int n, P, z = 1;
5      std::cin >> n >> P;
6      for (int i = 1; i <= n; ++i) {
7          for (int j = 1; j <= n; ++j) std::cin >> a[i][j];
8      }
9      for (int i = 1; i <= n; ++i) {
10         for (int j = i + 1; j <= n; ++j) {
11             while (a[j][i]) {
12                 z = -z;
13                 int d = a[i][i] / a[j][i];
14                 for (int k = i; k <= n; ++k) {
15                     int x = (a[i][k] - 111 * d * a[j][k]) % P;
16                     a[i][k] = a[j][k], a[j][k] = x;
17                 }
18             }
19         }
20         z = 111 * z * a[i][i] % P;
21     }
22     std::cout << (z + P) % P;
23 }

```

matrix multiplication

$A_{n \times m}$ 与 $B_{m \times k}$ 相乘并模 998244353.

```

1  /* matrix multiplication */
2  auto matmul = [&](int n, int m, int k, const vvi& a, const vvi& b) -> vvi {
3      vvi c(n + 1, vi(k + 1));
4      for (int i = 1; i <= n; i++) {
5          for (int l = 1; l <= m; l++) {

```

```

6         int x = a[i][1];
7         for (int j = 1; j <= k; j++) {
8             Add(c[i][j], mul(x, b[1][j]));
9         }
10    }
11 }
12 return c;
13 };

```

8.2 linear basis

```

1  /* linear basis */
2  vl p(63), s(63); /* basis and case */
3  auto insert = [&](LL x, int id) {
4      LL ans = 0;
5      for (int i = 62; i >= 0; i--) {
6          if (~(x >> i) & 1) continue;
7          if (!p[i]) {
8              p[i] = x;
9              s[i] = ans ^ (1ll << id);
10             break;
11         }
12         x ^= p[i], ans ^= s[i];
13     }
14     return x;
15 };
16 auto query = [&](LL x) {
17     LL ans = 0;
18     for (int i = 62; i >= 0; i--) {
19         if (~(x >> i) & 1) continue;
20         x ^= p[i], ans ^= s[i];
21     }
22     return (x ? -1 : ans);
23 };
24 auto queryMax = [&]() {
25     LL ans = 0;
26     for (int i = 62; i >= 0; i--)
27         if ((ans ^ p[i]) > ans) ans ^= p[i];
28     return ans;
29 };

```

```

1  /* linear basis @ wrb */
2  template<typename T, const int M = sizeof(T) * 8>
3  struct Liner_Basis {
4      T a[M];
5      size_t sz;
6      Liner_Basis() : a(), sz() {}
7      size_t size() const {
8          return sz;
9      }
10     void clear() {
11         memset(a, 0, sizeof a);
12     }
13     bool ins(T x) {
14         for (size_t i = M - 1; ~i && x; --i)
15             if (x >> i & 1) {
16                 if (a[i]) x ^= a[i];
17                 else return a[i] = x, true;
18             }
19         return false;
20     }
21     Liner_Basis& operator+=(const Liner_Basis&_) {
22         for (T x : _a) if (x) this->ins(x);
23         return *this;
24     }
25     Liner_Basis operator+(const Liner_Basis&_) {
26         Liner_Basis z = *this;
27         return z += _;
28     }
29     T qry(T x = 0) {
30         for (size_t i = M - 1; ~i; --i)
31             if ((x ^ a[i]) > x) x ^= a[i];
32         return x;
33     }
34 };
35 template<typename T>
36 using LB = Liner_Basis<T>;
37
38 //////////////////////////////////////
39
40

```

```

41 struct Liner_Basis {
42     using u64 = unsigned long long;
43     static const size_t M = 60;
44     u64 a[M + 1];
45     size_t sz;
46     size_t size() {
47         return sz;
48     }
49     Liner_Basis& operator+=(u64 x) {
50         for (size_t i = M; ~i && x; --i)
51             if (x >> i & 1)
52                 if (a[i]) x ^= a[i];
53             else return a[i] = x, ++sz, *this;
54         return *this;
55     }
56     Liner_Basis& operator+=(const Liner_Basis&) {
57         for (u64 x : _a) if (x) *this += x;
58         return *this;
59     }
60     Liner_Basis operator+(u64 x) {
61         Liner_Basis z = *this;
62         return z += x;
63     }
64     Liner_Basis operator+(const Liner_Basis&) {
65         Liner_Basis z = *this;
66         return z += _;
67     }
68     u64 qry(u64 x = 0) {
69         for (size_t i = M; ~i; --i)
70             if ((x ^ a[i]) > x) x ^= a[i];
71         return x;
72     }
73     u64 rank(u64 x) {
74         u64 h = 1, z = 0;
75         for (size_t i = 0; i <= M; ++i)
76             if (a[i]) {
77                 if (x >> i & 1) z += h;
78                 h <<= 1;
79             }
80         return z;
81     }
82     u64 kth(u64 x) {
83         u64 z = 0;
84         for (size_t i = M; ~i; --i)
85             if (x >> i & 1) z ^= a[i];
86         return z;
87     }
88 }v;
89 using LB = Liner_Basis;

```

8.3 linear programming

8.4 bm

```

1  /* bm @ wrb */
2  const int p = 998244353;
3  auto power = [](int a, int b = p - 2) {
4      int z = 1;
5      while (b) {
6          if (b & 1) z = 1ll * z * a % p;
7          a = 1ll * a * a % p, b >>= 1;
8      }
9      return z;
10 };
11 vector<int> berlekamp_massey(const vector<int> &a) {
12     vector<int> v, last; // v is the answer, 0-based, p is the module
13     int k = -1, delta = 0;
14
15     for (int i = 0; i < (int)a.size(); i++) {
16         int tmp = 0;
17         for (int j = 0; j < (int)v.size(); j++)
18             tmp = (tmp + (long long)a[i - j - 1] * v[j]) % p;
19
20         if (a[i] == tmp) continue;
21
22         if (k < 0) {
23             k = i;
24             delta = (a[i] - tmp + p) % p;
25             v = vector<int>(i + 1);
26
27             continue;

```

```

28     }
29
30     vector<int> u = v;
31     int val = (long long)(a[i] - tmp + p) * power(delta, p - 2) % p;
32
33     if (v.size() < last.size() + i - k) v.resize(last.size() + i - k);
34
35     (v[i - k - 1] += val) %= p;
36
37     for (int j = 0; j < (int)last.size(); j++) {
38         v[i - k + j] = (v[i - k + j] - (long long)val * last[j]) % p;
39         if (v[i - k + j] < 0) v[i - k + j] += p;
40     }
41
42     if ((int)u.size() - i < (int)last.size() - k) {
43         last = u;
44         k = i;
45         delta = a[i] - tmp;
46         if (delta < 0) delta += p;
47     }
48 }
49
50 for (auto &x : v) x = (p - x) % p;
51 v.insert(v.begin(), 1);
52
53 return v;
54 // $\forall i, \sum_{j=0}^m a_{i-j} v_j = 0$
55 }

```


9 complex number

```

1  tandu struct Comp {
2      T a, b;
3      Comp(T _a = 0, T _b = 0) { a = _a, b = _b; }
4      Comp operator+(const Comp& x) const { return Comp(a + x.a, b + x.b); }
5      Comp operator-(const Comp& x) const { return Comp(a - x.a, b - x.b); }
6      Comp operator*(const Comp& x) const { return Comp(a * x.a - b * x.b, a * x.b + b * x.a); }
7      bool operator==(const Comp& x) const { return a == x.a and b == x.b; }
8      T real() { return a; }
9      T imag() { return b; }
10     U norm() { return (U) a * a + (U) b * b; }
11     Comp conj() { return Comp(a, -b); }
12     Comp operator/(const Comp& x) const {
13         Comp y = x;
14         Comp c = Comp(a, b) * y.conj();
15         T d = y.norm();
16         return Comp(c.a / d, c.b / d);
17     }
18 };
19 typedef Comp<LL, LL> complex;
20 complex gcd(complex a, complex b) {
21     LL d = b.norm();
22     if (d == 0) return a;
23     std::vector<complex> v(4);
24     complex c = a * b.conj();
25     auto fdiv = [&](LL a, LL b) -> LL { return a / b - ((a ^ b) < 0 && (a % b)); };
26     v[0] = complex(fdiv(c.real(), d), fdiv(c.imag(), d));
27     v[1] = v[0] + complex(1, 0);
28     v[2] = v[0] + complex(0, 1);
29     v[3] = v[0] + complex(1, 1);
30     for (auto& x : v) {
31         x = a - x * b;
32     }
33     std::sort(all(v), [&](complex a, complex b) { return a.norm() < b.norm(); });
34     return gcd(b, v[0]);
35 };

```

10 graph

10.1 topology sort

```

1  /* topology sort */
2  vi top;
3  auto topsort = [&]() -> bool {
4      vi d(n + 1);
5      std::queue<int> q;
6      for (int i = 1; i <= n; i++) {
7          d[i] = e[i].size();
8          if (!d[i]) q.push(i);
9      }
10     while (!q.empty()) {
11         int u = q.front();
12         q.pop();
13         top.push_back(u);
14         for (auto v : e[u]) {
15             d[v]--;
16             if (!d[v]) q.push(v);
17         }
18     }
19     if (top.size() != n) return false;
20     return true;
21 };

```

10.2 shortest path

Floyd

```

1  /* floyd */
2  auto floyd = [&]() -> vvi {
3      vvi dist(n + 1, vi(n + 1, inf));
4      for (int i = 1; i <= n; i++) {
5          for (int j = 1; j <= n; j++) {
6              Min(dist[i][j], e[i][j]);
7          }
8          dist[i][i] = 0;
9      }
10     for (int k = 1; k <= n; k++) {
11         for (int i = 1; i <= n; i++) {
12             for (int j = 1; j <= n; j++) {
13                 Min(dist[i][j], dist[i][k] + dist[k][j]);
14             }
15         }
16     }
17     return dist;
18 };

```

Dijkstra

```

1  /* dijkstra */
2  auto dijkstra = [&](int s) -> vl {
3      vl dist(n + 1, INF);
4      vi vis(n + 1, 0);
5      dist[s] = 0;
6      std::priority_queue<PLI, std::vector<PLI>, std::greater<PLI>> q;
7      q.emplace(0LL, s);
8      while (!q.empty()) {
9          auto [dis, u] = q.top();
10         q.pop();
11         if (vis[u]) continue;
12         vis[u] = 1;
13         for (const auto& [v, w] : e[u]) {
14             if (dist[v] > dis + w) {
15                 dist[v] = dis + w;
16                 q.emplace(dist[v], v);
17             }
18         }
19     }
20     return dist;
21 };

```

SPFA

```

1  /* SPFA */
2  int n, m, s;
3  vl dist(n + 1, INF);
4  std::vector<bool> vis(n + 1);
5  std::vector<PLI> e(n + 1);
6  void spfa(int s){
7      for (int i = 1; i <= n; i++) dist[i] = INF;
8      dist[s] = 0;
9      std::queue<int> q;
10     q.push(s);
11     vis[s] = true;
12     while(q.size()){
13         auto u = q.front();
14         q.pop();
15         vis[u] = false;
16         for(const auto& [v, w] : e[u]){
17             if(dist[v] > dist[u] + w){
18                 dist[v] = dist[u] + w;
19                 if(!vis[v]){
20                     q.push(v);
21                     vis[v] = true;
22                 }
23             }
24         }
25     }
26 }

```

Johnson

```

1  /* johnson */
2  auto johnson = [&]() -> vvl {
3      /* 负环 */
4      vl dist1(n + 1);
5      vi vis(n + 1), cnt(n + 1);
6      auto spfa = [&]() -> bool {
7          std::queue<int> q;
8          for (int u = 1; u <= n; u++) {
9              q.push(u);
10             vis[u] = false;
11         }
12         while (!q.empty()) {
13             auto u = q.front();
14             q.pop();
15             vis[u] = false;
16             for (auto [v, w] : e[u]) {
17                 if (dist1[v] > dist1[u] + w) {
18                     dist1[v] = dist1[u] + w;
19                     Max(cnt[v], cnt[u] + 1);
20                     if (cnt[v] >= n) return true;
21                     if (!vis[v]) {
22                         q.push(v);
23                         vis[v] = true;
24                     }
25                 }
26             }
27         }
28         return false;
29     };
30     /* dijkstra */
31     vl dist2(n + 1);
32     auto dijkstra = [&](int s) {
33         for (int u = 1; u <= n; u++) {
34             dist2[u] = 1e9;
35             vis[u] = false;
36         }
37         dist2[s] = 0;
38         std::priority_queue<PLI, std::vector<PLI>, std::greater<PLI>> q;
39         q.emplace(0, s);
40         while (!q.empty()) {
41             auto [d, u] = q.top();
42             q.pop();
43             if (vis[u]) continue;
44             vis[u] = true;
45             for (const auto& [v, w] : e[u]) {
46                 if (dist2[v] > d + w) {
47                     dist2[v] = d + w;
48                     q.emplace(dist2[v], v);
49                 }
50             }
51         }
52     };
53 }

```

```

51     }
52 };
53 if (spfa()) return vvl{};
54 for (int u = 1; u <= n; u++) {
55     for (auto& [v, w] : e[u]) {
56         w += dist1[u] - dist1[v];
57     }
58 }
59 vvl dist(n + 1, vl(n + 1));
60 for (int u; u <= n; u++) {
61     dijkstra(u);
62     for (int v = 1; v <= n; v++) {
63         if (dist2[v] == 1e9) {
64             dist[u][v] = INF;
65         } else {
66             dist[u][v] = dist2[v] + dist1[v] - dist1[u];
67         }
68     }
69 }
70 return dist;
71 };

```

最短路计数 - Dijkstra

```

1  /* dijkstra */
2  auto dijkstra = [&](int s) -> std::pair<vl, vi> {
3      vl dist(n + 1, INF);
4      vi cnt(n + 1, vis(n + 1));
5      dist[s] = 0;
6      cnt[s] = 1;
7      std::priority_queue<PLI, std::vector<PLI>, std::greater<PLI>> q;
8      q.emplace(OLL, s);
9      while (!q.empty()) {
10         auto [dis, u] = q.top();
11         q.pop();
12         if (vis[u]) continue;
13         vis[u] = 1;
14         for (const auto& [v, w] : e[u]) {
15             if (dist[v] > dis + w) {
16                 dist[v] = dis + w;
17                 cnt[v] = cnt[u];
18                 q.push({dist[v], v});
19             } else if (dist[v] == dis + w) {
20                 // cnt[v] += cnt[u];
21                 cnt[v] += cnt[u];
22                 cnt[v] %= mod;
23             }
24         }
25     }
26     return {dist, cnt};
27 };

```

最短路计数 - Floyd

```

1  /* floyd */
2  auto floyd() = [&] -> std::pair<vvi, vvi> {
3      vvi dist(n + 1, vi(n + 1, inf));
4      vvi cnt(n + 1, vi(n + 1));
5      for (int i = 1; i <= n; i++) {
6          for (int j = 1; j <= n; j++) {
7              Min(dist[i][j], e[i][j]);
8          }
9          dist[i][i] = 0;
10     }
11     for (int k = 1; k <= n; k++) {
12         for (int i = 1; i <= n; i++) {
13             for (int j = 1; j <= n; j++) {
14                 if (dist[i][j] == dist[i][k] + dist[k][j]) {
15                     cnt[i][j] += cnt[i][k] * cnt[k][j];
16                 } else if (dist[i][j] > dist[i][k] + dist[k][j]) {
17                     cnt[i][j] = cnt[i][k] * cnt[k][j];
18                     dist[i][j] = dist[i][k] + dist[k][j];
19                 }
20             }
21         }
22     }
23     return {dist, cnt};
24 };

```

负环

判断的是最短路长度.

```

1  /* SPFA */
2  auto spfa = [&]() -> bool {
3      std::queue<int> q;
4      vi vis(n + 1), cnt(n + 1);
5      for (int i = 1; i <= n; i++) {
6          q.push(i);
7          vis[i] = true;
8      }
9      while (!q.empty()) {
10         auto u = q.front();
11         q.pop();
12         vis[u] = false;
13         for (const auto& [v, w] : e[u]) {
14             if (dist[v] > dist[u] + w) {
15                 dist[v] = dist[u] + w;
16                 cnt[v] = cnt[u] + 1;
17                 if (cnt[v] >= n) return true;
18                 if (!vis[v]) {
19                     q.push(v);
20                     vis[v] = true;
21                 }
22             }
23         }
24     }
25     return false;
26 }

```

10.3 minimum spanning tree

Kruskal

```

1  /* kruskal */
2  std::vector<std::tuple<int, int, int>> edge;
3  auto kruskal = [&]() -> int {
4      std::sort(all(edge), [&](std::tuple<int, int, int> a, std::tuple<int, int, int> b) {
5          auto [x1, y1, w1] = a;
6          auto [x2, y2, w2] = b;
7          return w1 < w2;
8      });
9      int res = 0, cnt = 0;
10     for (int i = 0; i < m; i++) {
11         auto [a, b, w] = edge[i];
12         a = find(a), b = find(b);
13         if (a != b) {
14             fa[a] = b;
15             res += w;
16             /* res = std::max(res, w); */
17             cnt++;
18         }
19     }
20     if (cnt < n - 1) return -1;
21     return res;
22 }

```

10.4 SCC

Tarjan

```

1  /* tarjan */
2  vi dfn(n + 1), low(n + 1), stk(n + 1), belong(n + 1);
3  int timestamp = 0, top = 0, scc_cnt = 0;
4  std::vector<bool> in_stk(n + 1);
5  auto tarjan = [&](auto&& self, int u) -> void {
6      dfn[u] = low[u] = ++timestamp;
7      stk[++top] = u;
8      in_stk[u] = true;
9      for (const auto& v : e[u]) {
10         if (!dfn[v]) {
11             self(self, v);

```

```

12         Min(low[u], low[v]);
13     } else if (in_stk[v]) {
14         Min(low[u], dfn[v]);
15     }
16 }
17 if (dfn[u] == low[u]) {
18     scc_cnt++;
19     int v;
20     do {
21         v = stk[top--];
22         in_stk[v] = false;
23         belong[v] = scc_cnt;
24     } while (v != u);
25 }
26 };

```

10.5 DCC

点双连通分量

求点双连通分量.

```

1 vi dfn(n + 1), low(n + 1), is_bcc(n + 1), stk;
2 int timestamp = 0, bcc_cnt = 0, root = 0;
3 vvi bcc(2 * n + 1);
4 std::function<void(int, int)> tarjan = [&](int u, int fa) {
5     dfn[u] = low[u] = ++timestamp;
6     int child = 0;
7     stk.push_back(u);
8     if (u == root and e[u].empty()) {
9         bcc_cnt++;
10        bcc[bcc_cnt].push_back(u);
11        return;
12    }
13    for (auto v : e[u]) {
14        if (!dfn[v]) {
15            tarjan(v, u);
16            low[u] = std::min(low[u], low[v]);
17            if (low[v] >= dfn[u]) {
18                child++;
19                if (u != root or child > 1) {
20                    is_bcc[u] = 1;
21                }
22                bcc_cnt++;
23                int z;
24                do {
25                    z = stk.back();
26                    stk.pop_back();
27                    bcc[bcc_cnt].push_back(z);
28                } while (z != v);
29                bcc[bcc_cnt].push_back(u);
30            }
31        } else if (v != fa) {
32            low[u] = std::min(low[u], dfn[v]);
33        }
34    }
35 };
36 for (int i = 1; i <= n; i++) {
37     if (!dfn[i]) {
38         root = i;
39         tarjan(i, i);
40     }
41 }

```

求割点.

```

1 vi dfn(n + 1), low(n + 1), is_bcc(n + 1);
2 int timestamp = 0, bcc = 0, root = 0;
3 std::function<void(int, int)> tarjan = [&](int u, int fa) {
4     dfn[u] = low[u] = ++timestamp;
5     int child = 0;
6     for (auto v : e[u]) {
7         if (!dfn[v]) {
8             tarjan(v, u);
9             low[u] = std::min(low[u], low[v]);
10            if (low[v] >= dfn[u]) {
11                child++;
12                if ((u != root or child > 1) and !is_bcc[u]) {
13                    bcc++;

```

```

14         is_bcc[u] = 1;
15     }
16 } else if (v != fa) {
17     low[u] = std::min(low[u], dfn[v]);
18 }
19 }
20 }
21 };
22 for (int i = 1; i <= n; i++) {
23     if (!dfn[i]) {
24         root = i;
25         tarjan(i, i);
26     }
27 }

```

边双连通分量

求边双连通分量.

```

1  std::vector<vpi> e(n + 1);
2  for (int i = 1; i <= m; i++) {
3      int u, v;
4      std::cin >> u >> v;
5      e[u].emplace_back(v, i);
6      e[v].emplace_back(u, i);
7  }
8  vi dfn(n + 1), low(n + 1), is_ecc(n + 1), fa(n + 1), stk;
9  int timestamp = 0, ecc_cnt = 0;
10 vvi ecc(2 * n + 1);
11 std::function<void(int, int)> tarjan = [&](int u, int id) {
12     low[u] = dfn[u] = ++timestamp;
13     stk.push_back(u);
14     for (auto [v, idx] : e[u]) {
15         if (!dfn[v]) {
16             tarjan(v, idx);
17             low[u] = std::min(low[u], low[v]);
18         } else if (idx != id) {
19             low[u] = std::min(low[u], dfn[v]);
20         }
21     }
22     if (dfn[u] == low[u]) {
23         ecc_cnt++;
24         int v;
25         do {
26             v = stk.back();
27             stk.pop_back();
28             ecc[ecc_cnt].push_back(v);
29         } while (v != u);
30     }
31 };
32 for (int i = 1; i <= n; i++) {
33     if (!dfn[i]) {
34         tarjan(i, 0);
35     }
36 }

```

另一个版本

```

1  /* DCC @ wrb */
2  // 割点
3  std::vector<int> G[N];
4  int dfn[N], low[N], is_cut[N], tm, rt;
5  void tar(int u) {
6      int c = 0;
7      dfn[u] = low[u] = ++tm;
8      for (int v : G[u]) {
9          if (!dfn[v]) {
10             ++c, tar(v);
11             low[u] = std::min(low[u], low[v]);
12             if (low[v] == dfn[u]) is_cut[u] = 1;
13         } else low[u] = std::min(low[u], dfn[v]);
14     }
15     if (u == rt) is_cut[u] = c > 1;
16 }
17 int main() {
18     int n, m;
19     std::cin >> n >> m;
20     for (int x, y; m--;) {

```

```

21     std::cin >> x >> y;
22     G[x].emplace_back(y);
23     G[y].emplace_back(x);
24 }
25 for (rt = 1; rt <= n; ++rt) {
26     if (!dfn[rt]) tar(rt);
27 }
28 std::vector<int> cut_v;
29 for (int i = 1; i <= n; ++i) {
30     if (is_cut[i]) cut_v.emplace_back(i);
31 }
32 std::cout << cut_v.size() << '\n';
33 for (int x : cut_v) std::cout << x << ' ';
34 }
35
36
37 // 桥
38 std::vector<std::pair<int, int>> G[N], brg;
39 int dfn[N], low[N], rt, tm;
40 void tar(int u, int fa, int fr) {
41     dfn[u] = low[u] = ++tm;
42     for (auto [v, i] : G[u]) {
43         if (!dfn[v]) {
44             tar(v, u, i), low[u] = std::min(low[u], low[v]);
45         } else if (i != fr) {
46             low[u] = std::min(low[u], dfn[v]);
47         }
48     }
49     if (u != rt && dfn[u] == low[u]) {
50         brg.emplace_back(std::minmax(u, fa));
51     }
52 }
53 int main() {
54     int n, m;
55     std::cin >> n >> m;
56     for (int i = 1, x, y; i <= m; ++i) {
57         std::cin >> x >> y;
58         G[x].emplace_back(y, i);
59         G[y].emplace_back(x, i);
60     }
61     for (rt = 1; rt <= n; ++rt) {
62         if (!dfn[rt]) tar(rt, -1, -1);
63     }
64     std::sort(begin(brg), end(brg));
65     for (auto [u, v] : brg) {
66         std::cout << u << ' ' << v << '\n';
67     }
68 }
69
70
71 // 点双
72 std::vector<int> G[N];
73 std::vector<std::vector<int>> bcc;
74 int dfn[N], low[N], tm, st[N], tp, rt;
75 void tar(int u) {
76     dfn[u] = low[u] = ++tm, st[++tp] = u;
77     if (G[u].empty()) bcc.push_back({u});
78     for (int v : G[u]) {
79         if (!dfn[v]) {
80             tar(v), low[u] = std::min(low[u], low[v]);
81             if (low[v] == dfn[u]) {
82                 bcc.push_back({u});
83                 do {
84                     bcc.back().emplace_back(st[tp]);
85                 } while (st[tp--] != v);
86             }
87         } else low[u] = std::min(low[u], dfn[v]);
88     }
89 }
90 int main() {
91     std::ios::sync_with_stdio(0);
92     std::cin.tie(0);
93     int n, m;
94     std::cin >> n >> m;
95     for (int x, y; m--;) {
96         std::cin >> x >> y;
97         G[x].emplace_back(y);
98         G[y].emplace_back(x);
99     }
100     for (int i = 0; i < n; ++i) tar(i);
101     std::cout << bcc.size() << '\n';
102     for (auto v : bcc) {
103         std::cout << v.size() << ' ';
104         for (int x : v) std::cout << x << ' ';
105         std::cout << '\n';
106     }
107 }

```



```

108
109
110 // 边双
111 std::vector<std::pair<int, int>> G[N];
112 std::vector<std::vector<int>> becc;
113 int dfn[N], low[N], tm, st[N], tp;
114 void tar(int u, int fr) {
115     dfn[u] = low[u] = ++tm, st[++tp] = u;
116     for (auto [v, i] : G[u]) {
117         if (!dfn[v]) {
118             tar(v, i), low[u] = std::min(low[u], low[v]);
119         } else if (i != fr) {
120             low[u] = std::min(low[u], dfn[v]);
121         }
122     }
123     if (dfn[u] == low[u]) {
124         becc.emplace_back();
125         do {
126             becc.back().emplace_back(st[tp]);
127         } while (st[tp--] != u);
128     }
129 }
130 int main() {
131     int n, m;
132     std::cin >> n >> m;
133     for (int i = 1, x, y; i <= m; ++i) {
134         std::cin >> x >> y;
135         G[x].emplace_back(y, i);
136         G[y].emplace_back(x, i);
137     }
138     for (int i = 0; i < n; ++i) {
139         if (!dfn[i]) tar(i, -1);
140     }
141     std::cout << becc.size() << '\n';
142     for (auto& v : becc) {
143         std::cout << v.size() << ' ';
144         for (int x : v) std::cout << x << ' ';
145         std::cout << '\n';
146     }
147 }

```

10.6 2-sat

给出 n 个集合, 每个集合有 2 个元素, 已知若干个数对 (a, b) , 表示 a 与 b 矛盾. 要从每个集合各选择一个元素, 判断能否一共选 n 个两两不矛盾的元素.

```

1 /* two sat */
2 auto 2sat = [&](int n, const vpi& v) -> vi {
3     /* tarjan */
4     vvi e(2 * n);
5     vi dfn(2 * n), low(2 * n), stk(2 * n), belong(2 * n);
6     int timestamp = 0, top = 0, scc_cnt = 0;
7     std::vector<bool> in_stk(2 * n);
8     auto tarjan = [&](auto&& self, int u) -> void {
9         dfn[u] = low[u] = ++timestamp;
10        stk[++top] = u;
11        in_stk[u] = true;
12        for (const auto& v : e[u]) {
13            if (!dfn[v]) {
14                self(self, v);
15                Min(low[u], low[v]);
16            } else if (in_stk[v]) {
17                Min(low[u], dfn[v]);
18            }
19        }
20        if (dfn[u] == low[u]) {
21            scc_cnt++;
22            int v;
23            do {
24                v = stk[top--];
25                in_stk[v] = false;
26                belong[v] = scc_cnt;
27            } while (v != u);
28        }
29    };
30    for (const auto& [a, b] : v) {
31        e[a].push_back(b ^ 1);
32        e[b].push_back(a ^ 1);
33    }
34    for (int i = 0; i < 2 * n; i++) {
35        if (!dfn[i]) tarjan(tarjan, i);
36    }
37 }

```

```

36     }
37     vi ans;
38     for (int i = 0; i < 2 * n; i += 2) {
39         if (belong[i] == belong[i + 1]) return vi{};
40         ans.push_back(belong[i] > belong[i + 1] ? i + 1 : i);
41     }
42     return ans;
43 };

```

上述将 i 与 $i + 1$ 作为一个集合里的元素, 编号为 0 至 $2n - 1$.

10.7 minimum ring

Floyd

```

1  /* minimum ring */
2  auto min_circle = [&]() -> int {
3      vvi dist(n + 1, vi(n + 1, inf));
4      for (int i = 1; i <= n; i++) {
5          for (int j = 1; j <= n; j++) {
6              Min(dist[i][j], g[i][j]);
7          }
8          dist[i][i] = 0;
9      }
10     for (int k = 1; k <= n; k++) {
11         for (int i = 1; i < k; i++) {
12             for (int j = 1; j < i; j++) {
13                 Min(ans, dist[i][j] + g[i][k] + g[k][j]);
14             }
15         }
16         for (int i = 1; i <= n; i++) {
17             for (int j = 1; j <= n; j++) {
18                 Min(dist[i][j], dist[i][k] + dist[k][j]);
19             }
20         }
21     }
22     return ans;
23 };

```

tree - diameter

10.8 tree - center of gravity

```

1  /* center of gravity */
2  int sum; /* 点权和 */
3  vi size(n + 1), weight(n + 1), w(n + 1), depth(n + 1);
4  std::array<int, 2> centroid = {0, 0};
5  auto get_centroid = [&](auto&& self, int u, int fa) -> void {
6      size[u] = w[u];
7      weight[u] = 0;
8      for (auto v : e[u]) {
9          if (v == fa) continue;
10         self(self, v, u);
11         size[u] += size[v];
12         Max(weight[u], size[v]);
13     }
14     Max(weight[u], sum - size[u]);
15     if (weight[u] <= sum / 2) {
16         centroid[centroid[0] != 0] = u;
17     }
18 };

```

10.9 tree - DSU on tree

给出一棵 n 个节点以 1 为根的树, 每个节点染上一种颜色, 询问以 u 为节点的子树中有多少种颜色.

```

1  // Problem: U41492 树上数颜色
2
3  int main() {

```

```

4   std::ios::sync_with_stdio(false);
5   std::cin.tie(0);
6   std::cout.tie(0);
7
8   int n, m, dfn = 0, cnttot = 0;
9   std::cin >> n;
10  vvi e(n + 1);
11  vi siz(n + 1), col(n + 1), son(n + 1), dfnl(n + 1), dfnr(n + 1), rank(n + 1);
12  vi ans(n + 1), cnt(n + 1);
13
14  for (int i = 1; i < n; i++) {
15      int u, v;
16      std::cin >> u >> v;
17      e[u].push_back(v);
18      e[v].push_back(u);
19  }
20  for (int i = 1; i <= n; i++) {
21      std::cin >> col[i];
22  }
23  auto add = [&](int u) -> void {
24      if (cnt[col[u]] == 0) cnttot++;
25      cnt[col[u]]++;
26  };
27  auto del = [&](int u) -> void {
28      cnt[col[u]]--;
29      if (cnt[col[u]] == 0) cnttot--;
30  };
31  auto dfs1 = [&](auto&& self, int u, int fa) -> void {
32      dfnl[u] = ++dfn;
33      rank[dfn] = u;
34      siz[u] = 1;
35      for (auto v : e[u]) {
36          if (v == fa) continue;
37          self(self, v, u);
38          siz[u] += siz[v];
39          if (!son[u] or siz[son[u]] < siz[v]) son[u] = v;
40      }
41      dfnr[u] = dfn;
42  };
43  auto dfs2 = [&](auto&& self, int u, int fa, bool op) -> void {
44      for (auto v : e[u]) {
45          if (v == fa or v == son[u]) continue;
46          self(self, v, u, false);
47      }
48      if (son[u]) self(self, son[u], u, true);
49      for (auto v : e[u]) {
50          if (v == fa or v == son[u]) continue;
51          rep(i, dfnl[v], dfnr[v]) { add(rank[i]); }
52      }
53      add(u);
54      ans[u] = cnttot;
55      if (op == false) {
56          rep(i, dfnl[u], dfnr[u]) { del(rank[i]); }
57      }
58  };
59  dfs1(dfs1, 1, 0);
60  dfs2(dfs2, 1, 0, false);
61  std::cin >> m;
62  for (int i = 1; i <= m; i++) {
63      int u;
64      std::cin >> u;
65      std::cout << ans[u] << endl;
66  }
67  return 0;
68 }

```

10.10 tree - AHU

```

1  /* AHU */
2  std::map<vi, int> mapple;
3  std::function<int(vvi&, int, int)> tree_hash = [&](vvi& e, int u, int fa) -> int {
4      vi code;
5      if (u == 0) code.push_back(-1);
6      for (auto v : e[u]) {
7          if (v == fa) continue;
8          code.push_back(tree_hash(e, v, u));
9      }
10     std::sort(all(code));
11     int id = mapple.size();
12     auto it = mapple.find(code);
13     if (it == mapple.end()) {
14         mapple[code] = id;

```

```

15     } else {
16         id = it->ss;
17     }
18     return id;
19 };

```

10.11 tree - LCA

```

1  /* LCA */
2  int B = 30;
3  vvi e(n + 1), fa(n + 1, vi(B));
4  vi dep(n + 1);
5  auto dfs = [&](auto&& self, int u) -> void {
6      for (auto v : e[u]) {
7          if (v == fa[u][0]) continue;
8          dep[v] = dep[u] + 1;
9          fa[v][0] = u;
10         self(self, v);
11     }
12 };
13 auto init = [&]() -> void {
14     dep[root] = 1;
15     dfs(dfs, root);
16     for (int j = 1; j < B; j++) {
17         for (int i = 1; i <= n; i++) {
18             fa[i][j] = fa[fa[i][j - 1]][j - 1];
19         }
20     }
21 };
22 init();
23 auto LCA = [&](int a, int b) -> int {
24     if (dep[a] > dep[b]) std::swap(a, b);
25     int d = dep[b] - dep[a];
26     for (int i = 0; (1 << i) <= d; i++) {
27         if (d & (1 << i)) b = fa[b][i];
28     }
29     if (a == b) return a;
30     for (int i = B - 1; i >= 0 and a != b; i--) {
31         if (fa[a][i] == fa[b][i]) continue;
32         a = fa[a][i];
33         b = fa[b][i];
34     }
35     return fa[a][0];
36 };
37 auto dist = [&](int a, int b) -> int { return dep[a] + dep[b] - dep[LCA(a, b)] * 2; };

```

10.12 tree - heavy light decomposition

对一棵有根树进行如下 4 种操作:

1. 1 $x y z$: 将节点 x 到节点 y 的最短路径上所有节点的值加上 z .
2. 2 $x y$: 查询节点 x 到节点 y 的最短路径上所有节点的值的和.
3. 3 $x z$: 将以节点 x 为根的子树上所有节点的值加上 z .
4. 4 x : 查询以节点 x 为根的子树上所有节点的值的和.

```

1  /* heavy light decomposition */
2  int cnt = 0;
3  vi son(n + 1), fa(n + 1), siz(n + 1), depth(n + 1);
4  vi dfn(n + 1), rank(n + 1), top(n + 1), botton(n + 1);
5  auto dfs1 = [&](auto&& self, int u) -> void {
6      son[u] = -1, siz[u] = 1;
7      for (auto v : e[u]) {
8          if (depth[v] != 0) continue;
9          depth[v] = depth[u] + 1;
10         fa[v] = u;
11         self(self, v);
12         siz[u] += siz[v];
13         if (son[u] == -1 or siz[v] > siz[son[u]]) son[u] = v;
14     }
15 };

```

```

16 auto dfs2 = [&](auto&& self, int u, int t) -> void {
17     top[u] = t;
18     dfn[u] = ++cnt;
19     rank[cnt] = u;
20     botton[u] = dfn[u];
21     if (son[u] == -1) return;
22     self(self, son[u], t);
23     Max(botton[u], botton[son[u]]);
24     for (auto v : e[u]) {
25         if (v != son[u] and v != fa[u]) {
26             self(self, v, v);
27             Max(botton[u], botton[v]);
28         }
29     }
30 };
31 depth[root] = 1;
32 dfs1(dfs1, root);
33 dfs2(dfs2, root, root);
34
35 /* 求 LCA */
36 auto LCA = [&](int a, int b) -> int {
37     while (top[a] != top[b]) {
38         if (depth[top[a]] < depth[top[b]]) std::swap(a, b);
39         a = fa[top[a]];
40     }
41     return (depth[a] > depth[b] ? b : a);
42 };
43
44 /* 维护 u 到 v 的路径 */
45 while (top[u] != top[v]) {
46     if (depth[top[u]] < depth[top[v]]) std::swap(u, v);
47     opt(dfn[top[u]], dfn[u]);
48     u = fa[top[u]];
49 }
50 if (dfn[u] > dfn[v]) std::swap(u, v);
51 opt(dfn[u], dfn[v]);
52
53 /* 维护 u 为根的子树 */
54 opt(dfn[u], botton[u]);
55
56 /*
57 线段树的 build() 函数中
58 if(l == r) tree[u] = {l, l, w[rank[l]], 0};
59 */
60
61 build(1, 1, n);
62 for (int i = 1; i <= m; i++) {
63     int op, u, v;
64     LL k;
65     std::cin >> op;
66     if (op == 1) {
67         std::cin >> u >> v >> k;
68         while (top[u] != top[v]) {
69             if (depth[top[u]] < depth[top[v]]) std::swap(u, v);
70             modify(1, dfn[top[u]], dfn[u], k);
71             u = fa[top[u]];
72         }
73         if (dfn[u] > dfn[v]) std::swap(u, v);
74         modify(1, dfn[u], dfn[v], k);
75     } else if (op == 2) {
76         std::cin >> u >> v;
77         LL ans = 0;
78         while (top[u] != top[v]) {
79             if (depth[top[u]] < depth[top[v]]) std::swap(u, v);
80             ans = (ans + query(1, dfn[top[u]], dfn[u])) % p;
81             u = fa[top[u]];
82         }
83         if (dfn[u] > dfn[v]) std::swap(u, v);
84         ans = (ans + query(1, dfn[u], dfn[v])) % p;
85         std::cout << ans << endl;
86     } else if (op == 3) {
87         std::cin >> u >> k;
88         modify(1, dfn[u], botton[u], k);
89     } else {
90         std::cin >> u;
91         std::cout << query(1, dfn[u], botton[u]) % p << endl;
92     }
93 }

```

10.13 tree - virtual tree

```
1 /* virtual tree */
```

```

2 | auto build_vtree = [&](vi ver) -> void {
3 |     std::sort(all(ver), [&](int x, int y) { return dfn[x] < dfn[y]; });
4 |     vi stk = {1};
5 |     for (auto v : ver) {
6 |         int u = stk.back();
7 |         int lca = LCA(v, u);
8 |         if (lca != u) {
9 |             while (dfn[lca] < dfn[stk.end()[-2]]) {
10 |                 g[stk.end()[-2]].push_back(stk.back());
11 |                 stk.pop_back();
12 |             }
13 |             u = stk.back();
14 |             if (dfn[lca] != dfn[stk.end()[-2]]) {
15 |                 g[lca].push_back(u);
16 |                 stk.pop_back();
17 |                 stk.push_back(lca);
18 |             } else {
19 |                 g[lca].push_back(u);
20 |                 stk.pop_back();
21 |             }
22 |         }
23 |         stk.push_back(v);
24 |     }
25 |     while (stk.size() > 1) {
26 |         int u = stk.end()[-2];
27 |         int v = stk.back();
28 |         g[u].push_back(v);
29 |         stk.pop_back();
30 |     }
31 | };

```

10.14 tree - pseudo tree

```

1 | /* ring detection (directed) */
2 | vi vis(n + 1), fa(n + 1), ring;
3 | auto dfs = [&](auto&& self, int u) -> bool {
4 |     vis[u] = 1;
5 |     for (const auto& v : e[u]) {
6 |         if (!vis[v]) {
7 |             fa[v] = u;
8 |             if (self(self, v)) {
9 |                 return true;
10 |             }
11 |         } else if (vis[v] == 1) {
12 |             ring.push_back(v);
13 |             for (auto x = u; x != v; x = fa[x]) {
14 |                 ring.push_back(x);
15 |             }
16 |             reverse(all(ring));
17 |             return true;
18 |         }
19 |     }
20 |     vis[u] = 2;
21 |     return false;
22 | };
23 | for (int i = 1; i <= n; i++) {
24 |     if (!vis[i]) {
25 |         if (dfs(dfs, i)) {
26 |             // operations //
27 |         }
28 |     }
29 | }
30 |
31 | /* cycle detection (undirected) */
32 | vi vis(n + 1), ring;
33 | vpi fa(n + 1);
34 | auto dfs = [&](auto&& self, int u, int from) -> bool {
35 |     vis[u] = 1;
36 |     for (const auto& [v, id] : e[u]) {
37 |         if (id == from) continue;
38 |         if (!vis[v]) {
39 |             fa[v] = {u, id};
40 |             if (self(self, v, id)) {
41 |                 return true;
42 |             }
43 |         } else if (vis[v] == 1) {
44 |             ring.push_back(v);
45 |             for (auto x = u; x != v; x = fa[x].ff) {
46 |                 ring.push_back(x);
47 |             }
48 |             return true;
49 |         }

```

```

50     }
51     vis[u] = 2;
52     return false;
53 };
54 for (int i = 1; i <= n; i++) {
55     if (!vis[i]) {
56         if (dfs(dfs, i, 0)) {
57             // operations //
58         }
59     }
60 }

```

10.15 tree - divide and conquer on tree

点分治

第一个题

一棵 $n \leq 10^4$ 个点的树，边权 $w \leq 10^4$ 。 $m \leq 100$ 次询问树上是否存在长度为 $k \leq 10^7$ 的路径。

```

1 // 洛谷 P3806 【模板】点分治1
2
3 int main() {
4     std::ios::sync_with_stdio(false);
5     std::cin.tie(0);
6     std::cout.tie(0);
7
8     int n, m, k;
9     std::cin >> n >> m;
10
11     std::vector<vpi> e(n + 1);
12     std::map<int, PII> mp;
13
14     for (int i = 1; i < n; i++) {
15         int u, v, w;
16         std::cin >> u >> v >> w;
17         e[u].emplace_back(v, w);
18         e[v].emplace_back(u, w);
19     }
20     for (int i = 1; i <= m; i++) {
21         std::cin >> k;
22         mp[i] = {k, 0};
23     }
24
25     /* centroid decomposition */
26     int top1 = 0, top2 = 0, root;
27     vi len1(n + 1), len2(n + 1), vis(n + 1);
28     static std::array<int, 20000010> cnt;
29
30     std::function<int(int, int)> get_size = [&](int u, int fa) -> int {
31         if (vis[u]) return 0;
32         int ans = 1;
33         for (auto [v, w] : e[u]) {
34             if (v == fa) continue;
35             ans += get_size(v, u);
36         }
37         return ans;
38     };
39
40     std::function<int(int, int, int, int)> get_root = [&](int u, int fa, int tot,
41                                                         int& root) -> int {
42         if (vis[u]) return 0;
43         int sum = 1, maxx = 0;
44         for (auto [v, w] : e[u]) {
45             if (v == fa) continue;
46             int tmp = get_root(v, u, tot, root);
47             Max(maxx, tmp);
48             sum += tmp;
49         }
50         Max(maxx, tot - sum);
51         if (2 * maxx <= tot) root = u;
52         return sum;
53     };
54
55     std::function<void(int, int, int)> get_dist = [&](int u, int fa, int dist) -> void {
56         if (dist <= 10000000) len1[++top1] = dist;
57         for (auto [v, w] : e[u]) {
58             if (v == fa or vis[v]) continue;
59             get_dist(v, u, dist + w);
60         }

```

```

61     };
62
63     auto solve = [&](int u, int dist) -> void {
64         top2 = 0;
65         for (auto [v, w] : e[u]) {
66             if (vis[v]) continue;
67             top1 = 0;
68             get_dist(v, u, w);
69             for (int i = 1; i <= top1; i++) {
70                 for (int tt = 1; tt <= m; tt++) {
71                     int k = mp[tt].ff;
72                     if (k >= len1[i]) mp[tt].ss |= cnt[k - len1[i]];
73                 }
74             }
75             for (int i = 1; i <= top1; i++) {
76                 len2[++top2] = len1[i];
77                 cnt[len1[i]] = 1;
78             }
79         }
80         for (int i = 1; i <= top2; i++) cnt[len2[i]] = 0;
81     };
82
83     std::function<void(int)> divide = [&](int u) -> void {
84         vis[u] = cnt[0] = 1;
85         solve(u, 0);
86         for (auto [v, w] : e[u]) {
87             if (vis[v]) continue;
88             get_root(v, u, get_size(v, u), root);
89             divide(root);
90         }
91     };
92
93     get_root(1, 0, get_size(1, 0), root);
94     divide(root);
95
96     for (int i = 1; i <= m; i++) {
97         if (mp[i].ss == 0) {
98             std::cout << "NAY" << endl;
99         } else {
100             std::cout << "AYE" << endl;
101         }
102     }
103
104     return 0;
105 }

```

第二个题

一棵 $n \leq 4 \times 10^4$ 个点的树, 边权 $w \leq 10^3$. 询问树上长度不超过 $k \leq 2 \times 10^4$ 的路径的数量.

```

1 // 洛谷 P4178 Tree
2
3 int main() {
4     std::ios::sync_with_stdio(false);
5     std::cin.tie(0);
6     std::cout.tie(0);
7
8     int n, k;
9     std::cin >> n;
10    std::vector<vpi> e(n + 1);
11    for (int i = 1; i < n; i++) {
12        int u, v, w;
13        std::cin >> u >> v >> w;
14        e[u].emplace_back(v, w);
15        e[v].emplace_back(u, w);
16    }
17    std::cin >> k;
18
19    /* centroid decomposition */
20    int root;
21    vi len, vis(n + 1);
22
23    std::function<int(int, int)> get_size = [&](int u, int fa) -> int {
24        if (vis[u]) return 0;
25        int ans = 1;
26        for (auto [v, w] : e[u]) {
27            if (v == fa) continue;
28            ans += get_size(v, u);
29        }
30        return ans;
31    };
32
33    std::function<int(int, int, int, int)> get_root = [&](int u, int fa, int tot,
34        int& root) -> int {
35        if (vis[u]) return 0;

```



```

36     int sum = 1, maxx = 0;
37     for (auto [v, w] : e[u]) {
38         if (v == fa) continue;
39         int tmp = get_root(v, u, tot, root);
40         maxx = std::max(maxx, tmp);
41         sum += tmp;
42     }
43     maxx = std::max(maxx, tot - sum);
44     if (2 * maxx <= tot) root = u;
45     return sum;
46 };
47
48 std::function<void(int, int, int)> get_dist = [&](int u, int fa, int dist) -> void {
49     len.push_back(dist);
50     for (auto [v, w] : e[u]) {
51         if (v == fa || vis[v]) continue;
52         get_dist(v, u, dist + w);
53     }
54 };
55
56 auto solve = [&](int u, int dist) -> int {
57     len.clear();
58     get_dist(u, 0, dist);
59     std::sort(all(len));
60     int ans = 0;
61     for (int l = 0, r = len.size() - 1; l < r;) {
62         if (len[l] + len[r] <= k) {
63             ans += r - l++;
64         } else {
65             r--;
66         }
67     }
68     return ans;
69 };
70
71 std::function<int(int)> divide = [&](int u) -> int {
72     vis[u] = true;
73     int ans = solve(u, 0);
74     for (auto [v, w] : e[u]) {
75         if (vis[v]) continue;
76         ans -= solve(v, w);
77         get_root(v, u, get_size(v, u), root);
78         ans += divide(root);
79     }
80     return ans;
81 };
82
83 get_root(1, 0, get_size(1, 0), root);
84 std::cout << divide(root) << endl;
85
86 return 0;
87 }

```

10.16 tree - matrix tree

```

1  const int N=33,M=152599,P=998244353;
2  int qpow(int a,int b=P-2){
3      int r=1;for(;b;b>>=1,a=1ll*a*a%P)if(b&1)r=1ll*r*a%P;return r;
4  }
5  struct T{int x,y,z;T(int a=0,int b=0,int c=0):x(a),y(b),z(c){}}e[N*N];
6  struct F{
7      int a,b;
8      F():a(),b(){}
9      F(int x,int y):a(x),b(y){}
10     F operator+(const F&_)const{return F((a+_a)%P,(b+_b)%P);}
11     F operator+=(const F&_)const{return *this=*this+_;}
12     F operator-(const F&_)const{return F((a-_a+P)%P,(b-_b+P)%P);}
13     F operator-=(const F&_)const{return *this=*this-_;}
14     F operator*(const F&_)const{return F((1ll*a*_a+1ll*b*_b)%P,1ll*b*_b%P);}
15     F operator*=(const F&_)const{return *this=*this*_;}
16     int operator&()const{return b?2:(a?1:0);}
17     bool operator!()const{return !a&&!b;}
18     F operator~()const{
19         int d=qpow(b);
20         return F((P-1ll*a*d%P*d%P)%P,d);
21     }
22 };
23 int fa[N],phi[M],n,m;
24 int gf(int x){return x==fa[x]?x:fa[x]=gf(fa[x]);}
25 int cal(int p){
26     F a[N][N],d,iv,z=F(0,1);
27     int i,j,k,l,x=0;iota(fa,fa+n+1,0);

```

```

28     for(i=1;i<=m;++i)if(e[i].z%p==0){
29         if((j=gf(e[i].x))!=(k=gf(e[i].y)))fa[j]=k;
30         j=e[i].x,k=e[i].y,l=e[i].z,++x;
31         a[j][k]-=F(1,1),a[k][j]-=F(1,1);
32         a[j][j]+=F(1,1),a[k][k]+=F(1,1);
33     }
34     for(j=0,i=1;i<=n;++i)if(fa[i]==i)++j;
35     if(j>1 || x<n-1)return 0;
36     for(i=1;i<=n;++i){
37         for(k=i,j=i+1;j<=n;++j)if(&a[j][i]>&a[k][i])k=j;
38         if(k!=i)swap(a[i],a[k]),z*=F(0,P-1);
39         if(!a[i][i])return 0;
40         for(z*=a[i][i][i],iv=-a[i][i],j=i;j<=n;++j)a[i][j]*=iv;
41         for(j=i+1;j<=n;++j)for(d=a[j][i],k=i;k<=n;++k)a[j][k]-=a[i][k]*d;
42     }
43     return z.a;
44 }
45 void work(){
46     int h=0,i,j,x,y,z;
47     for(cin>>n>>m,i=1;i<=m;++i)cin>>x>>y>>z,e[i]=T(x,y,z),h=max(h,z);
48     iota(phi+1,phi+h+1,1);
49     for(i=1;i<=h;++i)for(j=i<=1;j<=h;j+=i)phi[j]=(phi[j]-phi[i]+P)%P;
50     for(z=0,i=1;i<=h;++i)z=(z+1ll*phi[i]*cal(i)%P)%P;
51     cout<<z<<'n';
52 }

```

10.17 Prefür sequence

```

1  \* prefür @ wrb *\
2  for(int i=1;i<=n;i++)cin>>fa[i],d[fa[i]]++;
3  for(int i=1,j=1;i<=n-1;i++,j++){
4      while(d[j])j++;
5      p[i]=fa[j];
6      while(i<=n-1&&!--d[p[i]]&&j>p[i])p[i+1]=fa[p[i]],i++;
7  }
8
9  //////////////////////////////////////
10
11 for(int i=1;i<=n-1;i++)cin>>p[i],d[p[i]]++;
12 p[n-1]=n;
13 for(int i=1,j=1;i<=n;i++,j++){
14     while(d[j])j++;
15     fa[j]=p[i];
16     while(i<=n&&!--d[p[i]]&&j>p[i])fa[p[i]]=p[i+1],i++;
17 }

```

10.18 network flow - maximal flow

Dinic

```

1  /* dinic */
2  struct edge {
3      int from, to;
4      LL cap, flow;
5  };
6  edge(int u, int v, LL c, LL f) : from(u), to(v), cap(c), flow(f) {}
7  };
8
9  struct Dinic {
10     int n, m = 0, s, t;
11     std::vector<edge> e;
12     vi g[N];
13     int d[N], cur[N], vis[N];
14
15     void init(int n) {
16         for (int i = 0; i < n; i++) g[i].clear();
17         e.clear();
18         m = 0;
19     }
20
21     void add(int from, int to, LL cap) {
22         e.push_back(edge(from, to, cap, 0));
23         e.push_back(edge(to, from, 0, 0));
24         g[from].push_back(m++);
25         g[to].push_back(m++);
26     }

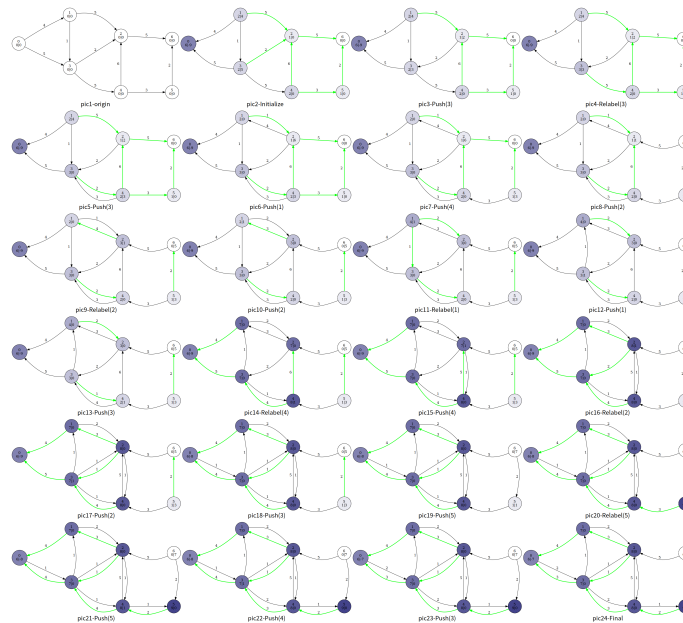
```

```

27
28 bool bfs() {
29     for (int i = 1; i <= n; i++) {
30         vis[i] = 0;
31     }
32     std::queue<int> q;
33     q.push(s), d[s] = 0, vis[s] = 1;
34     while (!q.empty()) {
35         int u = q.front();
36         q.pop();
37         for (int i = 0; i < g[u].size(); i++) {
38             edge& ee = e[g[u][i]];
39             if (!vis[ee.to] and ee.cap > ee.flow) {
40                 vis[ee.to] = 1;
41                 d[ee.to] = d[u] + 1;
42                 q.push(ee.to);
43             }
44         }
45     }
46     return vis[t];
47 }
48
49 LL dfs(int u, LL now) {
50     if (u == t || now == 0) return now;
51     LL flow = 0, f;
52     for (int& i = cur[u]; i < g[u].size(); i++) {
53         edge& ee = e[g[u][i]];
54         edge& er = e[g[u][i] ^ 1];
55         if (d[u] + 1 == d[ee.to] and (f = dfs(ee.to, std::min(now, ee.cap - ee.flow))) > 0) {
56             ee.flow += f, er.flow -= f;
57             flow += f, now -= f;
58             if (now == 0) break;
59         }
60     }
61     return flow;
62 }
63
64 LL dinic() {
65     LL ans = 0;
66     while (bfs()) {
67         for (int i = 1; i <= n; i++) cur[i] = 0;
68         ans += dfs(s, INF);
69     }
70     return ans;
71 }
72 } maxf;

```

HLPP



```

1  /* hlpp */
2  struct HLPP {
3      int n, m = 0, s, t;
4      std::vector<edge> e;    /* 边 */
5      std::vector<node> nd;   /* 点 */

```

```

6  std::vector<int> g[N];    /* 点的连边编号 */
7  std::priority_queue<node> q;
8  std::queue<int> qq;
9  bool vis[N];
10 int cnt[N];
11
12 void init() {
13     e.clear();
14     nd.clear();
15     for (int i = 0; i <= n + 1; i++) {
16         nd.pushback(node(inf, i, 0));
17         g[i].clear();
18         vis[i] = false;
19     }
20 }
21
22 void add(int u, int v, LL w) {
23     e.pushback(edge(u, v, w));
24     e.pushback(edge(v, u, 0));
25     g[u].pushback(m++);
26     g[v].pushback(m++);
27 }
28
29 void bfs() {
30     nd[t].hight = 0;
31     qq.push(t);
32     while (!qq.empty()) {
33         int u = qq.front();
34         qq.pop();
35         vis[u] = false;
36         for (auto j : g[u]) {
37             int v = e[j].to;
38             if (e[j].cap == 0 && nd[v].hight > nd[u].hight + 1) {
39                 nd[v].hight = nd[u].hight + 1;
40                 if (vis[v] == false) {
41                     qq.push(v);
42                     vis[v] = true;
43                 }
44             }
45         }
46     }
47     return;
48 }
49
50 void _push(int u) {
51     for (auto j : g[u]) {
52         edge &ee = e[j], &er = e[j ^ 1];
53         int v = ee.to;
54         node &nu = nd[u], &nv = nd[v];
55         if (ee.cap && nv.hight + 1 == nu.hight) {
56             LL flow = std::min(ee.cap, nu.flow);
57             ee.cap -= flow, er.cap += flow;
58             nu.flow -= flow, nv.flow += flow;
59             if (vis[v] == false && v != t && v != s) {
60                 q.push(nv);
61                 vis[v] = true;
62             }
63             if (nu.flow == 0) break;
64         }
65     }
66 }
67
68 void relabel(int u) {
69     nd[u].hight = inf;
70     for (auto j : g[u]) {
71         int v = e[j].to;
72         if (e[j].cap && nd[v].hight + 1 < nd[u].hight) {
73             nd[u].hight = nd[v].hight + 1;
74         }
75     }
76 }
77
78 LL hlpp() {
79     bfs();
80     if (nd[s].hight == inf) return 0;
81     nd[s].hight = n;
82     for (int i = 1; i <= n; i++) {
83         if (nd[i].hight < inf) cnt[nd[i].hight]++;
84     }
85     for (auto j : g[s]) {
86         int v = e[j].to;
87         int flow = e[j].cap;
88         if (flow) {
89             e[j].cap -= flow, e[j ^ 1].cap += flow;
90             nd[s].flow -= flow, nd[v].flow += flow;
91             if (vis[v] == false && v != s && v != t) {
92                 q.push(nd[v]);

```

```

93         vis[v] = true;
94     }
95 }
96 }
97 while (!q.empty()) {
98     int u = q.top().id;
99     q.pop();
100     vis[u] = false;
101     _push(u);
102     if (nd[u].flow) {
103         cnt[nd[u].hight]--;
104         if (cnt[nd[u].hight] == 0) {
105             for (int i = 1; i <= n; i++) {
106                 if (i != s && i != t && nd[i].hight > nd[u].hight && nd[i].hight < n + 1) {
107                     nd[i].hight = n + 1;
108                 }
109             }
110         }
111         relabel(u);
112         cnt[nd[u].hight]++;
113         q.push(nd[u]);
114         vis[u] = true;
115     }
116 }
117 return nd[t].flow;
118 }
119 } maxf;

```

10.19 network flow - minimum cost flow

Dinic + SPFA

```

1  /* Dinic + SPFA */
2  struct edge {
3      int from, to;
4      LL cap, cost;
5  };
6      edge(int u, int v, LL c, LL w) : from(u), to(v), cap(c), cost(w) {}
7  };
8
9  const int N = 2000;
10
11  struct MCMF {
12      int n, m = 0, s, t;
13      std::vector<edge> e;
14      vi g[N];
15      int cur[N], vis[N];
16      LL dist[N], minc;
17
18      void init(int n) {
19          for (int i = 0; i < n; i++) g[i].clear();
20          e.clear();
21          minc = m = 0;
22      }
23
24      void add(int from, int to, LL cap, LL cost) {
25          e.push_back(edge(from, to, cap, cost));
26          e.push_back(edge(to, from, 0, -cost));
27          g[from].push_back(m++);
28          g[to].push_back(m++);
29      }
30
31      bool spfa() {
32          for (int i = 1; i <= n; i++) {
33              dist[i] = INF, cur[i] = 0;
34          }
35          std::queue<int> q;
36          q.push(s), dist[s] = 0, vis[s] = 1;
37          while (!q.empty()) {
38              int u = q.front();
39              q.pop();
40              vis[u] = 0;
41              for (int j = cur[u]; j < g[u].size(); j++) {
42                  edge& ee = e[g[u][j]];
43                  int v = ee.to;
44                  if (ee.cap && dist[v] > dist[u] + ee.cost) {
45                      dist[v] = dist[u] + ee.cost;
46                      if (!vis[v]) {
47                          q.push(v);
48                          vis[v] = 1;
49                      }

```

```

50     }
51 }
52 }
53 return dist[t] != INF;
54 }
55
56 LL dfs(int u, LL now) {
57     if (u == t) return now;
58     vis[u] = 1;
59     LL ans = 0;
60     for (int& i = cur[u]; i < g[u].size() && ans < now; i++) {
61         edge &ee = e[g[u][i]], &er = e[g[u][i] ^ 1];
62         int v = ee.to;
63         if (!vis[v] && ee.cap && dist[v] == dist[u] + ee.cost) {
64             LL f = dfs(v, std::min(ee.cap, now - ans));
65             if (f) {
66                 minc += f * ee.cost, ans += f;
67                 ee.cap -= f;
68                 er.cap += f;
69             }
70         }
71     }
72     vis[u] = 0;
73     return ans;
74 }
75
76 PLL mcmf() {
77     LL maxf = 0;
78     while (spfa()) {
79         LL tmp;
80         while ((tmp = dfs(s, INF))) maxf += tmp;
81     }
82     return std::make_pair(maxf, minc);
83 }
84 } minc_maxf;

```

Primal-Dual 原始对偶算法

```

1  /* primal dual */
2  struct edge {
3      int from, to;
4      LL cap, cost;
5  };
6  edge(int u, int v, LL c, LL w) : from(u), to(v), cap(c), cost(w) {}
7  };
8
9  struct node {
10     int v, e;
11 };
12 node(int _v = 0, int _e = 0) : v(_v), e(_e) {}
13 };
14
15 const int maxn = 5000 + 10;
16
17 struct MCMF {
18     int n, m = 0, s, t;
19     std::vector<edge> e;
20     vi g[maxn];
21     int vis[maxn];
22     LL dis[maxn], h[maxn];
23     node p[maxn * 2];
24
25     void add(int from, int to, LL cap, LL cost) {
26         e.push_back(edge(from, to, cap, cost));
27         e.push_back(edge(to, from, 0, -cost));
28         g[from].push_back(m++);
29         g[to].push_back(m++);
30     }
31
32     bool dijkstra() {
33         std::priority_queue<PLL, std::vector<PLL>, std::greater<PLL>> q;
34         for (int i = 1; i <= n; i++) {
35             dis[i] = INF;
36             vis[i] = 0;
37         }
38         dis[s] = 0;
39         q.push({0, s});
40         while (!q.empty()) {
41             auto u = q.top().ss;
42             q.pop();
43             if (vis[u]) continue;
44             vis[u] = 1;
45             for (auto i : g[u]) {

```

```

46         edge ee = e[i];
47         int v = ee.to;
48         LL nc = ee.cost + h[u] - h[v];
49         if (ee.cap and dis[v] > dis[u] + nc) {
50             dis[v] = dis[u] + nc;
51             p[v] = node(u, i);
52             if (!vis[v]) q.push({dis[v], v});
53         }
54     }
55 }
56 return dis[t] != INF;
57 }
58
59 void spfa() {
60     std::queue<int> q;
61     for (int i = 1; i <= n; i++) h[i] = INF;
62     h[s] = 0, vis[s] = 1;
63     q.push(s);
64     while (!q.empty()) {
65         int u = q.front();
66         q.pop();
67         vis[u] = 0;
68         for (auto i : g[u]) {
69             edge ee = e[i];
70             int v = ee.to;
71             if (ee.cap and h[v] > h[u] + ee.cost) {
72                 h[v] = h[u] + ee.cost;
73                 if (!vis[v]) {
74                     vis[v] = 1;
75                     q.push(v);
76                 }
77             }
78         }
79     }
80 }
81
82 PLL mcmf() {
83     LL maxf = 0, minc = 0;
84     spfa();
85     while (dijkstra()) {
86         LL minf = INF;
87         for (int i = 1; i <= n; i++) h[i] += dis[i];
88         for (int i = t; i != s; i = p[i].v) minf = std::min(minf, e[p[i].e].cap);
89         for (int i = t; i != s; i = p[i].v) {
90             e[p[i].e].cap -= minf;
91             e[p[i].e ^ 1].cap += minf;
92         }
93         maxf += minf;
94         minc += minf * h[t];
95     }
96     return std::make_pair(maxf, minc);
97 }
98 } minc_maxf;

```

存在负环的网络

流满后推流, 转化为上下界网络流.

10.20 network flow - minimal cut

最小割解决的问题是将图中的点集 V 划分成 S 与 T , 使得 S 与 T 之间的连边的容量总和最小.

最大流最小割定理

网络中 s 到 t 的最大流流量的值等于所要求的最小割的值, 所以求最小割只需要跑 Dinic 即可.

获得 S 中的所有点

在 Dinic 的 bfs 函数中, 每次将所有点的 d 数组值改为无穷大, 最后跑完最大流之后 d 数组不为无穷大的就是和源点一起在 S 集合中的点.

例子

最小割的本质是对图中点集进行 2-划分, 网络流只是求解答案的手段.

1. 在图中花费最小的代价断开一些边使得源点 s 无法流到汇点 t .

直接跑最大流就得到了答案.

2. 在图中删除最少的点使得源点 s 无法流到汇点 t .

对每个点进行拆点, 在 i 与 i' 之间建立容量为 1 的有向边.

10.21 network flow - upper / lower bound

无源汇上下界可行流

每条有向边有流量的上下界限制, 但整张图并未确定源点与汇点. 如果存在满足每个点的流入量等于流出量, 且每条边的流量满足其上下界限制的流, 称之为可行流.

1. 将每条边先给予大小为下界的流量,
2. 对每个点计算总流入量 in_u 与总流出量 out_u 的值,
3. 建立超级源点到每个点, 容量大小为 $\max\{0, \text{in}_u - \text{out}_u\}$ 的边; 建立每个点到超级汇点, 容量大小为 $\max\{0, \text{out}_u - \text{in}_u\}$,
4. 跑从超级源点到超级汇点的最大流, 如果超级源点每条边都流满意味着存在可行流. 将每条边的流量加上预先给每条边设置的下界流量即为可行流方案.

有源汇上下界可行流

1. 建立汇点 t 到源点 s 的, 容量为 ∞ 的有向边, 将其转化为无源汇的问题.

有源汇上下界最大流

1. 建立汇点 t 到源点 s 的, 容量为 ∞ 的有向边, 将其转化为无源汇的问题,
2. 跑上下界可行流, 可行流流量为边 $t \xrightarrow{\infty} s$ 的流量.
3. 删除 $t \xrightarrow{\infty} s$ 的边, 再残量网络上跑 s 到 t 的最大流,
4. 答案等于可行流流量 + 最大流流量.

有源汇上下界最小流

1. 建立汇点 t 到源点 s 的, 容量为 ∞ 的有向边, 将其转化为无源汇的问题,
2. 跑上下界可行流, 可行流流量为边 $t \xrightarrow{\infty} s$ 的流量.
3. 删除 $t \xrightarrow{\infty} s$ 的边, 再残量网络上跑 t 到 s 的最大流,
4. 答案等于可行流流量 - 最大流流量.

有源汇上下界最小费用可行流

1. 按下界流满并计算费用,
2. 类似有源汇上下界最大流建图, 跑超级源点到超级汇点的费用流,
3. 答案等于按下界的费用加上后续残量网络.

10.22 network flow - other versions

```

1  /* dinic @ wrb */
2  template<typename T, T inf = numeric_limits<T>::max()>
3  struct Max_Flow {
4      vector<int> he, cur, d, ne, to;
5      vector<T> c;
6      int s, t;
7      Max_Flow(int m) : he(m, -1), s(-1), t(-1) {}
8      void add(int x, int y, T z = inf, T w = 0) {
9          // cerr << x << ' ' << y << ' ';
10         // if (z == inf) cerr << "inf\n";
11         // else cerr << z << '\n';
12         ne.emplace_back(he[x]);
13         he[x] = ne.size() - 1;
14         to.emplace_back(y);
15         c.emplace_back(z);
16         ne.emplace_back(he[y]);
17         he[y] = ne.size() - 1;
18         to.emplace_back(x);
19         c.emplace_back(w);
20     }
21     int bfs() {
22         queue<int> q;
23         d.assign(he.size(), -1);
24         q.emplace(s), d[s] = 0;
25         for (; q.size(); q.pop()) {
26             int u = q.front(), v;
27             for (int i = he[u]; ~i; i = ne[i]) {
28                 if (c[i] && d[v = to[i]] == -1) {
29                     d[v] = d[u] + 1;
30                     if (v == t) return 1;
31                     q.emplace(v);
32                 }
33             }
34         }
35         return 0;
36     };
37     T dfs(int u, T fl) {
38         if (u == t) return fl;
39         T z = 0, r;
40         for (int& i = cur[u], v; ~i; i = ne[i]) {
41             if (c[i] && d[v = to[i]] == d[u] + 1) {
42                 r = dfs(v, min(fl, c[i]));
43                 if (r == 0) d[v] = -1;
44                 else {
45                     fl -= r, z += r, c[i] -= r, c[i ^ 1] += r;
46                     if (fl == 0) return z;
47                 }
48             }
49         }
50         return z;
51     };
52     T dinic(int _s, int _t) {
53         T z = 0;
54         for (s = _s, t = _t; bfs();) {
55             cur = he, z += dfs(s, inf);
56         }
57         return z;
58     };
59 };

```

```

1  /* bounded flow @ lys */
2  #include <iostream>
3  #include <cstdio>
4  #include <algorithm>
5  #include <queue>
6  #include <vector>
7  #define int long long
8  using namespace std;
9

```

```

10 const int maxn = 50020;
11 const int inf = 1e18;
12
13 struct Dinic_limit
14 {
15     int st, sgn; // st = 1 表示有源汇; sgn 表示最大(1)最小(-1)流
16     struct edge
17     {
18         int x, y, cap, flow, cost;
19     };
20     int deg[maxn]; // rd - cd
21     vector<int> e[maxn];
22     vector<edge> edges;
23     int mx;
24     int mcmf;
25     void add(int x, int y, int cap, int cost)
26     {
27         edges.push_back({x, y, cap, 0, cost});
28         edges.push_back({y, x, 0, 0, -cost});
29         mx = max({mx, x, y});
30         int m = edges.size();
31         e[x].push_back(m - 2), e[y].push_back(m - 1);
32     }
33     void add(int x, int y, int l, int r, int cost)
34     {
35         if (cost >= 0)
36             add(x, y, r - l, cost), deg[y] += l, deg[x] -= l, mcmf += l * cost;
37         else
38             add(y, x, r - l, -cost), deg[y] += r, deg[x] -= r, mcmf += r * cost;
39     }
40     int s, t;
41     int vis[maxn], dis[maxn];
42     bool spfa()
43     {
44         queue<int> q;
45         fill(vis, vis + mx + 1, 0), fill(dis, dis + mx + 1, inf);
46         dis[s] = 0, q.push(s), vis[s] = 1;
47         while (!q.empty())
48         {
49             int x = q.front();
50             q.pop(), vis[x] = 0;
51             for (int i : e[x])
52             {
53                 auto k = edges[i];
54                 if (k.cap - k.flow > 0 && k.cost + dis[x] < dis[k.y])
55                 {
56                     dis[k.y] = dis[x] + k.cost;
57                     if (!vis[k.y])
58                         q.push(k.y), vis[k.y] = 1;
59                 }
60             }
61         }
62         return dis[t] != inf;
63     }
64     int cur[maxn];
65     int dfs(int x, int lim)
66     {
67         if (x == t || lim == 0)
68             return lim;
69         vis[x] = 1;
70         int res = 0, f;
71         for (int &i = cur[x]; i < (int)e[x].size(); i++)
72         {
73             auto &k = edges[e[x][i]];
74             if (!vis[k.y] && k.cost + dis[x] == dis[k.y] && (f = dfs(k.y, min(lim, k.cap - k.flow))))
75                 res += f, lim -= f, k.flow += f, edges[e[x][i] ^ 1].flow -= f, mcmf += f * k.cost;
76             if (lim == 0)
77                 break;
78         }
79         vis[x] = 0;
80         return res;
81     }
82     int dinic(int s_, int t_)
83     {
84         int ss = mx + 1, tt = ss + 1;
85         int tot = 0;
86         for (int i = 1; i <= mx; i++)
87             if (deg[i] > 0)
88                 add(ss, i, deg[i], 0), tot += deg[i];
89             else if (deg[i] < 0)
90                 add(i, tt, -deg[i], 0);
91         if (st)
92             add(t_, s_, 0, inf, 0);
93         s = ss, t = tt;
94         int res = 0;
95         while (spfa())
96             fill(cur, cur + mx + 1, 0), res += dfs(s, inf);

```

```

97     // cerr << res << " " << tot << endl;
98     if (res != tot)
99         return -1;
100    if (st == 0)
101        return 1;
102    res = -edges.back().flow;
103    edges.back().cap = edges.back().flow = 0;
104    edges[edges.size() - 2].cap = edges[edges.size() - 2].flow = 0;
105    s = s_, t = t_;
106    if (sgn == -1)
107        swap(s, t);
108    while (spfa())
109        fill(cur, cur + mx + 1, 0), res += sgn * dfs(s, inf);
110    return res;
111 }
112 void clear()
113 {
114     for (int i = 0; i <= mx; i++)
115         e[i].clear(), deg[i] = 0;
116     edges.clear();
117     mx = 0, mcmf = 0;
118 }
119 Dinic_limit(int st_ = 1, int sgn_ = 1) { st = st_, sgn = sgn_; }
120 // st = 1 表示有源汇; sgn 表示最大(1)最小(-1)流
121 // 使用时调用 dinic 函数, 返回-1表示无解, 否则返回最大/最小流
122 };
123
124 Dinic_limit G(1, 1);
125
126 signed main()
127 {
128     ios::sync_with_stdio(false), cin.tie(0);
129     int n, m, S, T;
130     cin >> n >> m >> S >> T;
131     for (int i = 0; i < m; i++)
132     {
133         int s, t, l, r, c;
134         cin >> s >> t >> l >> r >> c;
135         G.add(s, t, l, r, c);
136     }
137     int res = G.dinic(S, T);
138     if (res == -1)
139         cout << -1 << endl;
140     else
141     {
142         cout << res << " " << G.mcmf << endl;
143     }
144 }

```

10.23 matching - matching on bipartite graph

二分图最大匹配

Kuhn-Munkres

时间复杂度: $O(n^3)$.

```

1  /* Kuhn-Munkres */
2  auto KM = [&](int n1, int n2, vvi e) -> std::pair<vi, vi> {
3      vi vis(n2 + 1);
4      vi l(n1 + 1, -1), r(n2 + 1, -1);
5      std::function<bool(int)> dfs = [&](int u) -> bool {
6          for (auto v : e[u]) {
7              if (!vis[v]) {
8                  vis[v] = 1;
9                  if (r[v] == -1 or dfs(r[v])) {
10                     r[v] = u;
11                     return true;
12                 }
13             }
14         }
15         return false;
16     };
17     for (int i = 1; i <= n1; i++) {
18         std::fill(all(vis), 0);
19         dfs(i);
20     }
21     for (int i = 1; i <= n2; i++) {
22         if (r[i] == -1) continue;

```

```

23     l[r[i]] = i;
24 }
25 return {l, r};
26 };
27 auto [mchl, mchr] = KM(n1, n2, e);
28 std::cout << mchl.size() - std::count(all(mchl), -1) << endl;

```

Hopcroft-Karp

据说时间复杂度是 $O(m\sqrt{n})$ 的, 但是快的飞起.

```

1  /* Hopcroft-Karp */
2  vpi e(m);
3  auto hopcroft_karp = [&](int n, int m, vpi& e) -> std::pair<vi, vi> {
4      vi g(e.size()), l(n + 1, -1), r(m + 1, -1), d(n + 2);
5      for (auto [u, v] : e) d[u]++;
6      std::partial_sum(all(d), d.begin());
7      for (auto [u, v] : e) g[--d[u]] = v;
8      for (vi a, p, q(n + 1);) {
9          a.assign(n + 1, -1);
10         p.assign(n + 1, -1);
11         int t = 1;
12         for (int i = 1; i <= n; i++) {
13             if (l[i] == -1) {
14                 q[t++] = a[i] = p[i] = i;
15             }
16         }
17         bool match = false;
18         for (int i = 1; i < t; i++) {
19             int u = q[i];
20             if (l[a[u]] != -1) continue;
21             for (int j = d[u]; j < d[u + 1]; j++) {
22                 int v = g[j];
23                 if (r[v] == -1) {
24                     while (v != -1) {
25                         r[v] = u;
26                         std::swap(l[u], v);
27                         u = p[u];
28                     }
29                     match = true;
30                     break;
31                 }
32                 if (p[r[v]] == -1) {
33                     q[t++] = v = r[v];
34                     p[v] = u;
35                     a[v] = a[u];
36                 }
37             }
38         }
39         if (!match) break;
40     }
41     return {l, r};
42 };

```

二分图最大权匹配

Kuhn-Munkres

注意是否为完美匹配, 非完美选 0, 完美选 $-INF$. (存疑)

```

1  /* Kuhn-Munkres */
2  auto KM = [&](int n, vvl e) -> std::tuple<LL, vi, vi> {
3      vl la(n + 1), lb(n + 1), pp(n + 1), vx(n + 1);
4      vi l(n + 1, -1), r(n + 1, -1);
5      vi va(n + 1), vb(n + 1);
6      LL delta;
7      auto bfs = [&](int x) -> void {
8          int a, y = 0, y1 = 0;
9          std::fill(all(pp), 0);
10         std::fill(all(vx), INF);
11         r[y] = x;
12         do {
13             a = r[y], delta = INF, vb[y] = 1;
14             for (int b = 1; b <= n; b++) {
15                 if (!vb[b]) {
16                     if (vx[b] > la[a] + lb[b] - e[a][b]) {

```

```

17         vx[b] = la[a] + lb[b] - e[a][b];
18         pp[b] = y;
19     }
20     if (vx[b] < delta) {
21         delta = vx[b];
22         y1 = b;
23     }
24 }
25 }
26 for (int b = 0; b <= n; b++) {
27     if (vb[b]) {
28         la[r[b]] -= delta;
29         lb[b] += delta;
30     } else
31         vx[b] -= delta;
32 }
33 y = y1;
34 } while (r[y] != -1);
35 while (y) {
36     r[y] = r[pp[y]];
37     y = pp[y];
38 }
39 };
40 for (int i = 1; i <= n; i++) {
41     std::fill(all(vb), 0);
42     bfs(i);
43 }
44 LL ans = 0;
45 for (int i = 1; i <= n; i++) {
46     if (r[i] == -1) continue;
47     l[r[i]] = i;
48     ans += e[r[i]][i];
49 }
50 return {ans, l, r};
51 };
52
53 auto [ans, mchl, mchr] = KM(n, e);

```

10.24 matching - matching on general graph

11 geometry

11.1 two demention

点与向量

```

1 struct Point {
2     LL x = 0, y = 0;
3     Point() = default;
4     Point(long long x, long long y) : x(x), y(y) {}
5     operator bool() { return *this != Point{}; }
6     friend bool operator==(Point p, Point q) { return p.x == q.x and p.y == q.y; }
7     friend bool operator!=(Point p, Point q) { return !(p == q); }
8     friend Point operator+(Point p, Point q) { return {p.x + q.x, p.y + q.y}; }
9     friend Point operator-(Point p, Point q) { return {p.x - q.x, p.y - q.y}; }
10    friend LL dot(Point p, Point q) { return p.x * q.x + p.y * q.y; }
11    friend LL det(Point p, Point q) { return p.x * q.y - q.x * p.y; }
12    friend bool operator<(Point p, Point q) {
13        return std::pair{p.quad(), det(q, p)} < std::pair{q.quad(), 0ll};
14        return (p.x == q.x ? p.y < q.y : p.x < q.x);
15    }
16    int quad() const {
17        if (x > 0 && y >= 0) return 1;
18        if (x <= 0 and y > 0) return 2;
19        if (x < 0 and y <= 0) return 3;
20        if (x >= 0 and y < 0) return 4;
21        return 0;
22    }
23    friend LL dist(Point p, Point q) {
24        return (p.x - q.x) * (p.x - q.x) + (p.y - q.y) * (p.y - q.y);
25    }
26 };
27 std::istream& operator>>(std::istream& is, Point& p) { return is >> p.x >> p.y; }
28 std::ostream& operator<<(std::ostream& os, Point p) {
29     return os << '(' << p.x << ',' << p.y << ')';
30 }

```

线段

```

1 struct line {
2     point a, b;
3
4     line(point _a = {}, point _b = {}) { a = _a, b = _b; }
5
6     /* 交点类型为 double */
7     friend point iPoint(line p, line q) {
8         point v1 = p.b - p.a;
9         point v2 = q.b - q.a;
10        point u = q.a - p.a;
11        return q.a + (q.b - q.a) * ((u ^ v1) * 1. / (v1 ^ v2));
12    }
13
14    /* 极角排序 */
15    bool operator<(const line& p) const {
16        double t1 = std::atan2((b - a).y, (b - a).x);
17        double t2 = std::atan2((p.b - p.a).y, (p.b - p.a).x);
18        if (fabs(t1 - t2) > eps) {
19            return t1 < t2;
20        }
21        return ((p.a - a) ^ (p.b - a)) > eps;
22    }
23 };

```

11.2 convex

2D

```

1 /* andrew */
2 auto andrew = [&](std::vector<point>& v) -> std::vector<point> {
3     std::sort(all(v));

```

```

4      std::vector<point> stk;
5      for (int i = 0; i < n; i++) {
6          point x = v[i];
7          while (stk.size() > 1 and ((stk.end()[-1] - stk.end()[-2]) ^ (x - stk.end()[-2])) <= 0) {
8              stk.pop_back();
9          }
10         stk.push_back(x);
11     }
12     int tmp = stk.size();
13     for (int i = n - 2; i >= 0; i--) {
14         point x = v[i];
15         while (stk.size() > tmp and ((stk.end()[-1] - stk.end()[-2]) ^ (x - stk.end()[-2])) <= 0) {
16             stk.pop_back();
17         }
18         stk.push_back(x);
19     }
20     return stk;
21 };

```

求凸包 and 判断点与凸包关系

```

1  #include<bits/stdc++.h>
2  #ifdef LOCAL
3  #include "debug.h"
4  #else
5  #define debug(...) 0
6  #endif
7  #define all(v) begin(v), end(v)
8  using namespace std;
9  using pii = pair<int, int>;
10
11 template<typename T, typename P, T inf = numeric_limits<T>::max()>
12 struct Convex_Hull {
13     using ptt = pair<T, T>;
14     // using i128 = __int128;
15     vector<ptt> a, b;
16     T lox, hix;
17     P crs(const ptt& a, const ptt& b) {
18         return (P)a.first * b.second - (P)a.second * b.first;
19     }
20     ptt mns(const ptt& a, const ptt& b) {
21         return ptt{a.first - b.first, a.second - b.second};
22     };
23     Convex_Hull(vector<ptt> c) {
24         assert(c.size() > 0);
25         sort(begin(c), end(c));
26         vector<int> st = {0};
27         int n = c.size(), tp = 0;
28         for (int i = 1; i < n; ++i) {
29             while (tp > 0 &&
30                 crs(mns(c[st[tp]], c[st[tp - 1]]), mns(c[i], c[st[tp]])) <= 0) {
31                 --tp, st.pop_back();
32             }
33             st.emplace_back(i), ++tp;
34         }
35         int tmp = tp;
36         for (int i = n - 1; ~i; --i) {
37             while (tp > tmp &&
38                 crs(mns(c[st[tp]], c[st[tp - 1]]), mns(c[i], c[st[tp]])) <= 0) {
39                 --tp, st.pop_back();
40             }
41             st.emplace_back(i), ++tp;
42         }
43         for (int i = 0; i <= tmp; ++i) {
44             a.emplace_back(c[st[i]]);
45         }
46         for (int i = tmp; i <= tp; ++i) {
47             b.emplace_back(c[st[i]]);
48         }
49     }
50     // n >= 3
51     pair<int, vector<ptt>> insd(T x, T y) { // 0: outside, 1: invertex, 2:{u,v}: inedge, 3: inside
52         ptt o = {x, y};
53         if (x < a[0].first || x > b[0].first) return {0, {}};
54         int li = lower_bound(begin(a), end(a), ptt{x, -inf}) - begin(a);
55         if (o == a[li]) return {1, {}};
56         int hi = lower_bound(begin(b), end(b), ptt{x, inf}, greater{}) - begin(b);
57         if (o == b[hi]) return {1, {}};
58         if (li == 0) {
59             if (hi + 1 == b.size()) return {0, {}};
60             assert(b.end()[-1].first == b.end()[-2].first);
61             if (y < b.end()[-1].second || y > b.end()[-2].second) return {0, {}};
62             return {2, {b.end()[-1], b.end()[-2]}};

```

```

63     }
64     if (hi == 0) {
65         if (li + 1 == a.size()) return {0, {}};
66         assert(a.end()[-2].first == a.end()[-1].first);
67         if (y < a.end()[-2].second || y > a.end()[-1].second) return {0, {}};
68         return {2, {a.end()[-2], a.end()[-1]}};
69     }
70     P v1 = crs(mns(o, a[li - 1]), mns(a[li], a[li - 1]));
71     if (v1 == 0) return {2, {a[li - 1], a[li]}};
72     P v2 = crs(mns(o, b[hi - 1]), mns(b[hi], b[hi - 1]));
73     if (v2 == 0) return {2, {b[hi - 1], b[hi]}};
74     debug(v1, v2);
75     return {v1 > 0 && v2 > 0 || v1 < 0 && v2 < 0 ? 3 : 0, {}};
76 }
77 };
78
79 namespace Acc {
80     auto work = []() {
81         string ans[] = {"OUT", "ON", "ON", "IN"};
82         int n, q;
83         cin >> n;
84         vector<pair<int, int>> a(n);
85         for (auto& [x, y] : a) cin >> x >> y;
86         Convex_Hull<int, long long> ch(a);
87         debug(ch.a, ch.b);
88         cin >> q;
89         for (int x, y; q--;) {
90             cin >> x >> y;
91             cout << ans[ch.insd(x, y).first] << '\n';
92         }
93     };
94 }
95
96 int main() {
97     std::ios::sync_with_stdio(0);
98     std::cin.tie(0);
99     int T = 1;
100     // std::cin >> T;
101     while (T--) Acc::work();
102 }

```

11.3 half plane union

```

1  /* half plane union */
2  auto half_plane = [&](std::vector<line>& ln) -> std::vector<point> {
3      std::sort(all(ln));
4      ln.erase(
5          unique(
6              all(ln),
7              [](line& p, line& q) {
8                  double t1 = std::atan2((p.b - p.a).y, (p.b - p.a).x);
9                  double t2 = std::atan2((q.b - q.a).y, (q.b - q.a).x);
10                 return fabs((t1 - t2)) < eps;
11             }),
12          ln.end());
13      auto check = [&](line p, line q, line r) -> bool {
14          point a = iPoint(p, q);
15          return ((r.b - r.a) ^ (a - r.a)) < -eps;
16      };
17      line q[ln.size() + 2];
18      int hh = 1, tt = 0;
19      q[+tt] = ln[0];
20      q[+tt] = ln[1];
21      for (int i = 2; i < (int) ln.size(); i++) {
22          while (hh < tt and check(q[tt - 1], q[tt], ln[i])) tt--;
23          while (hh < tt and check(q[hh + 1], q[hh], ln[i])) hh++;
24          q[+tt] = ln[i];
25      }
26      while (hh < tt and check(q[tt - 1], q[tt], q[hh])) tt--;
27      while (hh < tt and check(q[hh + 1], q[hh], q[tt])) hh++;
28      q[tt + 1] = q[hh];
29      std::vector<point> ans;
30      for (int i = hh; i <= tt; i++) {
31          ans.push_back(iPoint(q[i], q[i + 1]));
32      }
33      return ans;
34 };

```

11.4 rotate


```

1  /* rotate @ wrb */
2  #include<cstdio>
3  #include<algorithm>
4  #define db double
5  namespace Acc{
6      const int N = 5e4+10;
7      struct node{
8          int x,y;
9      }a[N],stk[N];
10     db cmp(node a,node b,node c){return 1.*(c.x-a.x)*(c.y-b.y)-1.*(c.x-b.x)*(c.y-a.y);}
11     int dis(node a,node b){return ((a.x-b.x)*(a.x-b.x)+(a.y-b.y)*(a.y-b.y));}
12     int n,tp,ans;
13     void work(){
14         scanf("%d",&n);
15         for(int i=1;i<=n;i++)scanf("%d%d",&a[i].x,&a[i].y);
16         std::sort(a+1,a+n+1,[=](node a,node b)->bool{return a.x<b.x || (a.x==b.x && a.y<b.y);});
17         stk[1]=a[1],tp=1;
18         for(int i=2;i<=n;i++){
19             while(tp>1 && cmp(stk[tp-1],stk[tp],a[i])<=0)tp--;
20             stk[++tp]=a[i];
21         }
22         int tmp=tp;
23         for(int i=n-1;i>=1;i--){
24             while(tp>tmp && cmp(stk[tp-1],stk[tp],a[i])<=0)tp--;
25             stk[++tp]=a[i];
26         }
27         for(int i=1,j=3;i<tp;i++){
28             while(cmp(stk[i],stk[i+1],stk[j])<cmp(stk[i],stk[i+1],stk[j+1]))j=j%(tp-1)+1;
29             ans=std::max(ans,std::max(dis(stk[i],stk[j]),dis(stk[i+1],stk[j]))));
30         }
31         printf("%d",ans);
32     }
33 }
34 int main(){
35     return Acc::work(),0;
36 }

```

11.5 Simpson

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  double a;
4  double f(double x) {
5      return pow(x, a / x - x);
6  }
7  double simpson(double l, double r) {
8      double mid = (l + r) / 2;
9      return (r - l) * (f(l) + 4 * f(mid) + f(r)) / 6;
10 }
11 double asr(double l, double r, double eps, double ans, int d) {
12     double mid = (l + r) / 2;
13     double Fl = simpson(l, mid), Fr = simpson(mid, r);
14     if (abs(Fl + Fr - ans) <= 15 * eps && d < 0) {
15         return Fl + Fr + (Fl + Fr - ans) / 15;
16     }
17     return asr(l, mid, eps / 2, Fl, d - 1) + asr(mid, r, eps / 2, Fr, d - 1);
18 }
19 double calc(double l, double r, double eps) {
20     return asr(l, r, eps, simpson(l, r), 12);
21 }
22 int main() {
23     cin >> a;
24     if (a < 0) {
25         cout << "orz\n";
26     } else {
27         cout << fixed << setprecision(5) << calc(1e-8, 15, 1e-8) << '\n';
28     }
29 }

```

12 offline algorithm

12.1 discretization

```

1 std::sort(all(a));
2 a.erase(unique(all(a)), a.end());
3 auto get_id = [&](const int& x) -> int { return lower_bound(all(a), x) - a.begin() + 1; };

```

12.2 Mo algorithm

普通莫队

```

1 int block = n / sqrt(2 * m / 3);
2 std::sort(all(q), [&](node a, node b) {
3     return a.l / block == b.l / block ? (a.r == b.r ? 0 : ((a.l / block) & 1) ^ (a.r < b.r))
4         : a.l < b.l;
5 });
6 auto move = [&](int x, int op) -> void {
7     if (op == 1) {
8         /* operations */
9     } else {
10        /* operations */
11    }
12 };
13 for (int k = 1, l = 1, r = 0; k <= m; k++) {
14     node Q = q[k];
15     while (l > Q.l) {
16         move(--l, 1);
17     }
18     while (r < Q.r) {
19         move(++r, 1);
20     }
21     while (l < Q.l) {
22         move(l++, -1);
23     }
24     while (r > Q.r) {
25         move(r--, -1);
26     }
27 }

```

12.3 回滚莫队

```

1 /* rollback Mo */
2 #include<bits/stdc++.h>
3 namespace Acc {
4     const int N = 200009;
5     int a[N], b[N], id[N], f[N], g[N], p[N], z[N];
6     std::pair<int, int> st[N];
7     struct T {
8         int l, r, o;
9     } q[N];
10    auto work = []() {
11        int n, m;
12        std::cin >> n;
13        for (int i = 1; i <= n; ++i) {
14            std::cin >> a[i], b[i] = a[i];
15        }
16        std::sort(b + 1, b + n + 1);
17        int ct = std::unique(b + 1, b + n + 1) - b - 1;
18        for (int i = 1; i <= n; ++i) {
19            a[i] = std::lower_bound(b + 1, b + ct + 1, a[i]) - b;
20        }
21        std::cin >> m;
22        for (int i = 1; i <= m; ++i) {
23            auto&[l, r, o] = q[i];
24            std::cin >> l >> r, o = i;
25        }
26        int B = ceil(n / sqrt(m));
27        for (int i = 1; i <= n; ++i) {
28            id[i] = (i - 1) / B + 1;
29        }
30        std::sort(q + 1, q + m + 1, [](T a, T b) {

```

```

31     return id[a.l] == id[b.l] ? a.r < b.r : a.l < b.l;
32 };
33 int ans = 0, L = 1, R = 0;
34 for (int i = 1; i <= m; ++i) {
35     auto[l, r, o] = q[i];
36     if (id[l] != id[q[i - 1].l]) {
37         ans = 0;
38         R = std::min(n, id[l] * B), L = R + 1;
39         memset(f + 1, 0, ct << 2);
40         memset(g + 1, 0, ct << 2);
41     }
42     if (id[l] == id[r]) {
43         for (int j = 1; j <= r; ++j) {
44             if (p[a[j]] == 0) p[a[j]] = j;
45             else ans = std::max(ans, j - p[a[j]]);
46         }
47         for (int j = 1; j <= r; ++j) p[a[j]] = 0;
48         z[o] = ans, ans = 0;
49     } else {
50         while (R < r) {
51             ++R, g[a[R]] = R;
52             if (f[a[R]] == 0) f[a[R]] = R;
53             else ans = std::max(ans, R - f[a[R]]);
54         }
55         int las = ans, t = L;
56         while (l < L) {
57             --L;
58             int x = f[a[L]], y = g[a[L]];
59             st[L] = std::make_pair(x, y);
60             f[a[L]] = L;
61             if (g[a[L]] == 0) g[a[L]] = L;
62             else ans = std::max(ans, g[a[L]] - L);
63         }
64         z[o] = ans;
65         for (int j = 1; j < t; ++j) {
66             auto[x, y] = st[j];
67             f[a[j]] = x, g[a[j]] = y;
68         }
69         ans = las, L = t;
70     }
71 }
72 for (int i = 1; i <= m; ++i) {
73     std::cout << z[i] << '\n';
74 }
75 };
76 }
77 int main() {
78     std::ios::sync_with_stdio(0);
79     std::cin.tie(0), Acc::work();
80 }

```

12.4 CDQ

n 个三维数对 (a_i, b_i, c_i) , 设 $f(i)$ 表示 $a_j \leq a_i, b_j \leq b_i, c_j \leq c_i (i \neq j)$ 的个数. 输出 $f(i) (0 \leq i \leq n-1)$ 的值.

```

1 // 洛谷 P3810 【模板】三维偏序 (陌上花开)
2
3 struct data {
4     int a, b, c, cnt, ans;
5
6     data(int _a = 0, int _b = 0, int _c = 0, int _cnt = 0, int _ans = 0) {
7         a = _a, b = _b, c = _c, cnt = _cnt, ans = _ans;
8     }
9
10    bool operator!=(data x) {
11        if (a != x.a) return true;
12        if (b != x.b) return true;
13        if (c != x.c) return true;
14        return false;
15    }
16 };
17
18 int main() {
19     std::ios::sync_with_stdio(false);
20     std::cin.tie(0);
21
22     int n, k;
23     std::cin >> n >> k;
24     static data v1[N], v2[N];
25     for (int i = 1; i <= n; i++) {

```

```

26     std::cin >> v1[i].a >> v1[i].b >> v1[i].c;
27 }
28 std::sort(v1 + 1, v1 + n + 1, [&](data x, data y) {
29     if (x.a != y.a) return x.a < y.a;
30     if (x.b != y.b) return x.b < y.b;
31     return x.c < y.c;
32 });
33 int t = 0, top = 0;
34 for (int i = 1; i <= n; i++) {
35     t++;
36     if (v1[i] != v1[i + 1]) {
37         v2[++top] = v1[i];
38         v2[top].cnt = t;
39         t = 0;
40     }
41 }
42 vi tr(N);
43 auto add = [&](int pos, int val) -> void {
44     while (pos <= k) {
45         tr[pos] += val;
46         pos += lowbit(pos);
47     }
48 };
49 auto query = [&](int pos) -> int {
50     int ans = 0;
51     while (pos > 0) {
52         ans += tr[pos];
53         pos -= lowbit(pos);
54     }
55     return ans;
56 };
57 std::function<void(int, int)> CDQ = [&](int l, int r) -> void {
58     if (l == r) return;
59     int mid = (l + r) >> 1;
60     CDQ(l, mid), CDQ(mid + 1, r);
61     std::sort(v2 + 1, v2 + mid + 1, [&](data x, data y) {
62         if (x.b != y.b) return x.b < y.b;
63         return x.c < y.c;
64     });
65     std::sort(v2 + mid + 1, v2 + r + 1, [&](data x, data y) {
66         if (x.b != y.b) return x.b < y.b;
67         return x.c < y.c;
68     });
69     int i = l, j = mid + 1;
70     while (j <= r) {
71         while (i <= mid && v2[i].b <= v2[j].b) {
72             add(v2[i].c, v2[i].cnt);
73             i++;
74         }
75         v2[j].ans += query(v2[j].c);
76         j++;
77     }
78     for (int ii = l; ii < i; ii++) {
79         add(v2[ii].c, -v2[ii].cnt);
80     }
81     return;
82 };
83 CDQ(1, top);
84 vi ans(n + 1);
85 for (int i = 1; i <= top; i++) {
86     ans[v2[i].ans + v2[i].cnt] += v2[i].cnt;
87 }
88 for (int i = 1; i <= n; i++) {
89     std::cout << ans[i] << endl;
90 }
91 return 0;
92 }

```

12.5 segment tree devide and conquer

```

1  /* seg div @ wrb */
2  #include<bits/stdc++.h>
3  using namespace std;
4  namespace Acc {
5      const int N = 1e5;
6      pair<int, int> q[N * 2];
7      vector<int> v[N * 4];
8      int n, l, r, p;
9      int fa[N * 2], sz[N * 2];
10     pair<int, int> st[N * 2];
11     int tp;
12     void ins(int o, int L, int R) {

```

```

13     if (r < L || l > R) return ;
14     if (l <= L && R <= r) {
15         v[o].emplace_back(p);
16         return ;
17     }
18     int md = L + R >> 1;
19     ins(o << 1, L, md);
20     ins(o << 1 | 1, md + 1, R);
21 }
22 auto gf = [](int x) {
23     while (x != fa[x]) x = fa[x];
24     return x;
25 };
26 auto mg = [](int x, int y) {
27     x = gf(x), y = gf(y);
28     if (x != y) {
29         if (sz[x] < sz[y]) swap(x, y);
30         fa[y] = x, sz[x] += sz[y];
31         st[++tp] = {x, y};
32     }
33 };
34 void dfs(int o, int L, int R) {
35     int lastp = tp;
36     for (int i : v[o]) {
37         auto[x, y] = q[i];
38         mg(x, y + n), mg(x + n, y);
39         if (gf(x) == gf(x + n)) {
40             for (int i = L; i <= R; ++i) {
41                 cout << "No\n";
42             }
43             goto _;
44         }
45     }
46     if (L == R) {
47         cout << "Yes\n";
48     } else {
49         int md = L + R >> 1;
50         dfs(o << 1, L, md);
51         dfs(o << 1 | 1, md + 1, R);
52     }
53 _:
54     for (; tp > lastp; --tp) {
55         auto[x, y] = st[tp];
56         fa[y] = y, sz[x] -= sz[y];
57     }
58 }
59 auto work = []() {
60     int m, k;
61     cin >> n >> m >> k;
62     for (int i = 1; i <= m; ++i) {
63         int x, y;
64         cin >> x >> y >> l >> r;
65         if (++l <= r) {
66             q[p = i] = {x, y}, ins(1, 1, k);
67         }
68     }
69     iota(fa + 1, fa + n * 2 + 1, 1);
70     fill(sz + 1, sz + n * 2 + 1, 1);
71     dfs(1, 1, k);
72 };
73 }
74 int main() {
75     ios::sync_with_stdio(0);
76     cin.tie(0), Acc::work();
77 }

```

13 Print All Cases

13.1 print all trees with n nodes

构造所有 n 个节点的树.

13.1.1 有根树

表示其数量的数列在 oeis 上编号为 A000081. $n = 1, 2, 3 \dots, 20$ 的项分别为:

1, 1, 2, 4, 9,
20, 48, 115, 286, 719,
1842, 4766, 12486, 32973, 87811,
235381, 634847, 1721159, 4688676, 12826228.

构造所有 $n \leq 20$ 的有根树的 (平均) 运行时间为 15.7054s.

```

1  /* integer partition */
2  int n = 5;
3  std::vector<vvi> part(n + 1);
4  auto integerPartition = [&](int n) {
5      // part[i] = {{i}};
6      for (int i = 1; i <= n; i++) {
7          part[i].push_back({i});
8          for (int j = 1; j < i; j++) {
9              for (const auto& v : part[i - j]) {
10                 vi tmp = v;
11                 tmp.push_back(j);
12                 std::sort(all(tmp));
13                 part[i].push_back(tmp);
14             }
15         }
16         std::sort(all(part[i]));
17         part[i].erase(unique(all(part[i])), part[i].end());
18     }
19 };
20 integerPartition(n);
21 /* find all trees */
22 std::vector<std::vector<std::string>> trees(n + 1);
23 auto allTrees = [&](int n) {
24     std::string s;
25     for (int i = 1; i < n; i++) s += '(';
26     for (int i = 1; i < n; i++) s += ')';
27     trees[n].push_back(s);
28     for (const auto& v : part[n - 1]) {
29         std::vector<std::string> now;
30         auto dfs = [&](auto&& self, int i) {
31             if (i == v.size()) {
32                 std::string s = "";
33                 auto tmp = now;
34                 std::sort(all(tmp));
35                 for (const auto& ss : tmp) s += '(' + ss + ')';
36                 trees[n].push_back(s);
37                 return;
38             }
39             for (const auto& s : trees[v[i]]) {
40                 now.push_back(s);
41                 self(self, i + 1);
42                 now.pop_back();
43             }
44         };
45         dfs(dfs, 0);
46     }
47     std::sort(all(trees[n]));
48     trees[n].erase(unique(all(trees[n])), trees[n].end());
49 };
50 for (int i = 1; i <= n; i++) {
51     allTrees(i);
52     debug(i, trees[i].size());
53     std::cout << '\n';
54 }
55 for (const auto& s : trees[n]) {
56     vvi e(n + 1);
57     vi fa(n + 1);
58     int cnt = 1, now = 1;

```

```
59     for (const auto& c : s) {
60         if (c == '(') {
61             cnt += 1;
62             e[now].push_back(cnt);
63             e[cnt].push_back(now);
64             fa[cnt] = now;
65             now = cnt;
66         } else {
67             now = fa[now];
68         }
69     }
70     debug(e);
71     /* do the things you need */
72 }
```

14 Magic

14.1 magic heap

对顶堆维护中位数.

```

1  /* magic heap */
2  struct MagicHeap {
3      LL suml = 0, sumr = 0;
4      std::priority_queue<int> ql;
5      std::priority_queue<int, std::vector<int>, std::greater<int>> qr;
6      void le2ri() {
7          auto x = ql.top();
8          suml -= x, ql.pop();
9          sumr += x, qr.push(x);
10     };
11     void ri2le() {
12         auto x = qr.top();
13         sumr -= x, qr.pop();
14         suml += x, ql.push(x);
15     };
16     void pushL(int x) { suml += x, ql.push(x); }
17     void pushR(int x) { sumr += x, qr.push(x); }
18     void push(int x) {
19         if (ql.empty()) {
20             pushL(x);
21         } else if (qr.empty()) {
22             (x <= ql.top() ? le2ri(), pushL(x) : pushR(x));
23         } else {
24             int le = ql.top(), ri = qr.top();
25             if (le <= x and x <= ri) {
26                 (ql.size() == qr.size() ? pushL(x) : pushR(x));
27             } else if (x < le) {
28                 if (ql.size() != qr.size()) le2ri();
29                 pushL(x);
30             } else {
31                 if (ql.size() <= qr.size()) ri2le();
32                 pushR(x);
33             }
34         }
35     }
36     int size() { return ql.size() + qr.size(); }
37     bool empty() { return ql.empty() and qr.empty(); }
38     LL val() { return suml + sumr; }
39     LL mid() { return ql.top(); }
40     LL dist() { return sumr - suml + ql.top() * (ql.size() - qr.size()); }
41 };

```

14.2 operator queue

双栈维护队列半群.

```

1  template <typename T, typename Op>
2  struct OpQueue {
3      static_assert(std::is_convertible_v<std::invoke_result_t<Op, T, T>, T>);
4      const T e;
5      const Op op;
6      std::vector<T> l, r, a;
7      OpQueue(T e, Op op) : e(e), op(op), l{e}, r{e} {}
8      T val() const { return op(l.back(), r.back()); }
9      void push(T x) {
10         r.push_back(op(r.back(), x));
11         a.push_back(x);
12     }
13     void pop() {
14         if (l.size() == 1) {
15             for (; !a.empty(); a.pop_back()) {
16                 l.push_back(op(a.back(), l.back()));
17             }
18             r.resize(1);
19         }
20         assert(l.size() > 1);
21         l.pop_back();
22     }
23     int size() const { return l.size() + r.size() - 2; }
24     bool empty() const { return l.size() + r.size() == 2; }
25 };

```



```

26
27 /* When using this, remember to replace "T" with correct type and "e" with identity in the half group. */
28 auto op = [] (T a, T b) -> T {
29     /* You operations */
30 };
31 OpQueue<T, decltype(op)> a(e, op);

```

14.3 Fast GCD

$O(V)$ 预处理, $O(1)$ 查询 GCD.

```

1  /* fast GCD @ luogu shit */
2  const int N = 5005, M = 1e6 + 5, S = 1000, P = 998244353;
3
4  int _a[M], _b[M], _c[M];
5  int v[M], p[M], r;
6  int f[S + 1][S + 1];
7  int a[N], b[N];
8
9  void Init() {
10     _a[1] = _b[1] = _c[1] = 1;
11     for (int i = 2; i <= M - 5; ++i) {
12         if (!v[i]) {
13             p[++r] = i;
14             _a[i] = _b[i] = 1, _c[i] = i;
15         }
16         int tp;
17         for (int j = 1; j <= r && (tp = i * p[j]) <= M - 5; ++j) {
18             v[tp] = 1;
19             _a[tp] = _a[i] * p[j];
20             _b[tp] = _b[i];
21             _c[tp] = _c[i];
22             if (_a[tp] > _b[tp]) {
23                 swap(_a[tp], _b[tp]);
24                 if (_b[tp] > _c[tp]) {
25                     swap(_b[tp], _c[tp]);
26                 }
27             }
28             if (!(i % p[j])) {
29                 break;
30             }
31         }
32     }
33
34     for (int i = 1; i <= S; ++i) {
35         f[0][i] = f[i][0] = i;
36         for (int j = 1; j <= S; ++j) {
37             f[i][j] = f[j % i][i];
38         }
39     }
40     return;
41 }
42
43 int gcd(int x, int y) {
44     int A = 1, tp = f[_a[x]][y % _a[x]];
45     A *= tp;
46     y /= tp;
47     tp = f[_b[x]][y % _b[x]];
48     A *= tp;
49     y /= tp;
50     tp = (v[_c[x]] ? f[_c[x]][y % _c[x]] : (y % _c[x] ? 1 : _c[x]));
51     A *= tp;
52     y /= tp;
53     return A;
54 }

```

14.4 $q \equiv \frac{a}{b} \pmod{mod}$

```

1  /* find q = a / b @ luogu shit */
2  pair<int, int> approx(int mod, int q, int bound) {
3      int x = q, y = mod, a = 1, b = 0;
4      while (x > bound) {
5          swap(x, y), swap(a, b);
6          a -= x / y * b;
7          x %= y;
8      }
9      return make_pair(x, a);

```

10 | }
