

## 1 数据结构

- 1.1 并查集
- 1.2 单调栈
- 1.3 单调队列
- 1.4 树状数组
  - 1.4.1 单点修改与区间查询
  - 1.4.2 区间修改与单点查询
  - 1.4.3 区间修改与区间查询
- 1.5 线段树
  - temp1
  - temp2
  - temp3
  - 1.5.1 动态开点
- 1.6 对顶堆
- 1.7 主席树
- 1.8 Treap
  - 1.8.1 旋转Treap
  - 1.8.2 Fhq Treap
- 1.9 轻重链剖分
- 1.10 Dsu on Tree
- 1.11 Splay
- 1.12 莫队
- 1.13 二维数点
- 1.14 珂朵莉树
- 1.15 李超树

## 2. 图论

- 2.1 最短路
  - 2.1.1 朴素Dijkstra  $O(n^2)$
  - 2.1.2 堆优化Dijkstra  $O(m\log n)$
  - 2.1.3 Bellman-ford  $O(nm)$
  - 2.1.4 SPFA  $O(nm)$
  - 2.1.5 Floyd  $O(n^3)$
  - 2.1.6 路径还原
  - 2.1.7 最短路计数
- 2.2 最小生成树
  - 2.2.1 Prim
  - 2.2.2 Kruskal
- 2.3 二分图匹配
  - 2.3.1 染色法判断二分图
  - 2.3.2 匈牙利算法判断最大匹配
- 2.4 拓扑排序
- 2.5 倍增LCA
- 2.6 Tarjan
  - 2.6.1 SCC
  - 2.6.2 BCC\_V
  - 2.6.3 BCC\_E
- 2.7 差分约束
- 2.8 树哈希
- 2.9 虚树
- 2.10 2-SAT
- 2.11 基环树
- 2.12 网络流
  - 2.12.1 Dinic
  - 2.12.2 MCMF
- 2.13 三元环/四元环计数
- 2.14 点分治

## 3 数学

- 3.1 线性筛
- 3.2 快速乘 & 快速幂
- 3.3 拓展欧几里得算法 exgcd

- 3.4 欧拉函数
- 3.5 约数 个数/和 定理
- 3.6 组合数
  - 结论
  - 板子
- 3.7 数论分块
- 3.8 卡特兰数
- 3.9 线性求逆
- 3.10 博弈论
- 3.11 最小二乘法求线性回归方程
- 3.12 线性基
  - temp1
  - temp2
- 3.13 Pollard-Rho
  - 板子2
- 3.14 矩阵类
- 3.15 叉积求极角排序
- 3.16 两圆交面积
- 3.17 全概率公式/贝叶斯公式
- 3.18 多项式卷积
  - 3.18.1 FFT
    - Ver线下
    - Ver线上
  - 3.18.2 NTT
    - jly板子
    - 无模NTT
  - 3.18.3 分治NTT
- 3.19 拓展中国剩余定理 excrt
- 3.20 Point

## 4 动态规划

- 4.1 线性DP
  - 4.1.2 最长上升子序列  $O(n\log n)$
  - 4.1.3 最长公共子序列  $O(n^2)$
  - 4.1.4 最长公共上升子序列  $O(n^2)$
- 4.2 背包
  - 4.2.1 01背包  $O(nm)$
  - 4.2.2 完全背包  $O(nm)$
  - 4.2.3 多重背包
    - 4.2.3.1 二进制优化  $O(NV\log s)$
    - 4.2.3.2 单调队列优化  $O(NV)$
  - 4.2.4 分组背包  $O(nms)$
  - 4.2.5 超大背包
  - 4.2.6 二维费用背包(有物品数量限制的背包)  $O(nmk)$
  - 4.2.7 混合背包
  - 4.2.6 背包问题输出方案
- 4.3 区间DP  $O(n^3)$
- 4.4 树形DP
- 4.5 状压DP
- 4.6 数位DP
- 4.7 单调队列优化DP

## 5. 字符串

- 5.1 KMP
- 5.2 Manacher
- 5.3 Hash
  - 二维哈希
- 5.4 ACAM
- 5.5 最小表示法
- 5.6 Trie
- 5.7 PAM
- 5.8 Z
- 5.9 序列自动机

## 6 杂项

- 6.1 高精度
- 6.2 二维前缀和
- 6.3 双指针
- 6.4 ST表
  - 二维ST表
- 6.5 快读
- 6.6 模拟退火
- 6.7 STL
- 6.8 离散化
- 6.9 随机数
- 6.10 ModInt
  - Z
  - mint
- 6.11 重载哈希
- 6.12 取模
- 6.13 枚举子集

# 1 数据结构

## 1.1 并查集

```
1 struct DSU {
2     vector<int> f, siz;
3     DSU(int n) : f(n + 1), siz(n + 1, 1) {
4         siz[0] = 0;
5         iota(f.begin(), f.end(), 0);
6     }
7     bool same(int x, int y) {
8         return find(x) == find(y);
9     }
10    int find(int x) {
11        while (x != f[x]) x = f[x] = f[f[x]];
12        return x;
13    }
14    bool merge(int x, int y) {
15        x = find(x), y = find(y);
16        if (x == y) return 0;
17        if (siz[x] < siz[y]) swap(x, y);
18        siz[x] += siz[y]; f[y] = x;
19        return 1;
20    }
21    int size(int x) {
22        return siz[find(x)];
23    }
24 };
```

## 1.2 单调栈

```
1     找出每个数左边离它最近的比它大/小的数
2     stack<int> s;
3     for (int i = 1; i <= n; i++)
4     {
5         while (sz(s) && check(s.top(), i)) s.pop();
6         s.push(i);
7     }
```

## 1.3 单调队列

```
1 // 滑动窗口
2 int a[N], q[N]; // q为单调队列，需要注意，队列中存放的是数组下标。
3
4 int main()
5 {
6     int n, k;
7     //n个数，窗口长度为k
8     cin >> n >> k;
9
10    for (int i = 0; i < n; i++) cin >> a[i];
11
12    int hh = 0, tt = -1; // hh为队头，tt为队尾。tt < hh的原因是防止在刚进入循环时判断出错，所以
    //要将tt小于hh，从而达到先入一个数的情况。
13    for (int i = 0; i < n; i++) { // 模拟队尾碰到新数a[i]的过程
14        if (hh <= tt && i - k + 1 > q[hh]) hh++; // 单调队列长度大于k，队头出队，加上hh <=
        //tt的原因主要是防止初始状态时tt < hh。
15
16        while (hh <= tt && a[q[tt]] >= a[i]) tt--; // 碰到新数a[i]后，若当前队尾大于a[i]，则
        //放入后不满足单调递减
17        q[++tt] = i; // 新元素入队
18
19        if (i >= k - 1) printf("%d ", a[q[hh]]); // 用单调队列维护一个单调递减的区间，故每次窗口
        //内最大值一定是队头
20    }
21
22    printf("\n");
23
24    // 以下同上，改为维护单调递增即可。
25    hh = 0, tt = -1;
26    for (int i = 0; i < n; i++) {
27        if (hh <= tt && i - k + 1 > q[hh]) hh++;
28
29        while (hh <= tt && a[q[tt]] <= a[i]) tt--;
30        q[++tt] = i;
31
32        if (i >= k - 1) printf("%d ", a[q[hh]]);
33    }
34    printf("\n");
35
36    return 0;
37 }
```

## 1.4 树状数组

### 1.4.1 单点修改与区间查询

```
1 template <typename T>
2 struct Fenwick {
3     int n;
4     vector<T> a;
5     Fenwick (int n) : n (n), a (n + 10) {}
6     void add (int x, T v) {
7         for (int i = x; i <= n; i += i & -i) {
8             a[i] += v;
9         }
10    }
11    T sum (int x) {
```

```

12     T ans = 0;
13     for (int i = x; i; i -= i & -i) {
14         ans += a[i];
15     }
16     return ans;
17 }
18 T rangeSum (int l, int r) {
19     if (l <= 1) return sum(r);
20     return sum (r) - sum (l - 1);
21 }
22 int kth (T k) {
23     int x = 0;
24     for (int i = 1 << std::__lg (n); i; i /= 2) {
25         if (x + i <= n && k >= a[x + i]) {
26             x += i;
27             k -= a[x];
28         }
29     }
30     return x;
31 }
32 };

```

```

1 // sum sub
2 struct BIT {
3     int tr[1000010], N;
4     // BIT (int len) {setlen(len * 2); }
5     void setlen(int len) {N = len << 1; for (int i = 0; i <= N; i ++ ) tr[i] = 0; }
6     void add(int x, int k) {for (; x <= N; x += lowbit(x)) tr[x] += k; }
7     int pre(int x) {int res = 0; for (; x; x -= lowbit(x)) res = res + tr[x]; return res; }
8     int query(int l, int r) {if (l > r) swap(l, r); return pre(r) - pre(l - 1); }
9 };
10
11 //max min gcd
12 struct BIT {
13     int tr[1000010], info[1000010], N;
14     // BIT (int len) {setlen(len * 2); }
15     void setlen(int len) {N = len << 1; for (int i = 0; i <= N; i ++ ) tr[i] = info[i] = -
16     inf; }
17     void add(int x, int k) {
18         info[x] = k;
19         for (int i = x; i <= n; i += lowbit(i)) {
20             tr[i] = info[i];
21             for (int j = 1; j < lowbit(i); j <= 1) {
22                 tr[i] = max(tr[i], tr[i - j]);
23             }
24         }
25     }
26     int query(int l, int r) {
27         if (l > r) swap(l, r);
28         int ans = -inf;
29         while (r >= 1) {
30             ans = max(ans, info[r]);
31             r -- ;
32             for (; r - lowbit(r) >= 1; r -= lowbit(r)) {
33                 ans = max(ans, tr[r]);
34             }
35         }
36         return ans;
37     }
38 };

```

```

38
39
40 using i64 = long long;
41 template <typename T>
42 struct Fenwick {
43     int n;
44     std::vector<T> a;
45
46     Fenwick(int n = 0) {
47         init(n);
48     }
49
50     void init(int n) {
51         this->n = n;
52         a.assign(n, T());
53     }
54
55     void add(int x, T v) {
56         for (int i = x + 1; i <= n; i += i & -i) {
57             a[i - 1] += v;
58         }
59     }
60
61     T sum(int x) {
62         auto ans = T();
63         for (int i = x; i > 0; i -= i & -i) {
64             ans += a[i - 1];
65         }
66         return ans;
67     }
68
69     T rangeSum(int l, int r) {
70         return sum(r) - sum(l);
71     }
72
73     int kth(T k) {
74         int x = 0;
75         for (int i = 1 << std::__lg(n); i; i /= 2) {
76             if (x + i <= n && k >= a[x + i - 1]) {
77                 x += i;
78                 k -= a[x - 1];
79             }
80         }
81         return x;
82     }
83 };
84
85 struct Max {
86     int v;
87     Max(int x = -1E9) : v{x} {}
88
89     Max &operator+=(Max a) {
90         v = std::max(v, a.v);
91         return *this;
92     }
93 };

```

## 1.4.2 区间修改与单点查询

```
1  int t[maxn], q[maxn]; // 用t数组维护差分数组
2
3  int lowbit (int x) {
4      return x & -x;
5  }
6
7  void add(int x, int k){
8      for (; x <= n; x += x & -x) t[x] += k;
9  }
10
11 int query(int x){
12     int ans = 0;
13     for (; x; x -= x & -x) ans += t[x];
14     return ans;
15 }
16
17 int range (int l, int r) {
18     return query(r) - query(l - 1);
19 }
20
21 signed main()
22 {
23     cin >> n >> p;
24     for (int i = 1; i <= n; i ++ ){
25         cin >> q[i];
26         add(i, q[i] - q[i - 1]);
27     }
28
29     while (p -- ){
30         int op, l, r, x;
31         cin >> op;
32         if (op == 1){
33             cin >> l >> r >> x;
34             add(l, x), add(r + 1, -x);
35         }
36         if (op == 2){
37             cin >> x;
38             cout << q[x] + ask(x) << endl; // ask(x)即为q[x]的增量
39         }
40     }
41
42     return 0;
43 }
44
```

## 1.4.3 区间修改与区间查询

```
1  //区间修改与区间查询
2  #include <bits/stdc++.h>
3  using namespace std;
4  #define int long long
5  int n, p;
6  int q[1123456], t[1123456], s[1123456], sum[1123456];
7
8  inline int lowbit(int x)
9  {
10     return x & -x;
11 }
12
```

```

13 void add1(int x, int k)
14 {
15     for (int i = x; i <= n; i += lowbit(i)) t[i] += k, s[i] += x * k;
16 }
17
18 int ask1(int x)
19 {
20     int ans = 0;
21     for (int i = x; i; i -= lowbit(i)) ans += (x + 1) * t[i] - s[i];
22     return ans;
23 }
24
25 signed main()
26 {
27     cin >> n >> p;
28     for (int i = 1; i <= n; i ++ ){
29         cin >> q[i];
30         sum[i] = q[i] + sum[i - 1];
31         add1(i, q[i] - q[i - 1]);
32     }
33
34     while (p -- ){
35         int op, l, r, x;
36         cin >> op;
37         if (op == 1){
38             cin >> l >> r >> x;
39             add1(l, x), add1(r + 1, -x);
40         }
41         if (op == 2){
42             cin >> l >> r;
43             cout << ask1(r) - ask1(l - 1) << endl;
44         }
45     }
46
47     return 0;
48 }

```

## 1.5 线段树

### temp1

```

1  constexpr int N = 2e5 + 10;
2  array<int, N> a;
3  struct node {
4      ll sum;
5      int tag;
6  }tr[N << 2];
7  #define ls u << 1
8  #define rs u << 1 | 1
9  node pushup (node l, node r) {
10     node tmp;
11     tmp.tag = 0;
12     tmp.sum = l.sum + r.sum;
13     return tmp;
14 }
15 void pushdown (int u, int l, int r) {
16     if (tr[u].tag) {
17         int mid = l + r >> 1;
18         tr[ls].tag += tr[u].tag;

```



```

19     tr[rs].tag += tr[u].tag;
20     tr[ls].sum += (mid - l + 1) * tr[u].tag;
21     tr[rs].sum += (r - mid) * tr[u].tag;
22     tr[u].tag = 0;
23 }
24 }
25 void build (int u, int l, int r) {
26     if (l == r) {
27         tr[u] = {a[l], 0};
28     } else {
29         int mid = l + r >> 1;
30         build(ls, l, mid);
31         build(rs, mid + 1, r);
32         tr[u] = pushup(tr[ls], tr[rs]);
33     }
34 }
35 void modify (int u, int L, int R, int l, int r, int v) {
36     if (L >= l && R <= r) {
37         tr[u].tag += v;
38         tr[u].sum += (R - L + 1) * v;
39     } else {
40         pushdown(u, L, R);
41         int mid = L + R >> 1;
42         if (l <= mid) modify(ls, L, mid, l, r, v);
43         if (r > mid) modify(rs, mid + 1, R, l, r, v);
44         tr[u] = pushup(tr[ls], tr[rs]);
45     }
46 }
47 node query(int u, int L, int R, int l, int r) {
48     if (L >= l && R <= r) {
49         return tr[u];
50     } else {
51         pushdown(u, L, R);
52         int mid = L + R >> 1;
53         if (r <= mid) {
54             return query(ls, L, mid, l, r);
55         } else if (l > mid) {
56             return query(rs, mid + 1, R, l, r);
57         } else {
58             return pushup(query(ls, L, mid, l, r), query(rs, mid + 1, R, l, r));
59         }
60     }
61 }
62 int search (int u, int L, int R, int l, int r, int k) {
63     if (L == l && R == r) {
64         if (tr[u].mx < k) return -1;
65         else {
66             if (L == R) return L;
67             int mid = L + R >> 1;
68             if (tr[ls].mx >= k) return search(ls, L, mid, l, mid, k);
69             else return search(rs, mid + 1, R, mid + 1, r, k);
70         }
71     }
72     int mid = L + R >> 1;
73     if (r <= mid) return search(ls, L, mid, l, r, k);
74     else if (l > mid) return search(rs, mid + 1, R, l, r, k);
75     else {
76         int pos = search(ls, L, mid, l, mid, k);
77         if (pos == -1) return search(rs, mid + 1, R, mid + 1, r, k);
78         return pos;
79     }
80 }

```

```
81 #undef ls
82 #undef rs
```

## temp2

```
1 struct Info {
2     i64 mx;
3     Info() : mx(111 << 60) {}
4     Info(i64 mx) : mx(mx) {}
5 };
6 Info operator + (const Info &a, const Info &b) {
7     return {max(a.mx, b.mx)};
8 }
9 template<class Info, class Merge = plus<Info>>
10 struct SegmentTree {
11     const int n;
12     const Merge merge;
13     vector<Info> info;
14     SegmentTree(int n) : n(n), merge(Merge()), info((n << 2) + 1) {}
15     SegmentTree (vector<Info> init) : SegmentTree((int)init.size()) {
16         function<void(int, int, int)> build = [&] (int p, int l, int r) {
17             if (l == r) {
18                 info[p] = init[l - 1];
19                 return;
20             }
21             int m = (l + r) / 2;
22             build(2 * p, l, m); build(2 * p + 1, m + 1, r);
23             pull(p);
24         };
25         build(1, 1, n);
26     }
27     void pull (int p) {
28         info[p] = merge(info[2 * p], info[2 * p + 1]);
29     }
30     void modify (int p, int l, int r, int x, const Info &v) {
31         if (l == r) {
32             info[p] = v;
33             return;
34         }
35         int m = (l + r) / 2;
36         if (x <= m) {
37             modify(2 * p, l, m, x, v);
38         } else {
39             modify(2 * p + 1, m + 1, r, x, v);
40         }
41         pull(p);
42     }
43     void modify (int p, const Info &v) {
44         modify(1, 1, n, p, v);
45     }
46     Info query (int p, int l, int r, int x, int y) {
47         if (l > y || r < x) {
48             return Info();
49         }
50         if (l >= x && r <= y) {
51             return info[p];
52         }
53         int m = (l + r) / 2;
54         return merge(query(2 * p, l, m, x, y), query(2 * p + 1, m + 1, r, x, y));
55     }
56 }
```

```

56     Info query (int l, int r) {
57         return query(1, 1, n, l, r);
58     }
59 };

```

## temp3

```

1  struct Info {
2      int mx;
3      Info() : mx(0) {}
4      Info(int mx) : mx(mx) {}
5  };
6  using Tag = int;
7  Info operator + (const Info &a, const Info &b) {
8      return {a.mx + b.mx};
9  }
10 void apply (Info &x, Tag &a, Tag f) {
11     x.mx += f;
12     a += f;
13 }
14
15
16 template<class Info, class Tag, class Merge = plus<Info>>
17 struct LazySegmentTree {
18     const int n;
19     const Merge merge;
20     vector<Info> info;
21     vector<Tag> tag;
22     LazySegmentTree(int n) : n(n), merge(Merge()), info((n << 2) + 1), tag((n << 2) + 1)
23 {}
24     LazySegmentTree (vector<Info> init) : LazySegmentTree((int)init.size()) {
25         function<void(int, int, int)> build = [&] (int p, int l, int r) {
26             if (l == r) {
27                 info[p] = init[l - 1];
28                 return;
29             }
30             int m = (l + r) / 2;
31             build(2 * p, l, m); build(2 * p + 1, m + 1, r);
32             pull(p);
33         };
34         build(1, 1, n);
35     }
36     void pull (int p) {
37         info[p] = merge(info[2 * p], info[2 * p + 1]);
38     }
39     void apply (int p, const Tag &v) {
40         ::apply(info[p], tag[p], v);
41     }
42     void push(int p) {
43         apply(2 * p, tag[p]);
44         apply(2 * p + 1, tag[p]);
45         tag[p] = Tag();
46     }
47     void modify (int p, int l, int r, int x, const Info &v) {
48         if (l == r) {
49             ::apply(info[p], tag[p], v.mx);
50             return;
51         }
52         int m = (l + r) / 2;
53         push(p);

```

```

53     if (x <= m) {
54         modify(2 * p, l, m, x, v);
55     } else {
56         modify(2 * p + 1, m + 1, r, x, v);
57     }
58     pull(p);
59 }
60 void modify (int p, const Info &v) {
61     modify(1, 1, n, p, v);
62 }
63 void modify(int p, int l, int r, int x, int y, const Tag &v){
64     if (l > y || r < x) {
65         return;
66     }
67     if (l >= x && r <= y) {
68         apply(p, v);
69         return;
70     }
71     int m = (l + r) / 2;
72     push(p);
73     modify(2 * p, l, m, x, y, v);
74     modify(2 * p + 1, m + 1, r, x, y, v);
75     pull(p);
76 }
77 void modify(int l, int r, const Tag &v){
78     return modify(1, 1, n, l, r, v);
79 }
80 Info query (int p, int l, int r, int x, int y) {
81     if (l > y || r < x) {
82         return Info();
83     }
84     if (l >= x && r <= y) {
85         return info[p];
86     }
87     int m = (l + r) / 2;
88     push(p);
89     return merge(query(2 * p, l, m, x, y), query(2 * p + 1, m + 1, r, x, y));
90 }
91 Info query (int l, int r) {
92     return query(1, 1, n, l, r);
93 }
94 int find_first (int p, int l, int r, int L, int R, const function<bool(const Info&)>
&f, Info &pre) {
95     if (l < R || r < L) {
96         return r + 1;
97     }
98     if (l <= L && r <= R) {
99         if (!f(merge(pre, info[p]))) {
100             pre = merge(pre, info[p]);
101             return r + 1;
102         }
103         if (l == r) return r;
104         int m = (l + r) / 2;
105         push(p);
106         int res;
107         if (f(merge(pre, info[2 * p]))) {
108             res = find_first(2 * p, l, m, L, R, f, pre);
109         } else {
110             pre = merge(pre, info[2 * p]);
111             res = find_first(2 * p + 1, r, L, R, f, pre);
112         }
113         return res;

```

```

114     }
115     int m = (l + r) / 2;
116     push(p);
117     int res = m + 1;
118     if (L <= m) {
119         res = find_first(2 * p, l, m, L, R, f, pre);
120     }
121     if (R > m && res == m + 1) {
122         res = find_first(2 * p + 1, m + 1, r, L, R, f, pre);
123     }
124     return res;
125 }
126 int find_first(int l, int r, const function<bool(const Info&> &f){
127     Info pre = Info();
128     return find_first(1, 1, n, l, r, f, pre);
129 }
130 int find_last(int p, int l, int r, int L, int R, const function<bool(const Info&>
&f, Info &suf){
131     if (l > R || r < L){
132         return l - 1;
133     }
134     if (l >= L && r <= R){
135         if (!f(merge(info[p], suf))){
136             suf = merge(info[p], suf);
137             return l - 1;
138         }
139         if (l == r) return r;
140         int m = (l + r) / 2;
141         push(p);
142         int res;
143         if (f(merge(info[2 * p + 1], suf))){
144             res = find_last(2 * p + 1, m + 1, r, L, R, f, suf);
145         }
146         else{
147             suf = merge(info[2 * p + 1], suf);
148             res = find_last(2 * p, l, m, L, R, f, suf);
149         }
150         return res;
151     }
152     int m = (l + r) / 2;
153     push(p);
154     int res = m;
155     if (R > m){
156         res = find_last(2 * p + 1, m + 1, r, L, R, f, suf);
157     }
158     if (L <= m && res == m){
159         res = find_last(2 * p, l, m, L, R, f, suf);
160     }
161     return res;
162 }
163 int find_last(int l, int r, const function<bool(const Info&> &f){
164     Info suf = Info();
165     return find_last(1, 1, n, l, r, f, suf);
166 }
167 };
168

```

```

1 // pushup
2 void pushup1(int u) {
3     tr[u].sum = tr[u << 1].sum + tr[u << 1 | 1].sum;
4 } // 父节点的区间和等于左右儿子的区间和。
5 void pushup2(int u) {

```

```

6     tr[u].v = max(tr[u << 1].v, tr[u << 1 | 1].v);
7 } // 父节点的最大值等于左右儿子的最大值。
8
9
10 // pushdown
11 void pushdown(int u) // 例为维护区间加的lazytag
12 {
13     auto &r = tr[u], &left = tr[u << 1], &right = tr[u << 1 | 1];
14     if (r.add){
15         left.add += r.add, left.sum += (left.r - left.l + 1) * r.add;
16         right.add += r.add, right.sum += (right.r - right.l + 1) * r.add;
17         r.add = 0;
18     }
19 }
20
21 // build
22 void build (int u, int l, int r) {
23     if (l == r) tr[u] = {l, r, q[r]};
24     else {
25         tr[u] = {l, r, q[r], 0}; // 该节点的左端点, 右端点, 以及维护的值和lazytag (可以为1)
26         int mid = l + r >> 1;
27         build(u << 1, l, mid); // 向左下递归建树
28         build(u << 1 | 1, mid + 1, r); // 向右下递归建树
29         pushup(u); // 最后自底向上更新节点所维护的值
30     }
31 }
32
33
34 //modify
35
36 // 1.单点修改
37 void modify1(int u, int x, int v) // x为待修改的数的下标, v为修改后的值
38 {
39     if (tr[u].l == x && tr[u].r == x) tr[u].v = v; // 修改
40     else{ // 未达到就继续递归
41         int mid = tr[u].l + tr[u].r >> 1;
42         if (x <= mid) modify(u << 1, x, v);
43         else modify(u << 1 | 1, x, v);
44         pushup(u); // 修改后自底向上更新一下。
45     }
46 }
47
48 // 2.区间修改 (需要lazytag)
49 void modify2(int u, int l, int r, int d) {
50     if (tr[u].l >= l && tr[u].r <= r) {
51         tr[u].sum += (tr[u].r - tr[u].l + 1) * d;
52         tr[u].add += d;
53     }
54     else {
55         pushdown(u);
56         int mid = tr[u].l + tr[u].r >> 1;
57         if (l <= mid) modify(u << 1, l, r, d);
58         if (r > mid) modify(u << 1 | 1, l, r, d);
59         pushup(u);
60     }
61 }
62
63
64 //query
65 int query(int u, int l, int r)
66 {
67     if (tr[u].l >= l && tr[u].r <= r) return tr[u].sum;

```

```

68
69     pushdown(u); // 如果有则加
70     int mid = tr[u].l + tr[u].r >> 1;
71     int sum = 0;
72     if (l <= mid) sum += query(u << 1, l, r);
73     if (r > mid) sum += query(u << 1 | 1, l, r);
74     return sum;
75 }

```

## 1.5.1 动态开点

```

1  int tot = 1;
2  struct node {
3      int ls, rs;
4      int val, tag;
5  }tr[maxn << 2];
6  int root = 1;
7  il void pushup (int u) {
8      tr[u].val = tr[tr[u].ls].val + tr[tr[u].rs].val;
9  }
10
11 il void pushtag (int &u, int l, int r, int k) {
12     if (!u) u = ++ tot;
13     tr[u].val += (r - l + 1) * k;
14     tr[u].tag += k;
15 }
16
17 il void build (int &u, int l, int r) {
18     if (!u) u = ++ tot;
19     if (l == r) {
20         tr[u].val = q[l];
21         return;
22     }
23     int mid = l + r >> 1;
24     build(tr[u].ls, l, mid);
25     build(tr[u].rs, mid + 1, r);
26     pushup(u);
27 }
28
29 il void pushdown (int u, int l, int r) {
30     if (tr[u].tag) {
31         int mid = l + r >> 1;
32         pushtag(tr[u].ls, l, mid, tr[u].tag);
33         pushtag(tr[u].rs, mid + 1, r, tr[u].tag);
34         tr[u].tag = 0;
35     }
36 }
37
38 il void update (int &u, int l, int r, int L, int R, int k) {
39     if (!u) u = ++ tot;
40     if (l >= L && r <= R) {
41         pushtag(u, l, r, k);
42         return;
43     }
44     pushdown(u, l, r);
45     int mid = l + r >> 1;
46     if (L <= mid) update(tr[u].ls, l, mid, L, R, k);
47     if (R > mid) update(tr[u].rs, mid + 1, r, L, R, k);
48     pushup(u);
49 }

```

```

50
51 int ask (int u, int l, int r, int L, int R) {
52     if (!u) return 0;
53     if (L <= l && r <= R) return tr[u].val;
54     pushdown(u, l, r);
55     int sum = 0;
56     int mid = l + r >> 1;
57     if (L <= mid) sum += ask(tr[u].ls, l, mid, L, R);
58     if (R > mid) sum += ask(tr[u].rs, mid + 1, r, L, R);
59     return sum;
60 }
61

```

## 1.6 对顶堆

```

1  int q[i];
2
3  priority_queue<int> bg; // 大根堆
4  priority_queue<int, vector<int>, greater<int>> ss; // 小根堆
5
6  //bg.push(q[1]);
7
8  int mid = q[1];
9  cout << mid << endl;
10 for (int i = 2; i <= n; i++) {
11     if (q[i] > mid) ss.push(q[i]);
12     else bg.push(q[i]);
13
14     if (i & 1) { // 当i为奇数输出中位数
15         while (bg.size() != ss.size()) {
16             if (bg.size() > ss.size()) {
17                 ss.push(bg.top());
18                 bg.pop();
19             }
20             else {
21                 bg.push(ss.top());
22                 ss.pop();
23             }
24         }
25         cout << mid << endl;
26     }
27 }
28
29

```

## 1.7 主席树

```

1  constexpr int N = 1e5 + 10;
2  int q[N], rt[N], aft[N];
3  int cnt;
4  struct Chairman_Tree {
5      struct Node {
6          int l, r, val;
7      } tree[N * 500];
8      void init() {
9          memset(rt, 0, sizeof rt);
10         cnt = 0;
11     }
12     int buildT0(int l, int r) {

```



```

13     int k = cnt++;
14     tree[k].val = 0;
15     if (l == r) return k;
16     int mid = l + r >> 1;
17     tree[k].l = buildT0(l, mid);
18     tree[k].r = buildT0(mid + 1, r);
19     return k;
20 }
21 void pushup(int u) {
22     tree[u].val = tree[tree[u].l].val + tree[tree[u].r].val;
23 }
24 int update(int P, int l, int r, int ppos, int del) {
25     int k = cnt++;
26     if (l == r) {
27         tree[k].val = tree[P].val + del;
28     } else {
29         int mid = l + r >> 1;
30         if (ppos <= mid) {
31             tree[k].l = update(tree[P].l, l, mid, ppos, del);
32             tree[k].r = tree[P].r;
33         } else {
34             tree[k].l = tree[P].l;
35             tree[k].r = update(tree[P].r, mid + 1, r, ppos, del);
36         }
37         pushup(k);
38     }
39     return k;
40 }
41 int query(int x, int y, int l, int r, int k) {
42     if (l == r) {
43         return l;
44     }
45     int mid = l + r >> 1;
46     int res = tree[tree[y].l].val - tree[tree[x].l].val;
47     if (res >= k) {
48         return query(tree[x].l, tree[y].l, l, mid, k);
49     } else {
50         return query(tree[x].r, tree[y].r, mid + 1, r, k - res);
51     }
52 }
53 } T;
54
55 void solve() {
56     scanf("%lld%lld", &n, &m);
57     vector<int> a;
58     for (int i = 1; i <= n; i++) {
59         scanf("%lld", &q[i]);
60         a.pb(q[i]);
61     }
62     sort(all(a)); a.erase(unique(all(a)), a.end());
63     for (int i = 1; i <= n; i++) {
64         q[i] = lower_bound(all(a), q[i]) - a.begin() + 1;
65         // de(aft[i]);
66     }
67     tree.init();
68     rt[0] = tree.buildT0(1, n);
69     for (int i = 1; i <= n; i++) {
70         rt[i] = tree.update(rt[i - 1], 1, n, q[i], 1);
71     }
72     while (m--) {
73         int l, r, k; scanf("%lld%lld%lld", &l, &r, &k);
74         printf("%lld\n", a[tree.query(rt[l - 1], rt[r], 1, n, k) - 1]);

```

```
75     }
76 }
```

## 1.8 Treap

### 1.8.1 旋转Treap

```
1  int tt;
2  int n, m, k;
3  int now, root;
4  int sz[maxn], key[maxn], cnt[maxn], rd[maxn], son[maxn][2];
5
6  inline void push_up(int x) {
7      sz[x] = sz[son[x][0]] + sz[son[x][1]] + cnt[x];
8  }
9
10 inline void rotate (int &x, int y) {
11     int ii = son[x][y ^ 1];
12     son[x][y ^ 1] = son[ii][y];
13     son[ii][y] = x;
14     push_up(x), push_up(ii);
15     x = ii;
16 }
17
18 void insert (int &u, int x) {
19     if (!u) {
20         u = ++ now;
21         sz[u] = cnt[u] = 1;
22         key[u] = x;
23         rd[u] = rand();
24         return;
25     }
26     if (key[u] == x) {
27         cnt[u] ++ ;
28         sz[u] ++ ;
29         return;
30     }
31     int d = (x > key[u]);
32     insert(son[u][d], x);
33     if (rd[u] < rd[son[u][d]]) {
34         rotate(u, d ^ 1);
35     }
36     push_up(u);
37 }
38
39 void del (int &u, int x) {
40     if (!u) return;
41     if (x != key[u]) {
42         del (son[u][x > key[u]], x);
43     }
44     else {
45         if (!son[u][0] && !son[u][1]) {
46             cnt[u] -- ;
47             sz[u] -- ;
48             if (cnt[u] == 0) u = 0;
49         }
50         else if (son[u][0] && !son[u][1]) {
51             rotate(u, 1);
52             del (son[u][1], x);
53         }
54         else if (!son[u][0] && son[u][1]) {
```

```

55         rotate(u, 0);
56         del (son[u][0], x);
57     }
58     else {
59         int d = rd[son[u][0]] > rd[son[u][1]];
60         rotate(u, d);
61         del(son[u][d], x);
62     }
63 }
64 push_up(u);
65 }
66
67 int get_rank(int u, int x) { // 得到排名
68     if (!u) return 0;
69     if (key[u] == x) return sz[son[u][0]] + 1;
70     if (key[u] < x) return sz[son[u][0]] + cnt[u] + get_rank(son[u][1], x);
71     return get_rank(son[u][0], x);
72 }
73
74 int find (int u, int x) { // 查询排名
75     if (!u) return 0;
76     if (sz[son[u][0]] >= x) {
77         return find(son[u][0], x);
78     }
79     else if (sz[son[u][0]] + cnt[u] < x) {
80         return find(son[u][1], x - cnt[u] - sz[son[u][0]]);
81     }
82     else return key[u];
83 }
84
85 int pre (int u, int x) { // 查询前驱
86     if (!u) return -inf;
87     if (key[u] >= x) {
88         return pre(son[u][0], x);
89     }
90     else return max(key[u], pre(son[u][1], x));
91 }
92
93 int suf (int u, int x) { // 查询后继
94     if (!u) return inf;
95     if (key[u] <= x) {
96         return suf(son[u][1], x);
97     }
98     else return min(key[u], suf(son[u][0], x));
99 }
100
101 void solve() {
102     int query;
103     cin >> query;
104     while (query -- ) {
105         int op, x;
106         cin >> op >> x;
107         if (op == 1) insert(root, x);
108         if (op == 2) del(root, x);
109         if (op == 3) cout << get_rank(root, x) << endl;
110         if (op == 4) cout << find(root, x) << endl;
111         if (op == 5) cout << pre(root, x) << endl;
112         if (op == 6) cout << suf(root, x) << endl;
113     }
114 }

```

## 1.8.2 Fhq Treap

```
1  mt19937 rnd(233);
2  int root, idx;
3  int x, y, z;
4  struct fhq {
5      int l, r;
6      int key, val; // key权值, val堆值
7      int size;
8  }tr[maxn];
9
10 inline int get_node (int key) {
11     tr[ ++ idx].key = key;
12     tr[idx].val = rnd();
13     tr[idx].size = 1;
14     return idx;
15 }
16
17 inline void pushup (int u) {
18     tr[u].size = tr[tr[u].l].size + tr[tr[u].r].size + 1;
19 }
20
21 inline void split (int u, int k, int &x, int &y) {
22     if (!u) x = y = 0;
23     else {
24         if (tr[u].key <= k) {
25             x = u;
26             split(tr[u].r, k, tr[u].r, y);
27         }
28         else {
29             y = u;
30             split(tr[u].l, k, x, tr[u].l);
31         }
32         pushup(u);
33     }
34 }
35
36 inline int merge (int x, int y) {
37     if (!x || !y) return x + y;
38     if (tr[x].val > tr[y].val) {
39         tr[x].r = merge(tr[x].r, y);
40         pushup(x); return x;
41     }
42     else {
43         tr[y].l = merge(x, tr[y].l);
44         pushup(y); return y;
45     }
46 }
47
48 inline void insert (int k) {
49     split (root, k, x, y);
50     root = merge(merge(x, get_node(k)), y);
51 }
52
53 inline void del (int k) {
54     split (root, k, x, z);
55     split (x, k - 1, x, y);
56     y = merge(tr[y].l, tr[y].r);
57     root = merge(merge(x, y), z);
58 }
59
60 inline int get_rank (int k) {
```

```

61     split (root, k - 1, x, y);
62     k = tr[x].size + 1;
63     root = merge(x, y);
64     return k;
65 }
66
67 inline int get_key (int k) {
68     int p = root;
69     while (p) {
70         if (tr[tr[p].l].size + 1 == k) {
71             break;
72         }
73         else if (tr[tr[p].l].size >= k) {
74             p = tr[p].l;
75         }
76         else {
77             k -= tr[tr[p].l].size + 1;
78             p = tr[p].r;
79         }
80     }
81     return tr[p].key;
82 }
83
84 inline int pre (int k) {
85     split (root, k - 1, x, y);
86     int p = x;
87     while (tr[p].r) p = tr[p].r;
88     k = tr[p].key;
89     root = merge(x, y);
90     return k;
91 }
92
93 inline int suf (int k) {
94     split (root, k, x, y);
95     int p = y;
96     while (tr[p].l) p = tr[p].l;
97     k = tr[p].key;
98     root = merge(x, y);
99     return k;
100 }
101
102 void solve() {
103     read(n);
104     for (int i = 1; i <= n; i ++ ) {
105         int op, x; read(op); read(x);
106         if (op == 1) insert(x);
107         else if (op == 2) del(x);
108         else if (op == 3) printf("%lld\n", get_rank(x));
109         else if (op == 4) printf("%lld\n", get_key(x));
110         else if (op == 5) printf("%lld\n", pre(x));
111         else printf("%lld\n", suf(x));
112     }
113 }
114
115
116 // 区间反转版本
117 mt19937 rnd(233);
118 int root, idx;
119 int x, y, z;
120 struct fhq {
121     int l, r;
122     int key, val; // key权值, val堆值

```

```

123     int size;
124     int tag;
125 }tr[maxn];
126
127 inline int get_node (int key) {
128     tr[ ++ idx].key = key;
129     tr[idx].val = rnd();
130     tr[idx].size = 1;
131     tr[idx].tag = 0;
132     return idx;
133 }
134
135 inline void pushup (int u) {
136     tr[u].size = tr[tr[u].l].size + tr[tr[u].r].size + 1;
137 }
138
139 inline void pushdown (int u) {
140     if (tr[u].tag) {
141         swap(tr[u].l, tr[u].r);
142         tr[tr[u].l].tag ^= 1, tr[tr[u].r].tag ^= 1;
143         tr[u].tag = 0;
144     }
145 }
146
147 inline void split (int u, int k, int &x, int &y) {
148     if (!u) x = y = 0;
149     else {
150         pushdown(u);
151         if (tr[tr[u].l].size + 1 <= k) {
152             x = u;
153             split(tr[u].r, k - tr[tr[u].l].size - 1, tr[u].r, y);
154         }
155         else {
156             y = u;
157             split(tr[u].l, k, x, tr[u].l);
158         }
159         pushup(u);
160     }
161 }
162
163 inline int merge (int x, int y) {
164     if (!x || !y) return x + y;
165     if (tr[x].val > tr[y].val) {
166         pushdown(x);
167         tr[x].r = merge(tr[x].r, y);
168         pushup(x); return x;
169     }
170     else {
171         pushdown(y);
172         tr[y].l = merge(x, tr[y].l);
173         pushup(y); return y;
174     }
175 }
176
177 inline void insert (int k) {
178     split (root, k, x, y);
179     root = merge(merge(x, get_node(k)), y);
180 }
181
182 inline void del (int k) {
183     split (root, k, x, z);
184     split (x, k - 1, x, y);

```

```

185     y = merge(tr[y].l, tr[y].r);
186     root = merge(merge(x, y), z);
187 }
188
189 inline int get_rank (int k) {
190     split (root, k - 1, x, y);
191     k = tr[x].size + 1;
192     root = merge(x, y);
193     return k;
194 }
195
196 inline int get_key (int k) {
197     int p = root;
198     while (p) {
199         if (tr[tr[p].l].size + 1 == k) {
200             break;
201         }
202         else if (tr[tr[p].l].size >= k) {
203             p = tr[p].l;
204         }
205         else {
206             k -= tr[tr[p].l].size + 1;
207             p = tr[p].r;
208         }
209     }
210     return tr[p].key;
211 }
212
213 inline int pre (int k) {
214     split (root, k - 1, x, y);
215     int p = x;
216     while (tr[p].r) p = tr[p].r;
217     k = tr[p].key;
218     root = merge(x, y);
219     return k;
220 }
221
222 inline int suf (int k) {
223     split (root, k, x, y);
224     int p = y;
225     while (tr[p].l) p = tr[p].l;
226     k = tr[p].key;
227     root = merge(x, y);
228     return k;
229 }
230
231 inline void print (int u) {
232     if (!u) return;
233     pushdown(u);
234     print(tr[u].l); printf("%lld ", tr[u].key); print(tr[u].r);
235 }
236
237 void solve() {
238     read(n), read(m);
239     // insert(-inf); insert(inf);
240     for (int i = 1; i <= n; i++) insert(i);
241
242     while (m--) {
243         int l, r; read(l); read(r);
244         split (root, l - 1, x, y);
245         split (y, r - l + 1, y, z);
246         tr[y].tag ^= 1;

```

```

247         root = merge(x, merge(y, z));
248     }
249     print(root);
250 }

```

## 1.9 轻重链剖分

```

1  int h[maxn], ne[maxn], e[maxn], idx;
2  int w[maxn], wt[maxn], id[maxn], top[maxn], fa[maxn], depth[maxn], sz[maxn], son[maxn];
3  int tt;
4  int n, m, k;
5  int root, mod, cnt;
6
7  struct node {
8      int l, r;
9      int sum, add;
10 }tr[maxn << 2];
11
12 inline void add (int a, int b) {
13     e[idx] = b, ne[idx] = h[a], h[a] = idx ++ ;
14 }
15
16 void dfs1 (int u, int f) {
17     depth[u] = depth[f] + 1;
18     fa[u] = f;
19     sz[u] = 1;
20     int mxson = -1;
21     for (int i = h[u]; ~i; i = ne[i]) {
22         int j = e[i];
23         if (j != f) {
24             dfs1(j, u);
25             sz[u] += sz[j];
26             if (sz[j] > mxson) {
27                 mxson = sz[j];
28                 son[u] = j;
29             }
30         }
31     }
32 }
33
34 void dfs2 (int u, int topf) {
35     id[u] = ++ cnt;
36     wt[cnt] = w[u];
37     top[u] = topf;
38     if (!son[u]) return;
39     dfs2(son[u], topf);
40     for (int i = h[u]; ~i; i = ne[i]) {
41         int j = e[i];
42         if (!id[j]) dfs2(j, j);
43     }
44 }
45
46 inline void pushup (int u) {
47     tr[u].sum = (tr[u << 1].sum + tr[u << 1 | 1].sum) % mod;
48 }
49
50 inline void pushdown (int u) {
51     auto &r = tr[u], &left = tr[u << 1], &right = tr[u << 1 | 1];

```



```

52     if (r.add) {
53         right.add = (right.add + r.add) % mod;
54         right.sum = (right.sum + (right.r - right.l + 1) * r.add) % mod;
55         left.add = (left.add + r.add) % mod;
56         left.sum = (left.sum + (left.r - left.l + 1) * r.add) % mod;
57         r.add = 0;
58     }
59 }
60
61 inline void build (int u, int l, int r) {
62     if (l == r) {
63         tr[u] = {l, r, wt[r], 0};
64         return;
65     }
66     tr[u] = {l, r, 0, 0};
67     int mid = (l + r) >> 1;
68     build(u << 1, l, mid);
69     build(u << 1 | 1, mid + 1, r);
70     pushup(u);
71 }
72
73 inline void modify (int u, int l, int r, int d) {
74     if (tr[u].l >= l && tr[u].r <= r) {
75         tr[u].sum = (tr[u].sum + (tr[u].r - tr[u].l + 1) * d) % mod;
76         tr[u].add += d;
77     }
78     else {
79         pushdown(u);
80         int mid = tr[u].r + tr[u].l >> 1;
81         if (l <= mid) modify (u << 1, l, r, d);
82         if (r > mid) modify (u << 1 | 1, l, r, d);
83         pushup(u);
84     }
85 }
86
87 inline int query (int u, int l, int r) {
88     if (tr[u].l >= l && tr[u].r <= r) {
89         return tr[u].sum;
90     }
91     pushdown(u);
92     int mid = tr[u].l + tr[u].r >> 1;
93     int sum = 0;
94     if (l <= mid) sum = (sum + query(u << 1, l, r)) % mod;
95     if (r > mid) sum = (sum + query(u << 1 | 1, l, r)) % mod;
96     return sum;
97 }
98
99 inline void Uprange (int x, int y, int d) {
100     while (top[x] != top[y]) {
101         if (depth[top[x]] < depth[top[y]]) swap(x, y);
102         modify(1, id[top[x]], id[x], d);
103         x = fa[top[x]];
104     }
105     if (depth[x] > depth[y]) swap(x, y);
106     modify(1, id[x], id[y], d);
107 }
108
109 inline int Qrange (int x, int y) {
110     int sum = 0;
111     while (top[x] != top[y]) {
112         if (depth[top[x]] < depth[top[y]]) swap(x, y);
113         sum = (sum + query(1, id[top[x]], id[x])) % mod;

```

```

114     x = fa[top[x]];
115 }
116 if (depth[x] > depth[y]) swap(x, y);
117 sum = (sum + query(1, id[x], id[y])) % mod;
118 return sum;
119 }
120
121 inline int lca (int x, int y) {
122     while (top[x] != top[y]) {
123         depth[top[x]] > depth[top[y]] ? x = fa[top[x]] : y = fa[top[y]];
124     }
125     return depth[x] < depth[y] ? x : y;
126 }
127
128 void solve() {
129     scanf("%d%d%d", &n, &m, &root, &mod);
130     for (int i = 1; i <= n; i++) scanf("%d", w + i);
131     ms(h, -1);
132     for (int i = 1; i < n; i++) {
133         int u, v; scanf("%d%d", &u, &v);
134         add(u, v); add(v, u);
135     }
136     dfs1(root, 0);
137     dfs2(root, root);
138     build(1, 1, n);
139     for (int i = 1; i <= m; i++) {
140         int op;
141         scanf("%d", &op);
142         if (op == 1) {
143             int l, r, d;
144             scanf("%d%d%d", &l, &r, &d);
145             uprange(l, r, d);
146         }
147         if (op == 2) {
148             int l, r;
149             scanf("%d%d", &l, &r);
150             printf("%d\n", Qrange(l, r));
151         }
152         if (op == 3) {
153             int x, d;
154             scanf("%d%d", &x, &d);
155             modify(1, id[x], id[x] + sz[x] - 1, d % mod);
156         }
157         if (op == 4) {
158             int x;
159             scanf("%d", &x);
160             printf("%d\n", query(1, id[x], id[x] + sz[x] - 1));
161         }
162     }
163 }
164

```

边权:

```

1  int n, m;
2  vector<pair<int, int>> G[N];
3  int w[N], wt[N], id[N], top[N], fa[N], depth[N], sz[N], son[N];
4  int root, cnt;
5  struct node {
6      int l, r;

```

```

7   int sum, add;
8   }tr[N << 2];
9   void dfs1 (int u, int f) {
10      depth[u] = depth[f] + 1;
11      fa[u] = f;
12      sz[u] = 1;
13      int mxson = -1;
14      for (auto [j, w] : G[u]) {
15          if (j != f) {
16              dfs1(j, u);
17              sz[u] += sz[j];
18              if (sz[j] > mxson) {
19                  mxson = sz[j];
20                  son[u] = j;
21              }
22          }
23      }
24  }
25  void dfs2 (int u, int topf) {
26      id[u] = ++ cnt;
27      wt[cnt] = w[u];
28      top[u] = topf;
29      if (!son[u]) return;
30      dfs2(son[u], topf);
31      for (auto [j, w] : G[u]) {
32          if (!id[j]) dfs2(j, j);
33      }
34  }
35  void pushup (int u) {
36      tr[u].sum = (tr[u << 1].sum + tr[u << 1 | 1].sum);
37  }
38  void pushdown (int u) {
39      auto &r = tr[u], &left = tr[u << 1], &right = tr[u << 1 | 1];
40      if (r.add) {
41          right.add = left.add = r.add;
42          right.sum = left.sum = r.add;
43          r.add = 0;
44      }
45  }
46  void build (int u, int l, int r) {
47      if (l == r) {
48          tr[u] = {l, r, wt[r], 0};
49          return;
50      }
51      tr[u] = {l, r, 0, 0};
52      int mid = (l + r) >> 1;
53      build(u << 1, l, mid);
54      build(u << 1 | 1, mid + 1, r);
55      pushup(u);
56  }
57  void modify (int u, int l, int r, int d) {
58      if (tr[u].l >= l && tr[u].r <= r) {
59          tr[u].sum = (tr[u].r - tr[u].l + 1) * d;
60          tr[u].add = d;
61      }
62      else {
63          pushdown(u);
64          int mid = tr[u].r + tr[u].l >> 1;
65          if (l <= mid) modify (u << 1, l, r, d);
66          if (r > mid) modify (u << 1 | 1, l, r, d);
67          pushup(u);
68      }

```

```

69 }
70 int query (int u, int l, int r) {
71     if (tr[u].l >= l && tr[u].r <= r) {
72         return tr[u].sum;
73     }
74     pushdown(u);
75     int mid = tr[u].l + tr[u].r >> 1;
76     int sum = 0;
77     if (l <= mid) sum += query(u << 1, l, r);
78     if (r > mid) sum += query(u << 1 | 1, l, r);
79     return sum;
80 }
81 void Uprange (int x, int y, int d) {
82     while (top[x] != top[y]) {
83         if (depth[top[x]] < depth[top[y]]) swap(x, y);
84         modify(1, id[top[x]], id[x], d);
85         x = fa[top[x]];
86     }
87     if (x == y) return;
88     if (depth[x] > depth[y]) swap(x, y);
89     modify(1, id[x] + 1, id[y], d);
90 }
91
92 int Qrange (int x, int y) {
93     int sum = 0;
94     while (top[x] != top[y]) {
95         if (depth[top[x]] < depth[top[y]]) swap(x, y);
96         sum += query(1, id[top[x]], id[x]);
97         x = fa[top[x]];
98     }
99     if (x == y) return sum;
100    if (depth[x] > depth[y]) swap(x, y);
101    sum += query(1, id[x] + 1, id[y]);
102    return sum;
103 }
104 void Solve() {
105     cin >> n;
106     vector<pair<int, int>> edges(n);
107     for (int i = 1; i < n; i++) {
108         int u, v, w; cin >> u >> v >> w;
109         G[u].emplace_back(v, w); G[v].emplace_back(u, w);
110         edges[i] = {u, v};
111     }
112     function<void(int, int)> dfs = [&] (int u, int fa) {
113         for (auto [j, v] : G[u]) {
114             if (j != fa) {
115                 w[j] = v;
116                 dfs(j, u);
117             }
118         }
119     };
120     dfs(1, 0); dfs1(1, 0); dfs2(1, 1);
121     build(1, 1, n);
122     cin >> m;
123     while (m--) {
124         int op; cin >> op;
125         if (op == 1) {
126             int l, r; cin >> l >> r;
127             int sum = Qrange(l, r);
128             cout << sum << '\n';
129         } else {
130             int id, d; cin >> id >> d;

```

```

131     int l = edges[id].first, r = edges[id].second;
132     uprange(l, r, d);
133 }
134 }
135 }

```

## 1.10 Dsu on Tree

```

1  /*
2  dsu on tree的核心是对暴力( $n^2$ )的优化
3  暴力( $n^2$ )是以每个节点当根暴力统计，dsu on tree可以对每个节点，只暴力统计轻儿子，在统计时少算一次重儿
   子。
4  */
5  // 处理重儿子
6  void dfs (int u, int fa) {
7      sz[u] = 1;
8      for (auto j : G[u]) {
9          if (j != fa) {
10             dfs(j, u);
11             sz[u] += sz[j];
12             if (sz[j] > sz[son[u]]) son[u] = j;
13         }
14     }
15 }
16
17 // 这一步需要结合题意
18 void count (int u, int fa, int v) {
19     cnt[val[u]] += v; // 增加或删除贡献
20     if (cnt[val[u]] > mx) {
21         mx = cnt[val[u]];
22         sum = val[u];
23     }
24     else if (cnt[val[u]] == mx) {
25         sum += val[u];
26     }
27
28     // 统计除标记外的重儿子的所有子树的贡献
29     for (auto j : G[u]) {
30         if (j == fa || j == flag) continue;
31         count(j, u, v);
32     }
33 }
34
35 void dfs (int u, int fa, bool op) {
36     // 1. 计算子树答案，先轻后重。
37     for (auto j : G[u]) {
38         if (j != fa && j != son[u]) {
39             dfs(j, u, false);
40         }
41     }
42     if (son[u]) {
43         dfs(son[u], u, true);
44         flag = son[u];
45     }
46     // 2. 计算当前节点答案，重儿子信息被保存，只算轻儿子。
47     count(u, fa, 1);
48     flag = 0;
49     ans[u] = sum;
50
51     // 把需要删除贡献的删一删

```

```

52     if (!op) {
53         count(u, fa, -1);
54         sum = mx = 0;
55     }
56 }
57
58
59 // HDU 6504
60 // calc max(子树中不同 + 其余不同)
61 const int maxn = 200010;
62 vector<int> G[maxn];
63 int sz[maxn], son[maxn];
64 int cnt1[maxn], cnt2[maxn], w[maxn];
65 int com, ans, flag;
66
67 il void dfs (int u, int fa) {
68     sz[u] = 1;
69     for (auto j : G[u]) {
70         if (j != fa) {
71             dfs(j, u);
72             sz[u] += sz[j];
73             if (sz[j] > sz[son[u]]) son[u] = j;
74         }
75     }
76 }
77
78 il void count (int u, int fa) {
79     cnt1[w[u]] -- ;
80     if (!cnt2[w[u]] && cnt1[w[u]]) com ++ ;
81     else if (!cnt1[w[u]] && cnt2[w[u]]) com -- ;
82     cnt2[w[u]] ++ ;
83
84     for (auto j : G[u]) {
85         if (j != fa && j != son[u]) {
86             count(j, u);
87         }
88     }
89 }
90
91 il void del (int u, int fa) {
92     cnt2[w[u]] -- ;
93     if (!cnt1[w[u]] && cnt2[w[u]]) com ++ ;
94     else if (!cnt2[w[u]] && cnt1[w[u]]) com -- ;
95     cnt1[w[u]] ++ ;
96
97     for (auto j : G[u]) {
98         if (j != fa) {
99             del(j, u);
100         }
101     }
102 }
103
104 il void dfs (int u, int fa, int op) {
105     for (auto j : G[u]) {
106         if (j != fa && j != son[u]) {
107             dfs(j, u, false);
108         }
109     }
110     if (son[u]) {
111         dfs(son[u], u, true);
112         flag = son[u];
113     }

```

```

114     count(u, fa);
115     flag = 0;
116     ans = max(ans, com);
117     if (!op) del(u, fa);
118 }
119
120 void solve() {
121     com = ans = flag = 0;
122     ms(son, 0); ms(cnt1, 0); ms(cnt2, 0);
123     for (int i = 1; i <= n; i++) G[i].clear();
124     for (int u = 2; u <= n; u++) {
125         int v; scanf("%d", &v);
126         G[u].pb(v); G[v].pb(u);
127     }
128     for (int i = 1; i <= n; i++) {
129         scanf("%d", &w[i]);
130         if (!cnt1[w[i]]) com++;
131         cnt1[w[i]]++;
132     }
133     dfs(1, -1);
134     dfs(1, -1, false);
135     printf("%d\n", ans);
136 }

```

## 1.11 Splay

```

1  #define ls(x) tr[x].ch[0]
2  #define rs(x) tr[x].ch[1]
3  #define fa(x) tr[x].fa
4  #define root tr[0].ch[1]
5  struct node {
6      int fa, ch[2], val, rec, sum, tag;
7  } tr[maxn];
8  int tot, pointnum;
9  void update (int x) {tr[x].sum = tr[ls(x)].sum + tr[rs(x)].sum + tr[x].rec;}
10 int ident (int x) {return tr[fa(x)].ch[0] == x ? 0 : 1;}
11 void connect (int x, int fa, int how) {tr[fa].ch[how] = x; tr[x].fa = fa;}
12
13
14 inline void pushdown (int u) {
15     if (tr[u].tag) {
16         tr[tr[u].ch[0]].tag ^= 1;
17         tr[tr[u].ch[1]].tag ^= 1;
18         swap(tr[u].ch[0], tr[u].ch[1]);
19         tr[u].tag = 0;
20     }
21 }
22
23 void rotate (int x) {
24     int y = fa(x), r = fa(y);
25     int yson = ident(x), rson = ident(y);
26     connect(tr[x].ch[yson ^ 1], y, yson);
27     connect(y, x, yson ^ 1);
28     connect(x, r, rson);
29     update(y), update(x);
30 }
31
32 void splay (int x, int to) {
33     to = fa(to);
34     while (fa(x) != to) {

```

```

35         int y = fa(x);
36         if (tr[y].fa == to) rotate(x);
37         else if (ident(x) == ident(y)) rotate(y), rotate(x);
38         else rotate(x), rotate(x);
39     }
40 }
41
42 int newNode (int v, int f) {
43     tr[ ++ tot].fa = f;
44     tr[tot].rec = tr[tot].sum = 1;
45     tr[tot].val = v;
46     return tot;
47 }
48
49 void insert (int x) {
50     int now = root;
51     if (!root) {newNode(x, 0); root = tot;}
52     else {
53         while (1) {
54             tr[now].sum ++ ;
55             if (tr[now].val == x) {tr[now].rec ++ ; splay(now, root);
return;}
56
57             int ne = x < tr[now].val ? 0 : 1;
58             if (!tr[now].ch[ne]) {
59                 int p = newNode(x, now);
60                 tr[now].ch[ne] = p;
61                 splay(p, root); return;
62             }
63             now = tr[now].ch[ne];
64         }
65     }
66 }
67
68 int find (int x) { // 找到值为x的某一结点
69     int now = root;
70     while (1) {
71         if (!now) return 0;
72         if (tr[now].val == x) {splay(now, root); return now;}
73         int ne = x < tr[now].val ? 0 : 1;
74         now = tr[now].ch[ne];
75     }
76 }
77
78 void del (int x) {
79     int p = find(x);
80     if (!p) return;
81     if (tr[p].rec > 1) {tr[p].rec -- , tr[p].sum -- ; return;}
82     else {
83         if (!tr[p].ch[0] && !tr[p].ch[1]) {root = 0; return;}
84         else if (!tr[p].ch[0]) {
85             root = tr[p].ch[1]; tr[root].fa = 0; return;
86         }
87         else {
88             int left = tr[p].ch[0];
89             while (tr[left].ch[1]) left = tr[left].ch[1];
90             splay(left, tr[p].ch[0]);
91             connect (tr[p].ch[1], left, 1);
92             connect (left, 0, 1);
93             update(left);
94         }
95     }

```



```

96 }
97
98 int rk (int x) { // x的排名
99     int now = root, ans = 0;
100     while (1) {
101         if (tr[now].val == x) return ans + tr[tr[now].ch[0]].sum + 1;
102         int ne = x < tr[now].val ? 0 : 1;
103         if (ne == 1) ans = ans + tr[tr[now].ch[0]].sum + tr[now].rec;
104         now = tr[now].ch[ne];
105     }
106 }
107
108 int kth (int x) { // x排名的数
109     int now = root;
110     while (1) {
111         int used = tr[now].sum - tr[tr[now].ch[1]].sum;
112         if (tr[tr[now].ch[0]].sum < x && x <= used) {
113             splay(now, root); return tr[now].val;
114             // return now; 用于区间反转
115         }
116         if (x < used) now = tr[now].ch[0];
117         else now = tr[now].ch[1], x -= used;
118     }
119 }
120
121 int pre (int x) {
122     int now = root, ans = -inf;
123     while (now) {
124         if (tr[now].val <= x) ans = max(ans, tr[now].val);
125         int ne = x <= tr[now].val ? 0 : 1;
126         now = tr[now].ch[ne];
127     }
128     return ans;
129 }
130
131 int suf (int x) {
132     int now = root, ans = inf;
133     while (now) {
134         if (tr[now].val >= x) ans = min(ans, tr[now].val);
135         int ne = x < tr[now].val ? 0 : 1;
136         now = tr[now].ch[ne];
137     }
138     return ans;
139 }
140
141 inline void reverse (int x, int y) { // 区间反转
142     int l = kth(x), r = kth(y + 2);
143     splay(l, 0); splay(r, l);
144     tr[tr[tr[root].ch[1]].ch[0]].tag ^= 1;
145 }
146
147 inline void print (int u) { // 中序遍历
148     pushdown(u);
149     if (tr[u].ch[0]) print(tr[u].ch[0]);
150     if (tr[u].val > 1 && tr[u].val < n + 2) printf("%11d ", tr[u].val - 1);
151     if (tr[u].ch[1]) print(tr[u].ch[1]);
152 }
153

```

## 1.12 莫队

```

1 // 普通莫队，add和del可以修改
2 int sq;
3 struct query {
4     int l, r, id;
5     bool operator < (const query &it) const {
6         if (l / sq != it.l / sq) return l < it.l;
7         if (l / sq & 1) return r < it.r;
8         return r > it.r;
9     }
10 }Q[1000010];
11 int q[1000010], ans[1000010], cnt[2000010], cur, l = 1, r;
12
13 inline void add (int x) {
14     if (!cnt[x]) cur ++ ;
15     cnt[x] ++ ;
16 }
17
18 inline void del (int x) {
19     cnt[x] -- ;
20     if (!cnt[x]) cur -- ;
21 }
22
23 void solve() {
24     read(n);
25     sq = sqrt(n);
26     for (int i = 1; i <= n; i ++ ) read(q[i]);
27     int query; read(query);
28     for (int i = 0; i < query; i ++ ) read(Q[i].l), read(Q[i].r), Q[i].id = i;
29     sort(Q, Q + query);
30
31     for (int i = 0; i < query; i ++ ) {
32         while (l > Q[i].l) add(q[ -- l]);
33         while (r < Q[i].r) add(q[ ++ r]);
34         while (l < Q[i].l) del(q[l ++ ]);
35         while (r > Q[i].r) del(q[r -- ]);
36         ans[Q[i].id] = cur;
37     }
38     for (int i = 0; i < query; i ++ ) printf("%d\n", ans[i]);
39 }
40
41
42 //带修莫队，块长可以选择pow(n*t, 0.33333) 或者 pow(n, 0.66666)
43 const int maxn = 500010;
44 int sq;
45 struct query {
46     int l, r, t, id;
47     bool operator < (const query &it) const {
48         if (l / sq != it.l / sq) return l < it.l;
49         else if (r / sq != it.r / sq) return r < it.r;
50         else return t < it.t;
51     }
52 }Q[maxn];
53 struct change {
54     int p, col;
55 }c[maxn];
56 int q[maxn], ans[maxn], cnt[maxn * 2], cur;
57 int l = 1, r = 0, qcnt, ccnt;
58
59 inline void add (int x) {
60     if (!cnt[x]) cur ++ ;
61     cnt[x] ++ ;
62 }

```

```

63 inline void del (int x) {
64     cnt[x] -- ;
65     if (!cnt[x]) cur -- ;
66 }
67 inline void work (int x, int ti) {
68     if (c[ti].p >= Q[x].l && c[ti].p <= Q[x].r) {
69         del(q[c[ti].p]);
70         add(c[ti].col);
71     }
72     swap(q[c[ti].p], c[ti].col);
73 }
74
75 void solve() {
76     read(n); read(m);
77     sq = pow(n, 0.66666); // 块长
78     for (int i = 1; i <= n; i ++ ) {
79         read(q[i]);
80     }
81     char op[10];
82     for (int i = 1; i <= m; i ++ ) {
83         scanf("%s", op);
84         if (op[0] == 'Q') {
85             qcnt ++ ;
86             read(Q[qcnt].l); read(Q[qcnt].r);
87             Q[qcnt].t = ccnt;
88             Q[qcnt].id = qcnt;
89         }
90         else {
91             ccnt ++ ;
92             read(c[ccnt].p); read(c[ccnt].col);
93         }
94     }
95     sort(Q + 1, Q + qcnt + 1);
96
97     int now = 0;
98     for (int i = 1; i <= qcnt; i ++ ) {
99         while (l > Q[i].l) add(q[ -- l]);
100        while (r < Q[i].r) add(q[ ++ r]);
101        while (l < Q[i].l) del(q[l ++ ]);
102        while (r > Q[i].r) del(q[r -- ]);
103
104        while (now < Q[i].t) work(i, ++ now);
105        while (now > Q[i].t) work(i, now --);
106
107        ans[Q[i].id] = cur;
108    }
109
110    for (int i = 1; i <= qcnt; i ++ ) printf("%d\n", ans[i]);
111 }

```

## 1.13 二维数点

```

1 // 离线 + 扫描线 + 树状数组
2 // https://codeforces.com/gym/103687/problem/F
3 int n, m, k;
4 const int maxn = 2e5 + 10;
5 struct tp {
6     static const int maxnum = 2e5 + 5;
7     static const int maxn = 2e5 + 5;
8     int tree[maxnum];
9     void update (int idx, int x) {

```

```

10         for (int pos = idx; pos < maxnum; pos += lowbit(pos)) tree[pos] += x;
11     }
12     int query (int n) {
13         int ans = 0;
14         for (int pos = n; pos; pos -= lowbit(pos)) ans += tree[pos];
15         return ans;
16     }
17     int n = 0, m = 0; // n点数, m询问矩形数
18     struct node1 {int x, y;} v[maxn]; // 点
19     struct node2 {int x, y, id, type;} q[maxn << 2]; // 询问矩形
20     static bool cmp1 (node1 &a, node1 &b) {return a.x < b.x;}
21     static bool cmp2 (node2 &a, node2 &b) {return a.x < b.x;}
22     int cnt = 0;
23     void add (int x, int y) { v[ ++ n].x = x, v[n].y = y; } // 添加点
24     void que (int x1, int y1, int x2, int y2, int i) { // 添加询问矩形, 左下角, 右上角,
        询问id
25         q[ ++ cnt] = {x2, y2, i, 1};
26         q[ ++ cnt] = {x1 - 1, y2, i, -1};
27         q[ ++ cnt] = {x2, y1 - 1, i, -1};
28         q[ ++ cnt] = {x1 - 1, y1 - 1, i, 1};
29         m ++ ;
30     }
31     void get (vector<int> &ans) { // 传入接受答案的数组
32         sort(v + 1, v + n + 1, cmp1);
33         sort(q + 1, q + cnt + 1, cmp2);
34         int now = 1;
35         for (int i = 1; i <= cnt; i ++ ) {
36             while (v[now].x <= q[i].x && now <= n) {
37                 update(v[now].y, 1); now ++ ;
38             }
39             ans[q[i].id] += q[i].type * (query(q[i].y));
40         }
41         for (auto &x : ans) x = max(0ll, x);
42     }
43 }t1, t2, t3, t4, t5, xx, yy;
44 int tr[maxn << 1];
45 il void add (int x, int k) {
46     for (; x <= n; x += lowbit(x)) tr[x] += k;
47 }
48 il int query (int x) {
49     int res = 0;
50     for (; x; x -= lowbit(x)) res += tr[x];
51     return res;
52 }
53
54 void solve() {
55     scanf("%lld", &n);
56     vector<int> p(n + 1), A(n + 1), B(n + 1);
57     int sum = 0;
58     for (int i = 1; i <= n; i ++ ) scanf("%lld", &p[i]);
59     for (int i = 1; i <= n; i ++ ) {
60         A[i] = query(p[i]); B[i] = p[i] - 1 - A[i];
61         add(p[i], 1);
62         sum += min(A[i], B[i]);
63         xx.add(i, p[i]); yy.add(i, p[i]);
64     }
65     // de(sum);
66     for (int i = 1; i <= n; i ++ ) {
67         if (A[i] == B[i]) t1.add(i, p[i]);
68         if (A[i] + 1 == B[i]) t2.add(i, p[i]);
69         if (A[i] == B[i] + 1) t3.add(i, p[i]);
70         if (A[i] + 2 <= B[i]) t4.add(i, p[i]);

```

```

71         if (A[i] >= B[i] + 2) t5.add(i, p[i]);
72     }
73     scanf("%lld", &m);
74     vector<int> x(m + 1), y(m + 1);
75     vector<int> ans1(m + 1), ans2(m + 1), ans3(m + 1), ans4(m + 1), ans5(m + 1);
76     vector<int> ansxx(m + 1), ansyy(m + 1);
77     for (int i = 1; i <= m; i++) {
78         scanf("%lld%lld", &x[i], &y[i]);
79         if (x[i] > y[i]) swap(x[i], y[i]);
80         int l = x[i] + 1, r = y[i] - 1;
81         int mn = min(p[x[i]], p[y[i]]), mx = max(p[x[i]], p[y[i]]);
82         t1.que(l, mn, r, mx, i);
83         t2.que(l, mn, r, mx, i);
84         t3.que(l, mn, r, mx, i);
85         t4.que(l, mn, r, mx, i);
86         t5.que(l, mn, r, mx, i);
87         xx.que(x[i], 1, y[i], p[x[i]] - 1, i);
88         yy.que(x[i], 1, y[i], p[y[i]] - 1, i);
89     }
90     t1.get(ans1); t2.get(ans2); t3.get(ans3); t4.get(ans4); t5.get(ans5);
91     xx.get(ansxx); yy.get(ansyy);
92     vector<int> ans(m + 1);
93     for (int i = 1; i <= m; i++) {
94         if (p[x[i]] < p[y[i]]) {
95             ans[i] += sum - ans1[i] - ans2[i] - ans4[i] + ans5[i];
96         }
97         else ans[i] += sum - ans1[i] - ans3[i] + ans4[i] - ans5[i];
98         ans[i] -= min(A[x[i]], B[x[i]]); ans[i] -= min(A[y[i]], B[y[i]]);
99         ans[i] += min(A[x[i]] + ansxx[i], B[x[i]] - ansxx[i]);
100        ans[i] += min(A[y[i]] - ansyy[i], B[y[i]] + ansyy[i]);
101    }
102    for (int i = 1; i <= m; i++) printf("%lld\n", ans[i]);
103 }
104

```

## 1.14 珂朵莉树

```

1 //https://codeforces.com/contest/896/problem/C
2 struct Node {
3     int l, r;
4     mutable ll val;
5     bool operator < (const Node &it) const { return l < it.l; }
6     Node (int L, int R = -1, ll v = 0) : l(L), r(R), val(v) {}
7 };
8 struct Chtholly {
9     #define IT set<Node>::iterator
10    set<Node> s;
11    IT split (int pos) {
12        IT it = s.lower_bound(Node(pos));
13        if (it != s.end() && it->l == pos) return it;
14        it--;
15        int l = it->l, r = it->r;
16        ll val = it->val;
17        s.erase(it); s.insert(Node(l, pos - 1, val));
18        return s.insert(Node(pos, r, val)).fi;
19    }
20    void assign (int l, int r, int val) {
21        IT it2 = split(r + 1), it1 = split(l);
22        s.erase(it1, it2); s.insert(Node(l, r, val));
23    }
24    void add (int l, int r, int val) {

```

```

25         IT it2 = split(r + 1), it1 = split(1);
26         for (IT it = it1; it != it2; it ++ ) {
27             it->val += val;
28         }
29     }
30     ll kth (int l, int r, int k) {
31         IT it2 = split(r + 1), it1 = split(1);
32         vector<pair<ll, int>> res; res.clear();
33         for (IT it = it1; it != it2; it ++ ) {
34             res.pb(pair<ll, int>(it->val, it->r - it->l + 1));
35         }
36         sort(all(res));
37         for (auto [v, cnt] : res) {
38             k -= cnt;
39             if (k <= 0) return v;
40         }
41     }
42     ll qmi (int a, int x, ll y) {
43         ll b = 1; a %= y;
44         while (x) {
45             if (x & 1) b = (b * a) % y;
46             a = a * a % y;
47             x >>= 1;
48         }
49         return b;
50     }
51     ll query (int l, int r, int x, ll y) {
52         IT it2 = split(r + 1), it1 = split(1);
53         ll res = 0;
54         for (IT it = it1; it != it2; it ++ ) {
55             res = (res + (it->r - it->l + 1) * qmi(it->val, x, y) % y) % y;
56         }
57         return res;
58     }
59 };
60

```

## 1.15 李超树

```

1  constexpr int N = 2e5 + 10;
2  class LC_SegT {
3      struct Func {
4          ll k, b;
5          Func(ll k = 0, ll b = 0) : k(k), b(b) {}
6          ll operator()(const ll x) const {
7              return k * x + b;
8          }
9      } tr[N << 2];
10 #define lc (u << 1)
11 #define rc (u << 1 | 1)
12 void insert(int u, int L, int R, int l, int r, Func k) {
13     int mid = L + R >> 1;
14     if (L >= l && R <= r) {
15         Func &f = tr[u];
16         int res = (k(L) > f(L)) + (k(R) > f(R));
17         if (res == 2) {
18             f = k;
19         } else if (res) {
20             if (k(mid) > f(mid)) {
21                 swap(f, k);

```

```

22         }
23         if (k(L) > f(L)) {
24             insert(lc, L, mid, l, r, k);
25         } else {
26             insert(rc, mid + 1, R, l, r, k);
27         }
28     }
29 } else {
30     if (l <= mid) {
31         insert(lc, L, mid, l, r, k);
32     }
33     if (r > mid) {
34         insert(rc, mid + 1, R, l, r, k);
35     }
36 }
37 }
38 #undef lc
39 #undef rc
40 public:
41     int n;
42     void init(const int Lim) {
43         for (n = 1; n < Lim; n <= 1) ;
44     }
45     void insert(Func k) {
46         insert(1, 1, n, 1, n, k);
47     }
48     ll query(int x) {
49         ll ret = 0;
50         for (int p = x + n - 1; p; p >>= 1) {
51             ret = max(ret, tr[p](x));
52         }
53         return ret;
54     }
55 }T[4];

```

## 2. 图论

### 2.1 最短路

#### 2.1.1 朴素Dijkstra $O(n^2)$

```

1  int g[N][N]; // 邻接矩阵存稠密图
2  int dist[N]; // 每个点距离点1的距离
3  bool st[N]; // 判断该点是否确定
4
5  int dijkstra()
6  {
7      memset(dist, 0x3f, sizeof dist);
8
9      dist[1] = 0;
10
11     for (int i = 0; i < n; i ++ ) { // n个点循环n次
12
13         int t = -1;
14
15         for (int j = 1; j <= n; j ++ ) { // 找未确定的点中dist最小的点
16             if (!st[j] && (t == -1 || dist[j] < dist[t]))
17                 t = j;
18         }

```

```

19
20     if (t == n && dist[t] != 0x3f3f3f3f) return dist[t];
21     else if (t == n && dist[t] == 0x3f3f3f3f) return -1; // 提前结束循环的优化
22
23     st[t] = true;
24
25     for (int j = 1; j <= n; j ++ )// 用t更新其他未确定点的距离
26         dist[j] = min(dist[j], dist[t] + g[t][j]);
27 }
28
29 if (dist[n] == 0x3f3f3f3f) return -1;
30 else return dist[n];
31 }
32
33 int main()
34 {
35     cin >> n >> m;
36
37     memset(g, 0x3f, sizeof g);
38
39     while (m -- ){
40         int a, b, c;
41         cin >> a >> b >> c;
42         g[a][b] = min(g[a][b], c);
43     }
44
45     printf("%d\n", dijkstra());
46
47     return 0;
48 }

```

## 2.1.2 堆优化Dijkstra $O(m\log n)$

```

1  #include <bits/stdc++.h>
2  using namespace std;
3
4  typedef pair<int, int> PII;
5
6  const int N = 1e6 + 10;
7
8  int n, m;
9  int h[N], w[N], e[N], ne[N], idx;
10 int dist[N];
11 bool st[N];
12
13 int add(int a, int b, int c)
14 {
15     e[idx] = b, w[idx] = c, ne[idx] = h[a], h[a] = idx ++ ;
16 }
17
18 // 求1号点到n号点的最短距离，如果不存在，则返回-1
19 int dijkstra()
20 {
21     memset(dist, 0x3f, sizeof dist);
22     dist[1] = 0;
23     priority_queue<PII, vector<PII>, greater<PII>> heap;
24     heap.push({0, 1}); // first存储距离，second存储节点编号
25
26     while (heap.size()){
27         auto t = heap.top();

```



```

28     heap.pop();
29
30     int ver = t.second, distance = t.first;
31
32     if (st[ver]) continue; // 防止产生冗余
33     st[ver] = true;
34
35     for (int i = h[ver]; i != -1; i = ne[i]){
36         int j = e[i];
37         if (dist[j] > dist[ver] + w[i]){
38             dist[j] = dist[ver] + w[i];
39             heap.push({dist[j], j});
40         }
41     }
42
43 }
44
45 if (dist[n] == 0x3f3f3f3f) return -1;
46 else return dist[n];
47 }
48
49 int main()
50 {
51     cin >> n >> m;
52     memset(h, -1, sizeof h);
53     while (m -- ){
54         int a, b, c;
55         cin >> a >> b >> c;
56         add(a, b, c);
57     }
58
59     cout << dijkstra() << endl;
60
61     return 0;
62 }

```

## 2.1.3 Bellman-ford $O(nm)$

```

1 // 有边数限制的最短路
2
3 struct Edge
4 {
5     int a, b, c;
6 }edges[M]; // 结构体存边
7
8 int n, m, k;
9 int dist[N];
10 int backup[N]; // 题目有特殊的边数限制，因此在更新时只能更新上次备份，否则会出现串联
11
12 void bellman_ford()
13 {
14     memset(dist, 0x3f, sizeof dist);
15
16     dist[1] = 0;
17     for (int i = 0; i < k; i ++ ){
18         memcpy(backup, dist, sizeof dist); // 每次都得上次的dist存到备份里
19         for (int j = 0; j < m; j ++ ){
20             auto e = edges[j];
21             dist[e.b] = min(dist[e.b], backup[e.a] + e.c);
22         }
23     }
24 }

```

```

23     }
24 }
25
26 int main()
27 {
28     cin >> n >> m >> k;
29
30     for (int i = 0; i < m; i++){
31         int a, b, c;
32         cin >> a >> b >> c;
33         edges[i] = {a, b, c};
34     }
35
36     bellman_ford();
37
38     if (dist[n] > 0x3f3f3f3f / 2) printf("impossible\n");
39     else printf("%d\n", dist[n]);
40
41     return 0;
42 }
43
44
45
46 // 无边数限制的最短路
47 #include <bits/stdc++.h>
48 using namespace std;
49
50 const int M = 1000010;
51
52 struct Edge
53 {
54     int a, b, c;
55 }edges[M]; // 结构体存边
56
57 int n, m, k;
58 int dist[M];
59
60
61 int bellman_ford()
62 {
63     memset(dist, 0x3f, sizeof dist);
64
65     dist[1] = 0;
66     //// 如果第n次迭代仍然会松弛三角不等式，就说明存在一条长度是n+1的最短路径，由抽屉原理，路径中至少存在两个相同的点，说明图中存在负权回路。
67     for (int i = 0; i < n; i++){
68         for (int j = 0; j < m; j++){
69             auto e = edges[j];
70             dist[e.b] = min(dist[e.b], dist[e.a] + e.c);
71         }
72     }
73
74     if (dist[n] == 0x3f3f3f3f) return -1;
75     return dist[n];
76 }
77
78 int main()
79 {
80     cin >> n >> m;
81
82     for (int i = 0; i < m; i++){
83         int a, b, c;

```

```

84     cin >> a >> b >> c;
85     edges[i] = {a, b, c};
86 }
87
88 if (bellman_ford() == -1) printf("impossible\n");
89 else printf("%d\n", dist[n]);
90
91 return 0;
92 }

```

## 2.1.4 SPFA O(nm)

```

1  // 最短路
2  #include <bits/stdc++.h>
3  using namespace std;
4
5  const int N = 100010;
6
7  int n, m;
8  int h[N], w[N], e[N], ne[N], idx;
9  int dist[N];
10 bool st[N];
11
12 void add(int a, int b, int c)
13 {
14     e[idx] = b, w[idx] = c, ne[idx] = h[a], h[a] = idx ++ ;
15 }
16
17 int spfa()
18 {
19     memset(dist, 0x3f, sizeof dist);
20     dist[1] = 0;
21
22     queue<int> q;
23     q.push(1);
24     st[1] = true;
25
26     while (q.size()){
27         int t = q.front();
28         q.pop();
29
30         st[t] = false;
31
32         for (int i = h[t]; i != -1; i = ne[i]){
33             int j = e[i];
34             if (dist[j] > dist[t] + w[i]){
35                 dist[j] = dist[t] + w[i];
36                 if (!st[j]){
37                     q.push(j);
38                     st[j] = true;
39                 }
40             }
41         }
42     }
43     printf("%d\n", dist[n]);
44     return dist[n];
45 }
46
47 int main()
48 {
49     cin >> n >> m;

```

```

50     memset(h, -1, sizeof h);
51
52     while (m -- ){
53         int a, b, c;
54         cin >> a >> b >> c;
55         add(a, b, c);
56     }
57
58     if (spfa() == 0x3f3f3f3f) printf("impossible");
59     else printf("%d\n", spfa());
60
61     return 0;
62 }
63
64
65 // 判断负权环
66 #include <bits/stdc++.h>
67 using namespace std;
68
69 const int N = 2010, M = 10010;
70
71 int n, m;
72 int h[N], w[M], e[M], ne[M], idx;
73 int dist[N], cnt[N];
74 bool st[N];
75
76 void add(int a, int b, int c)
77 {
78     e[idx] = b, w[idx] = c, ne[idx] = h[a], h[a] = idx ++ ;
79 }
80 // 优化: queue改stack, 或入队总数大于2*n即有负环
81 bool spfa()
82 {
83     queue<int> q;
84
85     for (int i = 1; i <= n; i ++ ){
86         st[i] = true;
87         q.push(i);
88     }
89
90     while (q.size()){
91         int t = q.front();
92         q.pop();
93
94         st[t] = false;
95
96         for (int i = h[t]; i != -1; i = ne[i]){
97             int j = e[i];
98             if (dist[j] > dist[t] + w[i]){
99                 dist[j] = dist[t] + w[i];
100                 cnt[j] = cnt[t] + 1;
101
102                 if (cnt[j] >= n) return true;
103
104                 if (!st[j]){
105                     q.push(j);
106                     st[j] = true;
107                 }
108             }
109         }
110     }
111     return false;

```

```

112 }
113
114 int main()
115 {
116     cin >> n >> m;
117     memset (h, -1, sizeof h);
118
119     while (m -- ){
120         int a, b, c;
121         cin >> a >> b >> c;
122         add(a, b, c);
123     }
124
125     if (spfa()) cout << "Yes" << endl;
126     else cout << "No" << endl;
127
128     return 0;
129 }
130
131 // SLF优化
132 int dist[maxn];
133 int cnt[maxn];
134 bool st[maxn];
135 vector<PII> G[maxn];
136
137 bool spfa(int s)
138 {
139     deque<int> q;
140     dist[s] = 0;
141     q.push_back(s);
142     cnt[s] ++ ;
143     st[s] = 1;
144
145     while (q.size()) {
146         int t = q.front();
147         q.pop_front();
148         st[t] = 0;
149
150         for (auto it : G[t]) {
151             int j = it.first, w = it.second;
152             if (dist[j] > dist[t] + w) {
153                 dist[j] = dist[t] + w;
154                 if (!st[j]) {
155                     if (!q.empty() && dist[j] > dist[q.front()]) {
156                         q.push_back(j);
157                     }
158                     else q.push_front(j);
159                     cnt[j] ++ ;
160                     if (cnt[j] > n) return true;
161                     st[j] = 1;
162                 }
163             }
164         }
165     }
166     return false;
167 }

```

## 2.1.5 Floyd $O(n^3)$

```
1  #include <bits/stdc++.h>
2  using namespace std;
3
4  const int N = 210, INF = 1e9;
5
6  int n, m, Q;
7  int d[N][N]; // i到j距离
8
9  void floyd()
10 {
11     for (int k = 1; k <= n; k ++ ){
12         for (int j = 1; j <= n; j ++ ){
13             for (int i = 1; i <= n; i ++ ){
14                 d[i][j] = min(d[i][j], d[i][k] + d[k][j]);
15             }
16         }
17     }
18 }
19
20 int main()
21 {
22     cin >> n >> m >> Q;
23
24     for (int i = 1; i <= n; i ++ ){
25         for (int j = 1; j <= n; j ++ ){
26             if (i == j) d[i][j] = 0;
27             else d[i][j] = INF;
28         }
29     }
30
31     while (m -- ){
32         int a, b, c;
33         cin >> a >> b >> c;
34         d[a][b] = min(d[a][b], c);
35     }
36
37     floyd();
38
39     while (Q -- ){
40         int a, b;
41         cin >> a >> b;
42         int t = d[a][b];
43         if (t > INF / 2) printf("impossible");
44         else printf("%d\n", t);
45     }
46
47     return 0;
48 }
49
50 // 传递闭包
51 for (int k = 0; k < n; k ++ ) {
52     for (int i = 0; i < n; i ++ ) {
53         for (int j = 0; j < n; j ++ ) {
54             d[i][j] |= d[i][k] && d[k][j];
55         }
56     }
57 }
58
59
60 // 求最小环
```

```

61 int n, m;
62 int d[maxn][maxn], g[maxn][maxn];
63 int pos[maxn][maxn];
64 int path[maxn], cnt;
65
66 void get_path (int i, int j) {
67     if (pos[i][j] == 0) return;
68
69     int k = pos[i][j];
70     get_path(i, k);
71     path[cnt++] = k;
72     get_path(k, j);
73 }
74
75 int main () {
76     cin >> n >> m;
77
78     memset(g, 0x3f, sizeof g);
79     for (int i = 1; i <= n; i++) g[i][i] = 0;
80
81     while (m--) {
82         int a, b, c;
83         cin >> a >> b >> c;
84         g[a][b] = g[b][a] = min(g[a][b], c);
85     }
86
87     int res = inf;
88     memcpy(d, g, sizeof d);
89     for (int k = 1; k <= n; k++) {
90         for (int i = 1; i < k; i++) {
91             for (int j = i + 1; j < k; j++) {
92                 if ((long long)d[i][j] + g[j][k] + g[k][i] < res) {
93                     res = d[i][j] + g[j][k] + g[k][i];
94                     cnt = 0;
95                     path[cnt++] = k;
96                     path[cnt++] = i;
97                     get_path(i, j);
98                     path[cnt++] = j;
99                 }
100             }
101         }
102         for (int i = 1; i <= n; i++) {
103             for (int j = 1; j <= n; j++) {
104                 if (d[i][j] > d[i][k] + d[k][j]) {
105                     d[i][j] = d[i][k] + d[k][j];
106                     pos[i][j] = k;
107                 }
108             }
109         }
110     }
111
112     if (res == inf) puts("No solution.");
113     else {
114         for (int i = 0; i < cnt; i++) cout << path[i] << ' ';
115         cout << endl;
116     }
117 }
118
119

```

## 2.1.6 路径还原

```
1 //以朴素dijkstra为例，记录一个path数组，当dist数组被更新时，就同步跟新path数组
2
3 #include <bits/stdc++.h>
4 using namespace std;
5
6 const int maxn = 510;
7 const int inf = 0x3f3f3f3f;
8 int g[maxn][maxn];
9 int st[maxn];
10 int dist[500010];
11 int path[500010]; // 记录走来的路径
12 int n, m;
13
14 int dijkstra()
15 {
16     memset(dist, 0x3f, sizeof dist);
17     memset(path, -1, sizeof path);
18
19     dist[1] = 0;
20
21     for (int i = 1; i <= n; i ++ ){
22         int t = -1;
23         for (int j = 1; j <= n; j ++ ){
24             if (!st[j] && (t == -1 || dist[j] < dist[t])) t = j;
25         }
26
27         st[t] = 1;
28
29         for (int j = 1; j <= n; j ++ ){
30             if (dist[j] > dist[t] + g[t][j]){
31                 dist[j] = dist[t] + g[t][j];
32                 path[j] = t; // 记录
33             }
34         }
35     }
36     return dist[n];
37 }
38
39 vector<int> get_path(int x){
40     vector<int> p;
41     for (; x != -1; x = path[x]) p.push_back(x);
42     reverse(p.begin(), p.end()); //p中存下的是n到1的顺序，我们逆反一下顺序。
43     return p;
44 }
45
46 int main()
47 {
48     cin >> n >> m;
49     for (int i = 1; i <= n; i ++ ){
50         for (int j = 1; j <= n; j ++ ){
51             g[i][j] = (i == j) ? 0 : inf;
52         }
53     }
54
55     for (int i = 1; i <= m; i ++ ){
56         int a, b, c;
57         cin >> a >> b >> c;
58         g[a][b] = min(g[a][b], c);
59     }
60 }
```



```

61     printf("%d\n", dijkstra());
62     vector<int> p = get_path(n);
63     for (auto it : p){
64         printf("%d ", it);
65     }
66     return 0;
67 }

```

## 2.1.7 最短路计数

```

1  int cnt[maxn]; // 长度为i的路径的数量。
2
3
4  while (q.size()) { // 以bfs为例
5      auto t = q.front();
6      q.pop();
7
8      for (auto it : G[t]) {
9          if (dist[it] > dist[t] + 1) {
10             dist[it] = dist[t] + 1;
11             cnt[it] = cnt[t];
12             q.push(it);
13         }
14         else if (dist[it] == dist[t] + 1) cnt[it] = (cnt[it] + cnt[t]) %
mod;
15     }
16 }

```

## 2.2 最小生成树

### 2.2.1 Prim

```

1  #include <bits/stdc++.h>
2  using namespace std;
3
4  const int N = 510, INF = 0x3f3f3f3f;
5
6  int n, m;
7  int g[N][N]; // 邻接矩阵存图
8  int dist[N]; // 1~i的距离
9  bool st[N];
10
11 int prim()
12 {
13     memset(dist, 0x3f, sizeof dist);
14
15     int res = 0;
16     for (int i = 0; i < n; i ++ ){
17         int t = -1;
18
19         for (int j = 1; j <= n; j ++ ){
20             if (!st[j] && (t == -1 || dist[t] > dist[j]))
21                 t = j;
22         }
23
24         if (i && dist[t] == INF) return INF;
25
26         if (i) res += dist[t];

```

```

27     st[t] = true;
28
29     for (int j = 1; j <= n; j ++ ) dist[j] = min(dist[j], g[t][j]);
30 }
31
32 return res;
33 }
34
35 int main()
36 {
37     cin >> n >> m;
38
39     memset(g, 0x3f, sizeof g);
40
41     while (m -- ){
42         int a, b, c;
43         cin >> a >> b >> c;
44         g[a][b] = g[b][a] = min(g[a][b], c); // 无向图
45     }
46
47     int t = prim();
48
49     if (t == INF) printf("impossible");
50     else printf("%d\n", t);
51
52     return 0;
53 }

```

## 2.2.2 Kruskal

```

1  #include <bits/stdc++.h>
2  using namespace std;
3
4  const int N = 100010, M = 200010, INF = 0x3f3f3f3f;
5
6  int n, m;
7  int p[N]; // 并查集
8
9  struct Edge
10 {
11     int a, b, w;
12 }edges[M];
13
14 bool cmp(Edge a, Edge b)
15 {
16     return a.w < b.w;
17 }
18
19 int find (int x)
20 {
21     if (p[x] != x) p[x] = find(p[x]);
22     return p[x];
23 }
24
25 int kruskal()
26 {
27     sort(edges, edges + m, cmp);
28
29     for (int i = 0; i <= n; i ++ ) p[i] = i; // 并查集的初始操作
30

```

```

31     int res = 0, cnt = 0; // cnt表示连通的边数
32     for (int i = 0; i < m; i ++ ){
33         int a = edges[i].a, b = edges[i].b, w = edges[i].w;
34
35         a = find(a), b = find(b);
36         if (a != b){
37             p[a] = b; // 将边加入集合
38             res += w;
39             cnt ++ ;
40         }
41     }
42     if (cnt < n - 1) return INF;
43     return res;
44 }
45
46 int main()
47 {
48     cin >> n>> m;
49
50     for (int i = 0; i < m; i ++ ){
51         int a, b, w;
52         cin >> a >> b >> w;
53         edges[i] = {a, b, w};
54     }
55
56     int t = kruskal();
57
58     if (t == INF) printf("impossible"); // 不连通
59     else printf("%d", t);
60
61     return 0;
62 }
63
64 // 次小生成树
65 #include <bits/stdc++.h>
66 using namespace std;
67
68 #define int long long
69 const int maxn = 510, maxm = 10010;
70 typedef pair<int, int> PII;
71 int n, m;
72 struct node {
73     int a, b, w;
74     bool f;
75     bool operator < (const node &it) const {
76         return w < it.w;
77     }
78 }e[maxn];
79 int p[maxn];
80 int d1[maxn][maxn], d2[maxn][maxn];
81 vector<PII> G[maxn];
82
83 int find (int x) {
84     if (p[x] != x) p[x] = find(p[x]);
85     return p[x];
86 }
87
88 void dfs (int u, int fa, int maxd1, int maxd2, int d1[], int d2[]) {
89     d1[u] = maxd1, d2[u] = maxd2;
90     for (auto it : G[u]) {
91         int j = it.first, w = it.second;
92         if (j != fa) {

```

```

93         int td1 = maxd1, td2 = maxd2;
94         if (w > td1) td2 = td1, td1 = w;
95         else if (w < td1 && w > td2) td2 = w;
96         dfs(j, u, td1, td2, d1, d2);
97     }
98 }
99 }
100
101 signed main () {
102     cin >> n >> m;
103     for (int i = 1; i <= m; i++) {
104         int a, b, w;
105         cin >> a >> b >> w;
106         e[i] = {a, b, w};
107     }
108
109     sort(e + 1, e + m + 1);
110     for (int i = 1; i <= n; i++) p[i] = i;
111
112     int sum = 0;
113     for (int i = 1; i <= m; i++) {
114         int a = e[i].a, b = e[i].b, w = e[i].w;
115         int pa = find(a), pb = find(b);
116         if (pa != pb) {
117             p[pa] = pb;
118             sum += w;
119             G[a].push_back({b, w});
120             G[b].push_back({a, w});
121             e[i].f = 1;
122         }
123     }
124
125     for (int i = 1; i <= n; i++) {
126         dfs(i, -1, -1e9, -1e9, d1[i], d2[i]);
127     }
128
129     int res = 1e18;
130     for (int i = 1; i <= m; i++) {
131         if (!e[i].f) {
132             int a = e[i].a, b = e[i].b, w = e[i].w;
133             int t;
134             if (w > d1[a][b]) {
135                 t = sum + w - d1[a][b];
136             }
137             else if (w > d2[a][b]) {
138                 t = sum + w - d2[a][b];
139             }
140             res = min(res, t);
141         }
142     }
143     cout << res << endl;
144 }

```

## 2.3 二分图匹配

### 2.3.1 染色法判断二分图

```
1  /*
2  染色法的实现思路（DFS）：
3  1.用1, 2代表两个颜色，0代表未染色，任选一个点染成1或2
4
5  2.遍历所有点，每次将未染色的点进行dfs
6
7  3.若染色失败即break/return
8  */
9
10 #include <bits/stdc++.h>
11 using namespace std;
12
13 const int N = 100010, M = 200010;
14
15 int n, m;
16 int h[N], e[M], ne[M], idx;
17 int color[N];
18
19 void add(int a, int b)
20 {
21     e[idx] = b, ne[idx] = h[a], h[a] = idx ++;
22 }
23
24 bool dfs(int u, int c)
25 {
26     color[u] = c; // 染色
27     for (int i = h[u]; i != -1; i = ne[i]){
28         int j = e[i];
29         if (!color[j]){
30             if (!dfs(j, 3 - c)) return false; // 如果在dfs递归的过程中出现染色失败，则整个图都不
是二分图
31         }
32         else if (color[j] == c) return false; // 如果一条边的两 endpoint 同种颜色，则染色失败
33     }
34
35     return true; // 无染色错误则染色成功
36 }
37
38 int main()
39 {
40     cin >> n >> m;
41
42     memset(h, -1, sizeof h);
43
44     while (m -- ){
45         int a, b;
46         cin >> a >> b;
47         add(a, b), add(b, a);
48     }
49
50     bool flag = true;
51     for (int i = 1; i <= n; i ++ ){ // 遍历所有点，因为二分图不一定是连通图
52         if (!color[i]){
53             if (!dfs(i, 1)){
54                 flag = false;
55                 break;
56             }
57         }
58     }
59     if (flag) printf("Yes\n");
```

```

60     else printf("No");
61
62     return 0;
63 }

```

## 2.3.2 匈牙利算法判断最大匹配

```

1  #include <bits/stdc++.h>
2  using namespace std;
3
4  const int N = 510, M = 100010;
5
6  int n1, n2, m;
7  int h[N], e[M], ne[M], idx;
8  int match[N]; // match[a] = b, 表示点a目前匹配了b
9  bool st[N]; // st[a] = true, 表示点a目前已经有预定
10
11 void add(int a, int b)
12 {
13     e[idx] = b, ne[idx] = h[a], h[a] = idx ++ ;
14 }
15
16 bool find(int x) // 注意与并查集的find函数区别, 为x找一个匹配, 或x的匹配点被别人预定, x要重新找一个匹配
17 {
18     // 匹配成功返回true
19     for (int i = h[x]; i != -1; i = ne[i]){
20         int j = e[i];
21         if (!st[j]){ // 该点目前没有被匹配
22             st[j] = true; // 预定该点
23             if (match[j] == 0 || find(match[j])){ // j点没有匹配, 或与j匹配的的点可以更换匹配
24                 match[j] = x;
25                 return true;
26             }
27         }
28     }
29     return false;
30 }
31
32 int main()
33 {
34     scanf("%d%d%d", &n1, &n2, &m);
35
36     memset(h, -1, sizeof h);
37
38     while (m -- ){
39         int a, b;
40         cin >> a >> b;
41         add(a, b); // 从左集合找右集合, 只存一条边也可以
42     }
43
44     int res = 0;
45     for (int i = 0; i <= n1; i ++ ){ // 二分图不一定连通, 因此要为所有点尝试匹配
46         memset(st, false, sizeof st);
47         if (find(i)) res ++;
48     }
49     cout << res << endl;
50
51     return 0;
52 }

```

```
53
54 最小点覆盖 = 最大匹配
55 最大独立集 = n - 最大匹配
56 最小路径点覆盖 = n - 最大匹配
57
```

## 2.4 拓扑排序

```
1  const int N = 1e5 + 10;
2  int e[N], ne[N], h[N], idx;
3  int d[N]; // d 代表每个元素的入度
4  int top[N]; // top是拓扑排序的序列
5  int cnt = 1; // cnt代表top中有多少个元素
6  int n, m;
7  void add(int a, int b){
8      e[idx] = b;
9      ne[idx] = h[a];
10     h[a] = idx ++;
11 }
12 bool topsort(){
13     queue<int> q;
14     int t;
15     for(int i = 1; i <= n; ++i) // 将所有入度为0的点加入队列
16         if(d[i] == 0) q.push(i);
17     while(q.size()){
18         t = q.front(); // 每次取出队列的首部
19         top[cnt] = t; // 加入到 拓扑序列中
20         cnt ++; // 序列中的元素 ++
21         q.pop();
22         for(int i = h[t]; i != -1; i = ne[i]){
23             // 遍历 t 点的出边
24             int j = e[i];
25             d[j] --; // j 的入度 --
26             if(d[j] == 0) q.push(j); // 如果 j 入度为0, 加入队列当中
27         }
28     }
29     if(cnt < n) return 0;
30     else return 1;
31 }
32
33 int main(){
34     int a, b;
35     cin >> n >> m;
36     memset(h, -1, sizeof h);
37     while(m--){
38         cin >> a >> b;
39         add(a, b);
40         d[b] ++; // a -> b, b的入度++
41     }
42     if(topsort() == 0) cout << "-1"; // 序列不合法
43     else {
44         for(int i = 1; i <= n; ++i){
45             cout << top[i] << " ";
46         }
47     }
48     return 0;
49 }
```

## 2.5 倍增LCA

```

1 constexpr int MAXP = 30;
2 int pa[MAXP][N], dep[N];
3 void dfs (int u, int fa, int d) {
4     pa[0][u] = fa;
5     for (int i = 1; i < MAXP; i++) {
6         pa[i][u] = pa[i - 1][pa[i - 1][u]];
7     }
8     dep[u] = d;
9     for (auto v : g[u]) {
10         if (v == fa) {
11             continue;
12         }
13         dfs(v, u, d + 1);
14     }
15 }
16 int lca (int x, int y) {
17     if (dep[x] < dep[y]) swap(x, y);
18     for (int i = MAXP - 1; i >= 0; i--) {
19         if (dep[pa[i][x]] >= dep[y]) {
20             x = pa[i][x];
21         }
22     }
23     if (x == y) return x;
24     for (int i = MAXP - 1; i >= 0; i--) {
25         if (pa[i][x] != pa[i][y]) {
26             x = pa[i][x], y = pa[i][y];
27         }
28     }
29     return pa[0][x];
30 }
31 dfs(1, 0, 1);

```

## 2.6 Tarjan

### 2.6.1 SCC

```

1 // 求scc的时候不要连无向边
2 vector<int> scc[N];
3 int cnt, idx;
4 int dfn[N], ins[N], low[N], bel[N];
5 stack<int> stk;
6 void tarjan (int u) {
7     dfn[u] = low[u] = ++idx;
8     ins[u] = true;
9     stk.push(u);
10    for (auto j : g[u]) {
11        if (!dfn[j]) {
12            tarjan(j);
13            low[u] = min(low[u], low[j]);
14        } else {
15            if (ins[j]) low[u] = min(low[u], dfn[j]);
16        }
17    }
18    if (dfn[u] == low[u]) {
19        ++cnt;
20        while (true) {
21            int v = stk.top();
22            ins[v] = false;
23            bel[v] = cnt;
24            scc[cnt].push_back(v);
25            stk.pop();

```



```

26         if (v == u) break;
27     }
28 }
29 }

```

## 2.6.2 BCC\_V

```

1 //去掉一个点后，连通块数量增加，不一定联通
2 //luoguP3388
3 int dfn[maxn], low[maxn], idx, sz;
4 bool cut[maxn];
5 void dfs (int u, int fa, int rt) {
6     dfn[u] = low[u] = ++ idx;
7     int ch = 0;
8     for (auto v : G[u]) {
9         if (!dfn[v]) {
10             dfs(v, u, rt);
11             ch ++ ;
12             low[u] = min(low[u], low[v]);
13             if (low[v] >= dfn[u]) cut[u] = 1;
14         } else if (v != fa) {
15             low[u] = min(low[u], dfn[v]);
16         }
17     }
18     if (u == rt && ch <= 1) cut[u] = 0;
19     sz += cut[u];
20 }

```

## 2.6.3 BCC\_E

```

1 vector<pii> G[maxn];
2 int dfn[maxn], low[maxn], idx, sz;
3 vector<int> bridge;
4 // stack<int> stk;
5 // vector<int> bcc[maxn];
6 // int bel[maxn], cnt;
7 void dfs (int u, int id) {
8     dfn[u] = low[u] = ++ idx;
9     // stk.push(u);
10    for (auto [v, id2] : G[u]) {
11        if (!dfn[v]) {
12            dfs(v, id2);
13            low[u] = min(low[u], low[v]);
14            if (low[v] == dfn[v]) bridge.pb(id2);
15        } else if (id != id2) {
16            low[u] = min(low[u], dfn[v]);
17        }
18    }
19    // if (dfn[u] == low[u]) {
20    //     ++ cnt;
21    //     while (true) {
22    //         int v = stk.top();
23    //         bcc[cnt].push_back(v);
24    //         bel[v] = cnt;
25    //         stk.pop();
26    //         if (v == u) break;
27    //     }
28    // }
29 }

```

## 2.7 差分约束

```
1  /*
2  xa - xb <= c ==> add(a, b, c);
3  x >= c ==> x0 - x <= -c ==> add(0, x, -c);
4  输出 x - x0
5  跑最短路求得最大值 x <= c ==> x = c
6  如果要最小值, 考虑使得x' = -x, 求得x'的最大值, 即为x的最小值
7  则-xa - (-xb) <= c ==> xb - xa <= c ==> add(b, a, c); 方向相反
8  最后输出x0 - x;
9
10 xa == xb xa <= xb && xb <= xa ==> add(a, b, 0), add(b, a, 0);
11 xa < xb xa - xb <= -1 ==> add(a, b, -1);
12 xa <= xb xa - xb <= 0 ==> add(a, b, 0);
13 */
```

## 2.8 树哈希

```
1 //这棵树中最多能选出多少个互不同构的子树
2 vector<int> G[maxn];
3 mt19937_64 rnd(chrono::steady_clock::now().time_since_epoch().count());
4 ull h[maxn], bas = rnd();
5 ull H(ull x) {
6     return x * x * x * 19890535 + 19260817;
7 }
8 ull F(ull x) {
9     return H(x & ((1ll << 32) - 1)) + H(x >> 32);
10 }
11 void dfs (int u, int fa) {
12     h[u] = bas;
13     for (auto j : G[u]) {
14         if (j != fa) {
15             dfs(j, u);
16             h[u] += F(h[j]);
17         }
18     }
19 }
20 void solve() {
21     if (bas % 2 == 0) bas ++ ;
22     cin >> n;
23     for (int i = 1; i < n; i ++ ) {
24         int u, v; cin >> u >> v;
25         G[u].pb(v); G[v].pb(u);
26     }
27     dfs(1, -1);
28     sort(h + 1, h + n + 1);
29     int ans = unique(h + 1, h + n + 1) - h - 1;
30     cout << ans << endl;
31 }
```

## 2.9 虚树

```
1 //https://www.luogu.com.cn/problem/P2495
2 vector<pii> G[maxn];
3 vector<int> E[maxn];
4 const int MAXP = 30;
5 int pa[maxn][MAXP], dep[maxn];
6 int stk[maxn], tp;
```

```

7 vector<int> vec;
8 int minv[maxn], query[maxn], dfn[maxn], tot;
9 inline void dfs (int u, int fa, int d) {
10     pa[u][0] = fa;
11     for (int i = 1; i < MAXP; i++) pa[u][i] = pa[pa[u][i - 1]][i - 1];
12     dep[u] = d;
13     dfn[u] = ++ tot;
14     for (auto [j, w] : G[u]) {
15         if (j != fa) {
16             minv[j] = min(minv[u], w);
17             dfs(j, u, d + 1);
18         }
19     }
20 }
21 inline int lca (int x, int y) {
22     if (dep[x] < dep[y]) swap(x, y);
23     for (int i = MAXP - 1; i >= 0; i--) if (dep[pa[x][i]] >= dep[y]) x = pa[x][i];
24     if (x == y) return x;
25     for (int i = MAXP - 1; i >= 0; i--) if (pa[x][i] != pa[y][i]) x = pa[x][i], y =
    pa[y][i];
26     return pa[x][0];
27 }
28 inline int dp (int u) {
29     int sum = 0, now = 0;
30     for (auto j : E[u]) {
31         sum += dp(j);
32     }
33     if (query[u]) now = minv[u];
34     else now = min(minv[u], sum);
35     query[u] = false;
36     return now;
37 }
38 inline int build_VT() {
39     E[1].clear(); stk[tp = 1] = 1;
40     for (auto j : vec) {
41         E[j].clear();
42         int a = lca(j, stk[tp]);
43         if (a == stk[tp]) {
44             stk[++ tp] = j;
45             continue;
46         }
47         while (dep[stk[tp - 1]] > dep[a]) {
48             E[stk[tp - 1]].pb(stk[tp]);
49             tp--;
50         }
51         if (a == stk[tp - 1]) {
52             E[a].pb(stk[tp]); tp--;
53         } else {
54             E[a].clear(); E[a].pb(stk[tp]);
55             stk[tp] = a;
56         }
57         stk[++ tp] = j;
58     }
59     while (tp > 1) {
60         E[stk[tp - 1]].pb(stk[tp]);
61         tp--;
62     }
63     return dp(1);
64 }
65 void solve() {
66     minv[1] = 1e18;
67     cin >> n;

```

```

68     for (int i = 1; i < n; i ++ ) {
69         int u, v, w; cin >> u >> v >> w;
70         G[u].pb({v, w}); G[v].pb({u, w});
71     }
72     dfs(1, 0, 1);
73     cin >> m;
74     while (m -- ) {
75         int num; cin >> num;
76         vec.clear();
77         for (int i = 1; i <= num; i ++ ) {
78             int x; cin >> x; vec.pb(x);
79             query[x] = true;
80         }
81         sort(all(vec), [&] (int a, int b) {
82             return dfn[a] < dfn[b];
83         });
84         cout << build_VT() << endl;
85     }
86 }

```

## 2.10 2-SAT

```

1 //https://www.luogu.com.cn/problem/P4171
2 /*
3 存边时记得存下逆否命题的边, a -> ~b, b -> ~a都要连
4 ab至少选一: ~a -> b, ~b -> a
5 ab不能同时选-> a->~b, b->~a
6 ab都选: a -> b, ~b -> ~a
7 a不能选: a -> ~a
8 */
9 vector<int> G[maxn << 1];
10 int cnt, idx;
11 int dfn[maxn], ins[maxn], low[maxn], bel[maxn];
12 stack<int> stk;
13 inline void tarjan (int u) {
14     dfn[u] = low[u] = ++ idx;
15     ins[u] = true;
16     stk.push(u);
17     for (auto j : G[u]) {
18         if (!dfn[j]) {
19             tarjan(j);
20             low[u] = min(low[u], low[j]);
21         } else {
22             if (ins[j]) low[u] = min(low[u], dfn[j]);
23         }
24     }
25     if (dfn[u] == low[u]) {
26         ++ cnt;
27         while (true) {
28             int v = stk.top();
29             ins[v] = false;
30             bel[v] = cnt;
31             stk.pop();
32             if (v == u) break;
33         }
34     }
35 }
36 void solve() {
37     cin >> n >> m;
38     cnt = idx = 0;
39     for (int i = 0; i <= 2 * (n + 1); i ++ ) {

```

```

40     G[i].clear(); dfn[i] = 0;
41 }
42 for (int i = 1; i <= m; i++) {
43     string a, b; cin >> a >> b;
44     int u = (stoi(a.substr(1)) - 1) * 2 + (a[0] == 'h'), v = (stoi(b.substr(1)) - 1)
* 2 + (b[0] == 'h');
45     G[u ^ 1].pb(v); G[v ^ 1].pb(u);
46 }
47 for (int i = 0; i < 2 * n; i++) {
48     if (!dfn[i]) tarjan(i);
49 }
50 for (int i = 0; i < n; i++) {
51     if (bel[2 * i] == bel[2 * i + 1]) {
52         cout << "BAD" << endl; return;
53     }
54 }
55 cout << "GOOD" << endl;
56 }

```

```

1 //https://darkbzoj.cc/problem/2199
2
3 const int maxn = 2e3 + 10;
4 vector<int> G[maxn << 1];
5 int cnt, idx;
6 int dfn[maxn], ins[maxn], low[maxn], bel[maxn];
7 stack<int> stk;
8 int vis[maxn];
9 inline void tarjan (int u) {
10     dfn[u] = low[u] = ++ idx;
11     ins[u] = true;
12     stk.push(u);
13     for (auto j : G[u]) {
14         if (!dfn[j]) {
15             tarjan(j);
16             low[u] = min(low[u], low[j]);
17         } else {
18             if (ins[j]) low[u] = min(low[u], dfn[j]);
19         }
20     }
21     if (dfn[u] == low[u]) {
22         ++ cnt;
23         while (true) {
24             int v = stk.top();
25             ins[v] = false;
26             bel[v] = cnt;
27             stk.pop();
28             if (v == u) break;
29         }
30     }
31 }
32 void dfs (int u) {
33     vis[u] = 1;
34     for (auto v : G[u]) {
35         if (!vis[v]) dfs(v);
36     }
37 }
38 bool check (int s, int t) {
39     ms(vis, 0);
40     dfs(s);
41     return vis[t];
42 }
43 void solve() {

```

```

44     cin >> n >> m;
45     for (int i = 1; i <= m; i++) {
46         int u, v; string a, b;
47         cin >> u >> a >> v >> b;
48         u--, v--;
49         u = 2 * u + (a == "Y"), v = 2 * v + (b == "Y");
50         G[u ^ 1].pb(v); G[v ^ 1].pb(u);
51     }
52     for (int i = 0; i < 2 * n; i++) {
53         if (!dfn[i]) tarjan(i);
54     }
55     for (int i = 0; i < n; i++) {
56         if (bel[2 * i] == bel[2 * i + 1]) {
57             cout << "IMPOSSIBLE" << endl; return;
58         }
59     }
60     for (int i = 0; i < n; i++) {
61         if (check(i * 2, i * 2 + 1)) cout << "Y";
62         else if (check(i * 2 + 1, i * 2)) cout << "N";
63         else cout << "?";
64     }
65 }

```

## 2.11 基环树

```

1  for (int i = 1; i <= n; i++) {
2      if (vis[i])
3          continue;
4      int u = i;
5      //找环上一点u
6      while (!vis[u]) {
7          vis[u] = true;
8          u = p[u];
9      }
10     int v = u;
11     vector<int> cyc;
12     //通过从u开始遍历找到环上所有点
13     while (true) {
14         cyc.pb(v);
15         oncyc[v] = true;
16         v = p[v];
17         if (v == u) break;
18     }
19     maxd = 0;
20     //对环上每个子树dfs
21     for (auto v : cyc)
22         dfs(v);
23 }
24
25
26 // dfs找环上边, e1代表正向边, e1^1代表反向边。
27 for (int i = 0; i < m; i++) {
28     int u, v;
29     cin >> u >> v;
30     G[u].pb(mkp(v, 2 * i));
31     if (u != v) G[v].pb(mkp(u, 2 * i + 1));
32 }
33 function<void(int, int)> dfs = [&] (int u, int id) {
34     dfn[u] = ++idx;
35     for (auto [v, id2] : G[u]) {

```

```

36     if (!dfn[v]) {
37         par[v] = u;
38         pe[v] = id2;
39         dfs(v, id2);
40     } else if (id2 != (id ^ 1) && dfn[v] <= dfn[u]) {
41         int w = u;
42         while (w != v) {
43             cyc.pb(pe[w]);
44             w = par[w];
45         }
46         cyc.pb(id);
47     }
48 }
49 };

```

```

1  /*https://www.luogu.com.cn/problem/P2607
2  基环树最大独立集*/
3  const int maxn = 1e6 + 10;
4  vector<int> G[maxn];
5  void solve() {
6      cin >> n;
7      vector<int> val(n + 1), p(n + 1);
8      for (int i = 1; i <= n; i++) {
9          cin >> val[i] >> p[i];
10         G[p[i]].pb(i);
11     }
12     vector<int> vis(n + 1);
13     vector<vector<int>> dp(2, vector<int>(n + 1)), dp2(2, vector<int>(n + 1));
14     vector<int> oncyc(n + 1);
15     function<void(int)> dfs = [&](int u) {
16         vis[u] = true;
17         dp[1][u] = val[u];
18         for (auto v : G[u]) {
19             if (oncyc[v]) continue;
20             dfs(v);
21             dp[0][u] += max(dp[0][v], dp[1][v]);
22             dp[1][u] += dp[0][v];
23         }
24     };
25     int ans = 0;
26     for (int i = 1; i <= n; i++) {
27         if (vis[i]) continue;
28         int u = i;
29         while (!vis[u]) {
30             vis[u] = true;
31             u = p[u];
32         }
33         int v = u;
34         vector<int> cyc;
35         while (true) {
36             cyc.pb(v);
37             oncyc[v] = true;
38             v = p[v];
39             if (u == v) break;
40         }
41         for (auto v : cyc)
42             dfs(v);
43         int cur = -1e18;
44         for (int t = 0; t < 2; t++) {
45             for (int j = 0; j < 2; j++) {
46                 if (t == j) dp2[j][0] = dp[j][cyc[0]];
47                 else dp2[j][0] = -1e18;

```

```

48     }
49     for (int i = 1; i < sz(cyc); i ++ ) {
50         int u = cyc[i];
51         dp2[0][i] = max(dp2[1][i - 1], dp2[0][i - 1]) + dp[0][u];
52         dp2[1][i] = dp2[0][i - 1] + dp[1][u];
53     }
54     if (!t) cur = max({cur, dp2[0][sz(cyc) - 1], dp2[1][sz(cyc) - 1]});
55     else cur = max(cur, dp2[0][sz(cyc) - 1]);
56 }
57 ans += cur;
58 }
59 cout << ans << endl;
60 }
61
62
63 /*https://www.luogu.com.cn/problem/P4381
64 基环树直径*/
65 const int maxn = 1e6 + 1e3;
66 vector<int> G[maxn];
67 void solve() {
68     cin >> n;
69     vector<int> L(n + 1), p(n + 1);
70     for (int i = 1; i <= n; i ++ ) {
71         cin >> p[i] >> L[i];
72         G[p[i]].pb(i);
73     }
74     int maxd = 0;
75     vector<int> vis(n + 1), dp(n + 1), oncyc(n + 1);
76     function<void(int)> dfs = [&] (int u) {
77         vis[u] = true;
78         dp[u] = 0;
79         for (auto v : G[u]) {
80             if (oncyc[v]) continue;
81             dfs(v);
82             int d = dp[v] + L[v];
83             maxd = max(maxd, d + dp[u]);
84             dp[u] = max(dp[u], d);
85         }
86     };
87     vector<int> s(n + 10), pre(n + 10), suf(n + 10);
88     int ans = 0;
89     for (int i = 1; i <= n; i ++ ) {
90         if (vis[i])
91             continue;
92         int u = i;
93         while (!vis[u]) {
94             vis[u] = true;
95             u = p[u];
96         }
97         int v = u;
98         vector<int> cyc;
99         while (true) {
100             cyc.pb(v);
101             oncyc[v] = true;
102             v = p[v];
103             if (v == u) break;
104         }
105         maxd = 0;
106         for (auto v : cyc)
107             dfs(v);
108         for (int i = 2; i <= sz(cyc); i ++ ) {
109             s[i] = s[i - 1] + L[cyc[i - 2]];

```



```

110     }
111     int tot1 = s[sz(cyc)] + L[cyc[sz(cyc) - 1]];
112     pre[0] = -1e18;
113     for (int i = 1; i <= sz(cyc); i++) {
114         pre[i] = max(pre[i - 1], dp[cyc[i - 1]] - s[i]);
115     }
116     suf[sz(cyc) + 1] = -1e18;
117     for (int i = sz(cyc); i >= 1; i--) {
118         suf[i] = max(suf[i + 1], dp[cyc[i - 1]] - s[i]);
119     }
120     for (int i = 1; i <= sz(cyc); i++) {
121         maxd = max(maxd, pre[i - 1] + s[i] + dp[cyc[i - 1]]);
122         maxd = max(maxd, suf[i + 1] + s[i] + dp[cyc[i - 1]] + tot1);
123     }
124     ans += maxd;
125 }
126 cout << ans << endl;
127 }

```

## 2.12 网络流

### 2.12.1 Dinic

$O(n^2 m)$

用之前一定要注意点数边数开够，开不够是会WA的，不会RE

```

1  constexpr int V = 400010;
2  constexpr int E = 4001000;
3  template<typename T>
4  struct FlowGraph {
5      int s, t, vtot;
6      int head[V], etot;
7      int dis[V], cur[V];
8      struct edge {
9          int v, nxt;
10         T f;
11     } e[E << 1];
12     void addedge (int u, int v, T f) {
13         e[etot] = { v, head[u], f }; head[u] = etot++;
14         e[etot] = { u, head[v], 0 }; head[v] = etot++;
15     }
16
17     bool bfs () {
18         for (int i = 1; i <= vtot; i++) {
19             dis[i] = 0;
20             cur[i] = head[i];
21         }
22         queue<int> q;
23         q.push (s); dis[s] = 1;
24         while (!q.empty ()) {
25             int u = q.front (); q.pop ();
26             for (int i = head[u]; ~i; i = e[i].nxt) {
27                 if (e[i].f && !dis[e[i].v]) {
28                     int v = e[i].v;
29                     dis[v] = dis[u] + 1;
30                     if (v == t) return true;
31                     q.push (v);
32                 }
33             }

```

```

34     }
35     return false;
36 }
37
38 T dfs (int u, T m) {
39     if (u == t) return m;
40     T flow = 0;
41     for (int i = cur[u]; ~i; cur[u] = i = e[i].nxt) {
42         if (e[i].f && dis[e[i].v] == dis[u] + 1) {
43             T f = dfs (e[i].v, min (m, e[i].f));
44             e[i].f -= f;
45             e[i ^ 1].f += f;
46             m -= f;
47             flow += f;
48             if (!m) break;
49         }
50     }
51     if (!flow) dis[u] = -1;
52     return flow;
53 }
54 T dinic () {
55     T flow = 0;
56     while (bfs ()) flow += dfs (s, numeric_limits<T>::max ());
57     return flow;
58 }
59 void init (int s_, int t_, int vtot_) {
60     s = s_;
61     t = t_;
62     vtot = vtot_;
63     etot = 0;
64     for (int i = 1; i <= vtot; i++) head[i] = -1;
65 }
66 };
67
68 FlowGraph<ll> g;
69 void Solve () {
70     int n, m, s, t;
71     cin >> n >> m >> s >> t;
72     g.init (s, t, n);
73     for (int i = 1; i <= m; i++) {
74         int u, v, f;
75         cin >> u >> v >> f;
76         g.addegde (u, v, f);
77     }
78     cout << g.dinic () << '\n';
79 }

```

## 2.12.2 MCMF

```

1  constexpr int V = 20100;
2  constexpr int E = 201000;
3  template<typename T>
4  struct MinCostGragh {
5      int s, t, vtot;
6      int head[V], etot;
7      T dis[V], flow, cost;
8      int pre[V];
9      bool vis[V];
10
11     struct edge {
12         int v, nxt;

```

```

13     T f, c;
14 }e[E * 2];
15 void addedge (int u, int v, T f, T c, T f2 = 0) {
16     e[etot] = {v, head[u], f, c}; head[u] = etot++;
17     e[etot] = {u, head[v], f2, -c}; head[v] = etot++;
18 }
19
20 bool spfa () {
21     T inf = numeric_limits<T>::max() / 2;
22     for (int i = 1; i <= vtot; i++) {
23         dis[i] = inf;
24         vis[i] = false;
25         pre[i] = -1;
26     }
27     dis[s] = 0;
28     vis[s] = true;
29     queue<int> q;
30     q.push(s);
31     while (!q.empty()) {
32         int u = q.front();
33         for (int i = head[u]; ~i; i = e[i].nxt) {
34             int v = e[i].v;
35             if (e[i].f && dis[v] > dis[u] + e[i].c) {
36                 dis[v] = dis[u] + e[i].c;
37                 pre[v] = i;
38                 if (!vis[v]) {
39                     vis[v] = 1;
40                     q.push(v);
41                 }
42             }
43         }
44         q.pop();
45         vis[u] = false;
46     }
47     return dis[t] != inf;
48 }
49
50 void augment() {
51     int u = t;
52     T f = numeric_limits<T>::max();
53     while (~pre[u]) {
54         f = min(f, e[pre[u]].f);
55         u = e[pre[u] ^ 1].v;
56     }
57     flow += f;
58     cost += f * dis[t];
59     u = t;
60     while (~pre[u]) {
61         e[pre[u]].f -= f;
62         e[pre[u] ^ 1].f += f;
63         u = e[pre[u] ^ 1].v;
64     }
65 }
66
67 pair<T, T> solve() {
68     flow = 0;
69     cost = 0;
70     while (spfa()) {
71         augment();
72     }
73     return {flow, cost};
74 }

```

```

75 void init(int s_, int t_, int vtot_) {
76     s = s_;
77     t = t_;
78     vtot = vtot_;
79     etot = 0;
80     for (int i = 1; i <= vtot; i++) head[i] = -1;
81 }
82 };
83
84 MinCostGraph<int> g;
85
86 void Solve() {
87     int n, m, s, t;
88     cin >> n >> m >> s >> t;
89     g.init(s, t, n);
90     for (int i = 1; i <= m; i++) {
91         int u, v, f, c;
92         cin >> u >> v >> f >> c;
93         g.addedge(u, v, f, c);
94     }
95     auto [flow, cost] = g.solve();
96     cout << flow << ' ' << cost << '\n';
97 }

```

## 2.13 三元环/四元环计数

```

1 //HDU 6184
2 void Solve() {
3     int n, m;
4     cin >> n >> m;
5     vector<int> deg(n + 1);
6     vector<vector<pair<int, int>>> adj(n + 1);
7     vector<pair<int, int>> eds;
8     for (int i = 1; i <= m; i++) {
9         int u, v;
10        cin >> u >> v;
11        ++deg[u], ++deg[v];
12        eds.emplace_back(u, v);
13    }
14
15    for (int i = 0; i < m; i++) {
16        auto [a, b] = eds[i];
17        if (deg[a] > deg[b] || deg[a] == deg[b] && a > b) adj[a].emplace_back(b, i + 1);
18        else adj[b].emplace_back(a, i + 1);
19    }
20
21    vector<pair<int, int>> vis(n + 1);
22    vector<int> cnt(m + 1);
23    i64 ans = 0, cur = 0;
24    for (int u = 1; u <= n; u++) {
25        ++cur;
26        for (auto [v, id] : adj[u]) {
27            vis[v] = make_pair(cur, id);
28        }
29        for (auto [v, id1] : adj[u]) {
30            for (auto [k, id2] : adj[v]) {
31                if (vis[k].first == cur) {
32                    cnt[vis[k].second]++;
33                    cnt[id1]++;
34                    cnt[id2]++;

```

```

35     }
36     }
37     }
38 }
39
40 for (int i = 1; i <= m; i++) {
41     ans += 1ll * cnt[i] * (cnt[i] - 1) / 2;
42 }
43 cout << ans << '\n';
44 }

```

```

1 //四元环计数 O((n + m)sqrt(m))
2 void Solve() {
3     int n, m;
4     cin >> n >> m;
5     vector<vector<int>> adj(n + 1);
6     for (int i = 1; i <= m; i++) {
7         int u, v;
8         cin >> u >> v;
9         adj[u].emplace_back(v);
10        adj[v].emplace_back(u);
11    }
12    vector<int> deg(n + 1), id(n + 1);
13    for (int i = 1; i <= n; i++) {
14        deg[id[i] = i] = adj[i].size();
15    }
16    sort(id.begin() + 1, id.end(), [&](auto &a, auto &b) {
17        return deg[a] < deg[b];
18    });
19    vector<int> rk(n + 1);
20    for (int i = 1; i <= n; i++) {
21        rk[id[i]] = i;
22    }
23
24    vector<vector<int>> g(n + 1);
25    for (int u = 1; u <= n; u++) {
26        for (auto v : adj[u]) {
27            if (rk[v] > rk[u]) g[u].emplace_back(v);
28        }
29    }
30
31    vector<int> cnt(n + 1);
32    i64 ans = 0;
33    for (int u = 1; u <= n; u++) {
34        for (auto v : adj[u]) {
35            for (auto w : g[v]) {
36                if (rk[w] > rk[u]) ans += cnt[w]++;
37            }
38        }
39        for (auto v : adj[u]) {
40            for (auto w : g[v]) {
41                if (rk[w] > rk[u]) cnt[w] = 0;
42            }
43        }
44    }
45    cout << ans << '\n';
46 }

```

## 2.14 点分治

```

1 //https://www.luogu.com.cn/problem/P3806
2 constexpr int N = 1e4 + 10;
3 int q[N], siz[N], mxsz[N], del[N];
4 vector<array<int, 2>> g[N];
5 int n, m;
6
7 int tot, rt;
8 void getp (int u, int fa) {
9     siz[u] = 1, mxsz[u] = 0;
10    for (auto [v, w] : g[u]) {
11        if (v != fa && !del[v]) {
12            getp(v, u);
13            mxsz[u] = max(mxsz[u], siz[v]);
14            siz[u] += siz[v];
15        }
16    }
17    mxsz[u] = max(mxsz[u], tot - siz[u]);
18    if (mxsz[u] < mxsz[rt]) {
19        rt = u;
20    }
21 }
22
23 int d[N], cnt;
24 int dist[N], has[10000010], ok[N];
25 void calcinfo (int u, int fa) {
26     d[++cnt] = dist[u];
27     for (auto [v, w] : g[u]) {
28         if (v != fa && !del[v]) {
29             dist[v] = dist[u] + w;
30             calcinfo(v, u);
31         }
32     }
33 }
34
35 queue<int> que;
36
37 void dfz (int u, int fa) {
38     has[0] = true;
39     que.push(0);
40     del[u] = true;
41
42     for (auto [v, w] : g[u]) {
43         if (v != fa && !del[v]) {
44             dist[v] = w;
45             calcinfo(v, u);
46             for (int j = 1; j <= cnt; j++) {
47                 for (int i = 1; i <= m; i++) {
48                     if (q[i] >= d[j]) {
49                         ok[i] |= has[q[i] - d[j]];
50                     }
51                 }
52             }
53             for (int j = 1; j <= cnt; j++) {
54                 if (d[j] < 10000010) {
55                     que.push(d[j]);
56                     has[d[j]] = true;
57                 }
58             }
59             cnt = 0;
60         }
61     }
62 }

```

```

63     while (que.size()) {
64         has[que.front()] = false;
65         que.pop();
66     }
67
68     for (auto [v, w] : g[u]) {
69         if (v != fa && !del[v]) {
70             tot = siz[v];
71             rt = 0;
72             mxsz[rt] = 1e9;
73             getp(v, u);
74             getp(rt, -1);
75             dfz(rt, u);
76         }
77     }
78 }
79
80 void Solve() {
81     cin >> n >> m;
82
83     for (int i = 1; i < n; i++) {
84         int u, v, w;
85         cin >> u >> v >> w;
86         g[u].push_back({v, w});
87         g[v].push_back({u, w});
88     }
89
90     for (int i = 1; i <= m; i++) {
91         cin >> q[i];
92     }
93
94     rt = 0;
95     mxsz[rt] = 1e9;
96     tot = n;
97     getp(1, -1);
98     getp(rt, -1);
99     dfz(rt, -1);
100
101     for (int i = 1; i <= m; i++) {
102         cout << (ok[i] ? "AYE" : "NAY") << '\n';
103     }
104 }

```

## 3 数学

### 3.1 线性筛

```

1  constexpr int N = 1e5 + 10;
2  int primes[N], cnt;
3  bool st[N];
4  void LinearSieve(int N) {
5      for (int i = 2; i <= N; i++) {
6          if (!st[i]) primes[cnt++] = i;
7          for (int j = 0; primes[j] <= N / i; j++) {
8              st[primes[j] * i] = true;
9              if (i % primes[j] == 0) break;
10         }
11     }
12 }

```

## 3.2 快速乘 & 快速幂

```
1 // 快速幂
2 int qmi (int a, int b) { // a ^ b
3     int ans = 1;
4     while (b) {
5         if (b & 1) ans = ans * a % mod; // 或 ans = mul(ans, a);
6         a = a * a % mod; // 或 ans = mul(a, a);
7         b >>= 1;
8     }
9     return ans;
10 }
11
12
13 // 快速乘
14 int mul (int a, int b) { // a * b
15     int ans = 0;
16     while (b) {
17         if (b & 1) ans = (ans + a) % mod;
18         b >>= 1;
19         a = (a * 2) % mod;
20     }
21     return ans;
22 }
23 //高精快速幂, mod过大时使用
24 typedef unsigned long long llong;
25 llong power (llong x, llong n, llong mod)
26 {
27     __int128 a = (__int128) x;
28     llong res = 1;
29     while (n > 0) {
30         if (n & 1) res = res * a % mod;
31         a = a * a % mod;
32         n >>= 1;
33     }
34     return res;
35 }
```

## 3.3 拓展欧几里得算法 exgcd

```
1 // 求x, y, 使得ax + by = n
2 int exgcd(int a, int b, int &x, int &y) {
3     if (!b) {
4         x = 1; y = 0;
5         return a;
6     }
7     int d = exgcd(b, a % b, y, x);
8     y -= (a / b) * x;
9     return d;
10 }
11 //调整
12 x *= n / g, y *= n / g;
13 int t = x / (b / g);
14 x -= b / g * t, y += a / g * t;
15 if (x < 0) x += b / g, y -= a / g;
```



```

1  int a, b, c;
2  cin >> a >> b >> c;
3  int x, y;
4  int g = exgcd(a, b, x, y);
5  if (c % g != 0) {
6      cout << -1 << '\n';
7      return;
8  }
9  x *= c / g, y *= c / g;
10 int tx = (-x) / (b / g);
11 if (x + tx * (b / g) <= 0) tx++;
12 int ty = y / (a / g);
13 if (y - ty * (a / g) <= 0) ty--;
14 //没有正整数解
15 if (tx > ty) {
16     // x的最小整数解, y的最小整数解
17     cout << x + tx * (b / g) << ' ' << y - ty * (a / g) << '\n';
18 } else {
19     //正整数解个数
20     cout << ty - tx + 1 << ' ';
21     //x的最小正整数解, y最小正整数解
22     cout << x + tx * (b / g) << ' ' << y - ty * (a / g) << ' ';
23     //x的最大正整数解, y的最大正整数解
24     cout << x + ty * (b / g) << ' ' << y - tx * (a / g) << '\n';
25 }

```

## 3.4 欧拉函数

```

1  // 求1~N中与N互质的个数
2  int phi(int x)
3  {
4      int res = x;
5      for (int i = 2; i <= x / i; i++)
6          if (x % i == 0)
7          {
8              res = res / i * (i - 1);
9              while (x % i == 0) x /= i;
10         }
11     if (x > 1) res = res / x * (x - 1);
12
13     return res;
14 }
15
16 // 筛法求欧拉函数
17 int primes[N], cnt;    // primes[]存储所有素数
18 int euler[N];          // 存储每个数的欧拉函数
19 bool st[N];            // st[x]存储x是否被筛掉
20
21
22 void get_eulers(int n)
23 {
24     euler[1] = 1;
25     for (int i = 2; i <= n; i++)
26     {
27         if (!st[i])
28         {
29             primes[cnt++] = i;
30             euler[i] = i - 1;
31         }
32         for (int j = 0; primes[j] <= n / i; j++)

```

```

33     {
34         int t = primes[j] * i;
35         st[t] = true;
36         if (i % primes[j] == 0)
37         {
38             euler[t] = euler[i] * primes[j];
39             break;
40         }
41         euler[t] = euler[i] * (primes[j] - 1);
42     }
43 }
44 }

```

### 3.5 约数个数/和 定理

```

1 // 约数个数
2 //n=p1^a1*p2^a2*p3^a3*...*pk^ak
3 //f(n) = (a1+1)(a2+1)(a3+1)...(ak+1)
4 int main()
5 {
6     int n;
7     cin >> n;
8
9     unordered_map<int, int> primes;
10
11     while (n -- )
12     {
13         int x;
14         cin >> x;
15
16         for (int i = 2; i <= x / i; i ++ )
17             while (x % i == 0)
18             {
19                 x /= i;
20                 primes[i] ++ ;
21             }
22
23         if (x > 1) primes[x] ++ ;
24     }
25
26     LL res = 1;
27     for (auto p : primes) res = res * (p.second + 1) % mod;
28
29     cout << res << endl;
30
31     return 0;
32 }
33
34 // 约数之和
35 //n=p1^a1*p2^a2*p3^a3*...*pk^ak
36 //f(n)=(p1^0+p1^1+p1^2+...p1^a1)(p2^0+p2^1+p2^2+...p2^a2)...(pk^0+pk^1+pk^2+...pk^ak)
37 unordered_map<int, int> mp;
38 while(t -- )
39 {
40     int x; scanf("%d", &x); // 即n
41     for(int i = 2; i <= x / i; i ++ )
42     {
43         while(x % i == 0)
44         {
45             x /= i;

```

```

46         mp[i]++;
47     }
48 }
49 if(x > 1) mp[x] ++;
50 }
51 long long res = 1;
52 for(auto p : mp)
53 {
54     long long a = p.first, b = p.second;
55     long long t = 1;
56     while(b -- )
57     {
58         t = (t * a + 1) % mod; // 秦九韶算法
59     }
60     res = res * t % mod;
61 }
62 cout << res << endl;

```

## 3.6 组合数

### 结论

1.有 $n$ 个**完全相同**的元素，将其分为 $k$ 组。

考虑 $k - 1$ 块板子插入 $n - 1$ 个空白。

$$\binom{n-1}{k-1} \quad (1)$$

2.有 $n$ 个**完全相同**的元素，将其分为 $k$ 组，允许每组为空。

考虑求一组新解 $x'_i = x_i - 1$ ，即 $\sum_{i=1}^k x'_i = n$ ，把1全部移至等式右边。

$$\binom{n+k-1}{k-1} \quad (2)$$

3.有 $n$ 个**完全相同**的元素，将其分为 $k$ 组，每组有下界 $a_i$ 。

考虑求一组新解 $x'_i = x_i - (a_i - 1)$ ，同上。

$$\binom{n+k-1-\sum a_i}{k-1} \quad (3)$$

(1)中的情况可看为下界是1的特殊情况。

4.1 ~  $n$ 的排列中选出 $k$ 个两两不相邻的方案。

$$\binom{n-k+1}{k} \quad (4)$$

5.多重集组合数， $S = \{n_1 \cdot a_1, n_2 \cdot a_2, n_3 \cdot a_3, \dots, n_k \cdot a_k\}$ ，则 $S$ 的全排列个数：

$$\frac{n!}{\prod_{i=1}^k n_i!} \quad (5)$$

6.多重集组合数， $S = \{n_1 \cdot a_1, n_2 \cdot a_2, n_3 \cdot a_3, \dots, n_k \cdot a_k\}$ ，从 $S$ 中任选出 $r$ 个元素的方案数，其中 $(r < n_i, \forall i \in [1, k])$

用隔板法解决，等价于 $x_1 + x_2 + \dots + x_k = r$ 的非负整数解的数目。

$$\binom{r+k-1}{k-1} \quad (6)$$

7.多重集组合数， $S = \{n_1 \cdot a_1, n_2 \cdot a_2, n_3 \cdot a_3, \dots, n_k \cdot a_k\}$ ，从 $S$ 中任选出 $r$ 个元素的方案数

答案有：

$$\left| \bigcap_{i=1}^k S_i \right| = |U| - \left| \bigcup_{i=1}^k \overline{S_i} \right| \quad (7)$$

根据容斥，得：

$$\begin{aligned} \left| \bigcup_{i=1}^k \overline{S_i} \right| &= \sum_i |\overline{S_i}| - \sum_{i,j} |\overline{S_i} \cap \overline{S_j}| + \sum_{i,j,k} |\overline{S_i} \cap \overline{S_j} \cap \overline{S_k}| - \dots \\ &\quad + (-1)^{k-1} \left| \bigcap_{i=1}^k \overline{S_i} \right| \\ &= \sum_i \binom{k+r-n_i-2}{k-1} - \sum_{i,j} \binom{k+r-n_i-n_j-3}{k-1} + \sum_{i,j,k} \binom{k+r-n_i-n_j-n_k-4}{k-1} - \dots \\ &\quad + (-1)^{k-1} \binom{k+r-\sum_{i=1}^k n_i-k-1}{k-1} \end{aligned} \quad (8)$$

最后可以转化成：

$$Ans = \sum_{p=0}^k (-1)^p \sum_A \binom{k+r-1-\sum_A n_{A_i}-p}{k-1} \quad (9)$$

## 范德蒙德卷积

$$\sum_{i=0}^k \binom{n}{i} \binom{m}{k-i} = \binom{n+m}{k} \quad (10)$$

推论1：

$$\sum_{i=1}^n \binom{n}{i} \binom{n}{i-1} = \binom{2n}{n+1} \quad (11)$$

推论2

$$\sum_{i=0}^n \binom{n}{i}^2 = \binom{2n}{n} \quad (12)$$

推论3

$$\sum_{i=0}^m \binom{n}{i} \binom{m}{i} = \binom{n+m}{m} \quad (13)$$

## 板子

```
1 vector<mint>fac(n + 1), ifac(n + 1);
2 fac[0] = 1;
3 for (int i = 1; i <= n; i++) {
4     fac[i] = fac[i - 1] * i;
5 }
6 ifac[n] = fac[n].inv();
7 for (int i = n; i >= 1; i--) {
8     ifac[i - 1] = ifac[i] * i;
9 }
10
11 auto C = [&] (int n, int m) {
12     if (n < m || n < 0 || m < 0) {
13         return mint(0);
14     }
15     return fac[n] * ifac[m] * ifac[n - m];
```

```
16 };
```

```
1 constexpr int N = 2e5 + 10;
2 mint fac[N], ifac[N];
3 void Init(int n) {
4     fac[0] = 1;
5     for (int i = 1; i <= n; i++) {
6         fac[i] = fac[i - 1] * i;
7     }
8     ifac[n] = fac[n].inv();
9     for (int i = n; i >= 1; i--) {
10        ifac[i - 1] = ifac[i] * i;
11    }
12 }
13 mint C (int n, int m) {
14     if (n < m || n < 0 || m < 0) {
15         return mint(0);
16     }
17     return fac[n] * ifac[m] * ifac[n - m];
18 }
```

```
1 const int N = 2010;
2 int C[N][N];
3
4 void Init() {
5     for (int i = 0; i < N; ++i) C[i][0] = 1;
6     for (int i = 0; i < N; ++i)
7         for (int j = 1; j <= i; ++j)
8             C[i][j] = (C[i - 1][j - 1] + C[i - 1][j]) % Mod;
9 }
```

```
1 // lucas定理
2 // C(a, b) = C(a % p, b % p) * C(a / p, b / p); p为质数
3 int a, b, p; // 求C(a, b) % p;
4 inline int qmi (int a, int b) { int res = 1; while (b) { if (b & 1) res = res * a % p; a = a * a % p; b >>= 1; } return res; }
5 inline int inv (int x) {return qmi(x, p - 2)};
6 int fac[maxn]; inline void getfac () {for (int i = 1; i <= maxn - 10; i ++ ) fac[i] = fac[i - 1] * i % p;}
7 inline int C (int n, int m) {
8     if (n < m) return 0;
9     if (n < p) return fac[n] * inv(fac[n - m]) % p * inv(fac[m]) % p;
10    return C(n / p, m / p) * C(n % p, m % p) % p;
11 }
12
13 // k次前缀和的组合数
14 C (k-1+i, k-1)
```

### 3.7 数论分块

$$\sum_{i=1}^m \left\lfloor \frac{n}{i} \right\rfloor \quad (14)$$

```
1 // 下取整
```

```

2 inline int getdown (int n) {
3     int ans = 0;
4     for(int l = 1, r, len; l <= m; l = r + 1) {
5         if (n / l == 0) break;
6         r = min(m, n / (n / l)), len = r - l + 1;
7         ans += len * (n / l);
8     }
9     return ans;
10 }
11
12 // 上取整
13 for (int l = 1, r; l <= n; l = r + 1) {
14     if (l == n) {
15         ans++;
16         break;
17     } else {
18         int v = (n + l - 1) / l;
19         r = (n + v - 2) / (v - 1) - 1;
20         ans += (r - l + 1) * v;
21     }
22 }

```

### 3.8 卡特兰数

$$f(n) = C_{2n}^n - C_{2n}^{n-1} \quad (15)$$

- 1 1.n 个元素进栈序列为: 1, 2, 3, 4, ..., n, 则有多少种出栈序列。
- 2 2.n 对括号, 则有多少种 “括号匹配” 的括号序列
- 3 3.n + 1 个叶子节点能够构成多少种形状不同的 (国际) 满二叉树
- 4 4.电影票一张 50 coin, 且售票厅没有 coin, m 个人各自持有 50 coin, n 个人各自持有 100 coin。则有多少种排队方式, 可以让每个人都买到电影票。
- 5 5.8 个高矮不同的人需要排成两队, 每队 4 个人。其中, 每排都是从低到高排列, 且第二排的第 i 个人比第一排中第 i 个人高, 则有多少种排队方式?
- 6 6.在一个凸多边形中, 通过若干条互不相交的对角线, 把这个多边形划分成了若干个三角形。任务是键盘上输入凸多边形的边数n, 求不同划分的方案数f (n)。
- 7
- 8 前几项:
- 9 1, 1, 2, 5, 14, 42, 132, 429, 1430, 4862, 16796, 58786, 208012, 742900, 2674440, 9694845, 35357670, 129644790, 477638700, 1767263190, 6564120420, 24466267020, 91482563640, 343059613650, 1289904147324, 4861946401452, ...

### 3.9 线性求逆

```

1 ll inv[maxn] = {0,1};
2 for (int i = 2; i < maxn; i++)
3     inv[i] = (mod - mod / i) * inv[mod % i] % mod;

```

### 3.10 博弈论

```

1 经典nim
2 n堆, 每次拿任意个, 不可不拿, 先拿完赢
3 // 先手是否必胜?
4 cin >> n;
5 int res = 0;
6 for (int i = 0; i < n; i++){
7     int a;
8     cin >> a;
9     res ^= a;
10 }
11

```

```

12  if (res) printf("Yes\n");
13  else printf("No\n");
14
15
16  anti-nim游戏
17  n堆，每次拿任意个，不可不拿，先拿完输
18  先手胜当且仅当 ①所有堆石子数都为1且游戏的SG值为0（即有偶数个孤单堆-每堆只有1个石子数）；②存在某堆石子数大
    于1且游戏的SG值不为0.
19  bool ok = 0;
20  int ans = 0;
21  for (int i = 0; i < sz(res); i ++ ) {
22      ans ^= res[i];
23      if (res[i] > 1) ok = 1;
24  }
25
26  if ((!ok && ans == 0) || (ok && ans != 0)) puts("Alice win");
27  else puts("Bob win");
28
29
30  对称博弈
31

```

### 树上博弈

设 $T$ 为一个森林，其中有 $n$ 颗有根树，且树根都在地面上。 $Alice$ 、 $Bob$ 每次选择某一棵树的一条边，删除这条边以及这条边所连接的地上部分，最后无法操作的人输掉博弈。

### 克朗原理

$$SG(x) = \begin{cases} 0 & t = 0 \\ (SG(x_1) + 1) \oplus (SG(x_2) + 1) \oplus \cdots \oplus (SG(x_n) + 1) & t > 0 \end{cases}$$

## 3.11 最小二乘法求线性回归方程

$$k = \frac{\sum_{i=1}^n x_i y_i - n \bar{x} \bar{y}}{\sum_{i=1}^n x_i^2 - n \bar{x}^2} \quad (16)$$

$$d = \bar{y} - k \bar{x} \quad (17)$$

```

1  for (int i = 1; i <= n; i ++ ) q[i] = read();
2  double sumx = 0, sumy = 0, iq = 0, ii = 0;
3  for (int i = 1; i <= n; i ++ ) {
4      sumx += i, sumy += q[i];
5      iq += i * q[i];
6      ii += i * i;
7  }
8  double k = (iq - sumx * sumy / n) / (ii - sumx * sumx / n);
9  double d = sumy / n - k * sumx / n;
10
11  double ans = 0;
12  for (int i = 1; i <= n; i ++ ) {
13      ans += (k * i + d - q[i]) * (k * i + d - q[i]);
14  }
15  printf("%.6Lf\n", ans);

```

## 3.12 线性基

## temp1

```
1  const int B = 60;
2  struct LinearBasis {
3      array<ll, 61> a, tmp;
4      bool flag;
5      LinearBasis () {
6          fill(a.begin(), a.end(), 0);
7          fill(tmp.begin(), tmp.end(), 0);
8          flag = false;
9      }
10     bool ins (ll x) {
11         for (int i = B; i >= 0; i--) {
12             if (x >> i & 1) {
13                 if (!a[i]) {
14                     a[i] = x;
15                     return true;
16                 } else x ^= a[i];
17             }
18         }
19         flag = true;
20         return false;
21     }
22     bool check (ll x) {
23         for (int i = B; i >= 0; i--) {
24             if (x & (1ll << i)) {
25                 if (!a[i]) return false;
26                 else x ^= a[i];
27             }
28         }
29         return true;
30     }
31     ll qmax (ll ret = 0) {
32         for (int i = B; i >= 0; i--) {
33             ret = max(ret, ret ^ a[i]);
34         }
35         return ret;
36     }
37     ll qmin () {
38         if (flag) return 0;
39         for (int i = 0; i <= B; i++) {
40             if (a[i]) return a[i];
41         }
42     }
43     ll query (ll k) {
44         ll ret = 0, cnt = 0;
45         k -= flag;
46         if (!k) return 0;
47         for (int i = 0; i <= B; i++) {
48             for (int j = i - 1; j >= 0; j--) {
49                 if (a[i] >> j & 1) a[i] ^= a[j];
50             }
51             if (a[i]) tmp[cnt++] = a[i];
52         }
53         if (k >= (1ll << cnt)) return -1;
54         for (int i = 0; i < cnt; i++) {
55             if (k >> i & 1) ret ^= tmp[i];
56         }
57         return ret;
58     }
59     void debug() {
60         for (int i = B; i >= 0; i--) {
```



```

61         dbg(i, a[i]);
62     }
63 }
64 };

```

## temp2

```

1  const int MAXL = 60;
2  struct LinearBasis {
3      int a[MAXL + 1];
4      int cnt = 0;
5
6      LinearBasis () {
7          fill(a, a + MAXL + 1, 0);
8      }
9
10     LinearBasis (int *x, int n) { // 快速构造线性基
11         build (x, n);
12     }
13
14     inline bool insert (int t) { // 线性基动态插入数
15         for (int j = MAXL; j >= 0; j -- ) {
16             if (!t) return false;
17             if (!(t & (1ll << j))) continue;
18
19             if (a[j]) t ^= a[j];
20             else {
21                 cnt ++ ;
22                 // for (int k = 0; k < j; k ++ ) {
23                 //     if (t & (1ll << k)) t ^= a[k];
24                 // }
25                 // for (int k = j + 1; k <= MAXL; k ++ ) {
26                 //     if (a[k] & (1ll << j)) a[k] ^= t;
27                 // }
28                 a[j] = t;
29                 return false;
30             }
31         }
32         return true;
33     }
34
35     void build (int *x, int n) {
36         fill(a, a + MAXL + 1, 0);
37         for (int i = 1; i <= n; i ++ ) insert(x[i]);
38     }
39     inline void rebuild () { // 重构成易于求kth的形式
40         for (int i = 0; i <= MAXL; i ++ ) {
41             for (int j = i - 1; j >= 0; j -- ) {
42                 if (a[i] & (1ll << j)) a[i] ^= a[j];
43             }
44         }
45     }
46
47     inline void mergefrom (const LinearBasis &b) { // 合并两个线性基
48         for (int i = 0; i <= MAXL; i ++ ) insert(b.a[i]);
49     }
50
51     static LinearBasis merge (const LinearBasis &a, const LinearBasis &b) {
52         LinearBasis res = a;
53         for (int i = 0; i <= MAXL; i ++ ) res.insert(b.a[i]);
54         return res;

```

```

55     }
56
57     inline int querymax () { // 查询子集最大异或和
58         int res = 0;
59         for (int i = 0; i <= MAXL; i ++ ) res ^= a[i];
60         return res;
61     }
62     inline int querymin () { // 查询子集最小异或和
63         for (int i = 0; i <= MAXL; i ++ ) if (a[i]) return a[i];
64     }
65
66     inline int kth (int k) { // k大异或和
67         if (cnt < n) k -- ;
68         if (k >= (1ll << cnt)) return -1;
69
70         int ans = 0;
71         for (int i = 0, j = 0; i <= MAXL; i ++ ) {
72             if (a[i]) {
73                 if (k & (1ll << j)) ans ^= a[i];
74                 j ++ ;
75             }
76         }
77         return ans;
78     }
79
80     inline int rank (int x) { // 查询异或和排名，不去重需要 k % mod * qmi(2, n - a.cnt) % mod
+ 1) % mod;
81         int ans = 0;
82         for (int i = 0, j = 0; i <= MAXL; i ++ ) {
83             if (a[i]) {
84                 if (x & (1ll << i)) ans |= (1ll << j);
85                 j ++ ;
86             }
87         }
88         return ans;
89     }
90 };

```

### 3.13 Pollard-Rho

```

1  namespace prime_fac {
2
3      const int S = 8; // 随机算法判定次数，8~10 就够了
4
5      // 龟速乘
6      long long mult_mod(long long a, long long b, long long c) {
7          a %= c, b %= c;
8          long long ret = 0;
9          long long tmp = a;
10         while (b) {
11             if (b & 1) {
12                 ret += tmp;
13                 if (ret > c) ret -= c;
14             }
15             tmp <<= 1;
16             if (tmp > c) tmp -= c;
17             b >>= 1;
18         }
19         return ret;
20     }
21 }

```

```

22 // 快速幂
23 long long qow_mod(long long a, long long n, long long _mod) {
24     long long ret = 1;
25     long long temp = a % _mod;
26     while (n) {
27         if (n & 1) ret = mult_mod(ret, temp, _mod);
28         temp = mult_mod(temp, temp, _mod);
29         n >>= 1;
30     }
31     return ret;
32 }
33
34 // 是合数返回true, 不一定是合数返回false
35 bool check(long long a, long long n, long long x, long long t) {
36     long long ret = qow_mod(a, x, n);
37     long long last = ret;
38     for (int i = 1; i <= t; i++) {
39         ret = mult_mod(ret, ret, n);
40         if (ret == 1 && last != 1 && last != n - 1) return true;
41         last = ret;
42     }
43     if (ret != 1) return true;
44     return false;
45 }
46
47 // 是素数返回true, 不是返回false
48 mt19937 rng(chrono::steady_clock::now().time_since_epoch().count());
49 bool Miller_Rabin(long long n) {
50     if (n < 2) return false;
51     if (n == 2) return true;
52     if ((n & 1) == 0) return false;
53     long long x = n - 1;
54     long long t = 0;
55     while ((x & 1) == 0) { x >>= 1; t++; }
56
57     for (int i = 0; i < S; i++) {
58         long long a = rng() % (n - 1) + 1;
59         if (check(a, n, x, t))
60             return false;
61     }
62
63     return true;
64 }
65
66 long long factor[100]; // 存质因数
67 int tol; // 质因数的个数, 0~tol-1
68
69 long long gcd(long long a, long long b) {
70     long long t;
71     while (b) {
72         t = a;
73         a = b;
74         b = t % b;
75     }
76     if (a >= 0) return a;
77     return -a;
78 }
79
80 long long pollard_rho(long long x, long long c) {
81     long long i = 1, k = 2;
82     long long x0 = rng() % (x - 1) + 1;
83     long long y = x0;

```

```

84     while (1) {
85         i++;
86         x0 = (mult_mod(x0, x0, x) + c) % x;
87         long long d = gcd(y - x0, x);
88         if (d != 1 && d != x) return d;
89         if (y == x0) return x;
90         if (i == k) { y = x0; k += k; }
91     }
92 }
93 // 对n质因数分解，存入factor，k一般设置为107左右
94 void findfac(long long n, int k) {
95     if (n == 1) return;
96     if (Miller_Rabin(n)) {
97         factor[tol++] = n;
98         return;
99     }
100     long long p = n;
101     int c = k;
102     while (p >= n) p = pollard_rho(p, c--);
103     findfac(p, k);
104     findfac(n / p, k);
105 }
106 vector<int> fac(long long n) {
107     tol = 0;
108     vector<int> ret;
109     findfac(n, 107);
110     for (int i = 0; i < tol; i++) ret.push_back(factor[i]);
111     return ret;
112 }
113 }
114
115 //vector<int> fac = prime_fac::fac(n);

```

## 板子2

```

1  using i64 = long long;
2  using i128 = __int128;
3
4  i64 power (i64 a, i64 b, i64 m) {
5      i64 res = 1;
6      for (; b >= 1, a = i128 (a) * a % m) {
7          if (b & 1) {
8              res = i128 (res) * a % m;
9          }
10     }
11     return res;
12 }
13
14 bool isprime (i64 p) {
15     if (p < 2) {
16         return 0;
17     }
18     i64 d = p - 1, r = 0;
19     while (!(d & 1)) {
20         r++;
21         d >>= 1;
22     }
23     int prime[] = { 2, 3, 5, 7, 11, 13, 17, 19, 23 };
24     for (auto a : prime) {
25         if (p == a) {

```

```

26         return true;
27     }
28     i64 x = power (a, d, p);
29     if (x == 1 || x == p - 1) {
30         continue;
31     }
32     for (int i = 0; i < r - 1; i++) {
33         x = i128 (x) * x % p;
34         if (x == p - 1) {
35             break;
36         }
37     }
38     if (x != p - 1) {
39         return false;
40     }
41 }
42 return true;
43 }
44
45 mt19937 rng ((unsigned int)chrono::steady_clock::now ().time_since_epoch ().count ());
46
47 i64 pollard_rho (i64 x) {
48     i64 s = 0, t = 0;
49     i64 c = i64 (rng ()) % (x - 1) + 1;
50     i64 val = 1;
51     for (int goal = 1; ; goal <= 1, s = t, val = 1) {
52         for (int step = 1; step <= goal; step++) {
53             t = (i128 (t) * t + c) % x;
54             val = i128 (val) * abs (t - s) % x;
55             if (step % 127 == 0) {
56                 i64 g = gcd (val, x);
57                 if (g > 1) {
58                     return g;
59                 }
60             }
61         }
62         i64 g = gcd (val, x);
63         if (g > 1) {
64             return g;
65         }
66     }
67 }
68
69 unordered_map<i64, int> getprimes (i64 x) {
70     unordered_map<i64, int> p;
71     function<void (i64)> get = [&](i64 x) {
72         if (x < 2) {
73             return;
74         }
75         if (isprime (x)) {
76             p[x]++;
77             return;
78         }
79         i64 mx = pollard_rho (x);
80         get (x / mx);
81         get (mx);
82     };
83     get (x);
84     return p;
85 }

```

## 3.14 矩阵类

```
1  template<class Type>
2  class Matrix {
3  private:
4      vector<vector<Type>>> data;
5  public:
6      int width, height;
7      Matrix (int height = 0, int width = 0, Type value = 0);
8      Matrix<Type> (const Matrix<Type> &other);
9      Matrix<Type> operator + (const Matrix<Type> &other);
10     Matrix<Type> operator - (const Matrix<Type> &other);
11     Matrix<Type> operator * (const Matrix<Type> &other);
12     Matrix<Type> operator ~();
13     Matrix<Type> pow(int x);
14     vector<Type> operator [] (int row) const;
15     vector<Type>& operator [] (int row);
16     void print();
17     static Matrix<Type> eye(int n);
18 };
19 typedef Matrix<double> Mat;
20 template<class Type>
21 Matrix<Type>::Matrix(const Matrix<Type> &other) {
22     height = other.height;
23     width = other.width;
24     data = other.data;
25 }
26 template<class Type>
27 Matrix<Type>::Matrix(int height_, int width_, Type value_) {
28     height = height_;
29     width = width_;
30     data.resize(height);
31     for (int i = 0; i < height; i++) {
32         data[i].resize(width, value_);
33     }
34 }
35 template<class Type>
36 void Matrix<Type>::print() {
37     for (int i = 0; i < height; i++) {
38         for (int j = 0; j < width; j++) {
39             cout << data[i][j] << ' ';
40         }
41         cout << endl;
42     }
43 }
44 template<class Type>
45 Matrix<Type> Matrix<Type>::operator + (const Matrix<Type> &other) {
46     if (other.height != height || other.width != width) {
47         throw -1;
48     }
49     Matrix<Type> res(height, width);
50     for (int i = 0; i < height; i++) {
51         for (int j = 0; j < width; j++) {
52             res.data[i][j] = data[i][j] + other.data[i][j];
53         }
54     }
55     return res;
56 }
57 template<class Type>
58 Matrix<Type> Matrix<Type>::operator - (const Matrix<Type> &other) {
59     if (other.height != height || other.width != width) {
60         throw -1;
```

```

61     }
62     Matrix<Type> res(height, width);
63     for (int i = 0; i < height; i ++ ) {
64         for (int j = 0; j < width; j ++ ) {
65             res.data[i][j] = data[i][j] - other.data[i][j];
66         }
67     }
68     return res;
69 }
70 template<class Type>
71 Matrix<Type> Matrix<Type> :: operator * (const Matrix <Type> &other) {
72     if (other.height != width) {
73         throw -2;
74     }
75     Matrix<Type> res(height, other.width);
76     for (int i = 0; i < height; i ++ ) {
77         for (int j = 0; j < other.width; j ++ ) {
78             for (int k = 0; k < width; k ++ ) {
79                 res.data[i][j] += data[i][k] * other.data[k][j];
80             }
81         }
82     }
83     return res;
84 }
85 template<class Type>
86 Matrix<Type> Matrix<Type> :: operator ~() {
87     Matrix<Type> res(width, height);
88     for (int i = 0; i < width; i ++ ) {
89         for (int j = 0; j < height; j ++ ) {
90             res[i][j] = data[j][i];
91         }
92     }
93     return res;
94 }
95 template<class Type>
96 vector<Type> Matrix<Type> :: operator [] (int row) const {
97     if (row > height) throw -5;
98     return data[row];
99 }
100 template<class Type>
101 vector<Type>& Matrix<Type> :: operator [] (int row) {
102     if (row > height) throw -5;
103     return data[row];
104 }
105 template<class Type>
106 Matrix<Type> Matrix<Type> :: eye(int n){
107     Matrix<Type> res(n, n);
108     for (int i = 0; i < n; i ++ ){
109         res[i][i] = 1;
110     }
111     return res;
112 }
113 template<class Type>
114 Matrix<Type> pow (Matrix<Type> mat, int x) {
115     Matrix<Type> res = mat.eye;
116     while (x) {
117         if (x & 1) res = res * mat;
118         mat = mat * mat;
119         x >>= 1;
120     }
121     return res;
122 }

```

```

123 void solve() {
124     Mat a(3, 3);
125     a[0] = {1, 1, 1};
126     Mat b(3, 4);
127     (a * b).print();
128 }

```

### 3.15 叉积求极角排序

```

1 //https://codeforces.com/contest/598/problem/C
2 //求两两点之间最小夹角
3 struct Point {
4     int x, y;
5     int id;
6     Point (int x = 0, int y = 0, int id = 0) : x(x), y(y), id(id) {}
7 };
8 int dot (const Point &a, const Point &b) {return a.x * b.x + a.y * b.y;}
9 int det (const Point &a, const Point &b) {return a.x * b.y - a.y * b.x;}
10 void sortPoint (vector<Point> &vec) {
11     vector<Point> p, q;
12     for (auto cur : vec) {
13         if (cur.y > 0 || (cur.y == 0 && cur.x > 0)) p.pb(cur);
14         else q.pb(cur);
15     }
16     sort(all(p), [&](const Point &a, const Point &b) {
17         return det(a, b) > 0;
18     });
19     sort(all(q), [&](const Point &a, const Point &b) {
20         return det(a, b) > 0;
21     });
22     vec.clear();
23     for (auto cur : p) vec.pb(cur);
24     for (auto cur : q) vec.pb(cur);
25 }
26 void solve() {
27     cin >> n;
28     vector<Point> a(n);
29     for (int i = 0; i < n; i++) {
30         cin >> a[i].x >> a[i].y;
31         a[i].id = i;
32     }
33     sortPoint(a);
34     int p = 0, q = n - 1;
35     for (int i = 0; i < n - 1; i++) {
36         Point v1(dot(a[i], a[i + 1]), abs(det(a[i], a[i + 1])));
37         Point v2(dot(a[p], a[q]), abs(det(a[p], a[q])));
38         if (det(v1, v2) > 0) p = i, q = i + 1;
39     }
40     cout << a[p].id + 1 << ' ' << a[q].id + 1 << endl;
41 }
42
43 using T = double;
44 struct Point {
45     T x;
46     T y;
47     Point(T x = 0, T y = 0) : x(x), y(y) {}
48
49     Point &operator+=(const Point &p) {
50         x += p.x, y += p.y;
51         return *this;
52     }

```



```

53     Point &operator-=(const Point &p) {
54         x -= p.x, y -= p.y;
55         return *this;
56     }
57     Point &operator*=(const T &v) {
58         x *= v, y *= v;
59         return *this;
60     }
61     friend Point operator-(const Point &p) {
62         return Point(-p.x, -p.y);
63     }
64     friend Point operator+(Point lhs, const Point &rhs) {
65         return lhs += rhs;
66     }
67     friend Point operator-(Point lhs, const Point &rhs) {
68         return lhs -= rhs;
69     }
70     friend Point operator*(Point lhs, const T &rhs) {
71         return lhs *= rhs;
72     }
73 };
74
75 T dot(const Point &a, const Point &b) {
76     return a.x * b.x + a.y * b.y;
77 }
78
79 T cross(const Point &a, const Point &b) {
80     return a.x * b.y - a.y * b.x;
81 }

```

## 3.16 两圆交面积

```

1 struct Circle {
2     long double x, y;
3     long double r;
4     long double area (const Circle &b) {
5         long double d = sqrt((x - b.x) * (x - b.x) + (y - b.y) * (y - b.y));
6         if (d >= r + b.r) return 0;
7         long double r1 = r, r2 = b.r;
8         if (r1 > r2) swap(r1, r2);
9         if (r2 - r1 >= d) return pi * r1 * r1;
10        long double ang1 = acos((r1 * r1 + d * d - r2 * r2) / (2 * r1 * d));
11        long double ang2 = acos((r2 * r2 + d * d - r1 * r1) / (2 * r2 * d));
12        return ang1 * r1 * r1 - 0.5 * r1 * r1 * sin(ang1 * 2) + ang2 * r2 * r2 - 0.5 * r2
13        * r2 * sin(ang2 * 2);
14    }
15 };

```

## 3.17 全概率公式/贝叶斯公式

全概率公式：

$$(1) \forall i \neq j, A_i A_j = \emptyset$$

$$(2) A_1 \cup A_2 \cup \dots A_n = \Omega.$$

则：

$$P(B) = \sum_{i=1}^n P(A_i)P(B|A_i)$$

贝叶斯公式:

设 $A_1, A_2, \dots, A_n$ 是完备事件组, 且 $P(A_i) > 0 (i = 1, 2, \dots, n)$ ,  $B$ 为任意事件,  $P(B) > 0$ , 则

$$P(A_k|B) = \frac{P(A_k)P(B|A_k)}{P(B)} = \frac{P(A_k)P(B|A_k)}{\sum_{i=1}^n P(A_i)P(B|A_i)}$$

## 3.18 多项式卷积

### 3.18.1 FFT

$$c[x] = \sum a[i] * b[x - i] \quad (18)$$

Ver线下

```
1 typedef complex<double> C;
2 typedef vector<double> vd;
3 void fft(vector<C>& a) {
4     int n = a.size(), L = 31 - __builtin_clz(n);
5     static vector<complex<long double>> R(2, 1);
6     static vector<C> rt(2, 1); // (^ 10% fas te r i f double)
7     for (static int k = 2; k < n; k *= 2) {
8         R.resize(n); rt.resize(n);
9         auto x = polar(1.0L, acos(-1.0L) / k);
10        for (int i = k; i < 2 * k; i++) rt[i] = R[i] = i & 1 ? R[i/2] * x : R[i/2];
11    }
12    vector<int> rev(n);
13    for (int i = 0; i < n; i++)
14        rev[i] = (rev[i / 2] | (i & 1) << L) / 2;
15    for (int i = 0; i < n; i++)
16        if (i < rev[i]) swap(a[i], a[rev[i]]);
17    for (int k = 1; k < n; k *= 2)
18        for (int i = 0; i < n; i += 2 * k) {
19            for (int j = 0; j < k; j++) {
20                C z = rt[j + k] * a[i + j + k]; // (25% fas te r i f hand-ro l led )
21                a[i + j + k] = a[i + j] - z;
22                a[i + j] += z;
23            }
24        }
25 }
26 vd convo (const vd& a, const vd& b) {
27     if (a.empty() || b.empty()) return {};
28     vd res(a.size() + b.size() - 1);
29     int L = 32 - __builtin_clz(res.size()), n = 1 << L;
30     vector<C> in(n), out(n);
31     copy(a.begin(), a.end(), begin(in));
32     for (int i = 0; i < b.size(); i++) in[i].imag(b[i]);
33     fft(in);
34     for (C& x : in) x *= x;
35     for (int i = 0; i < n; i++) out[i] = in[-i & (n - 1)] - conj(in[i]);
36     fft(out);
37     for (int i = 0; i < res.size(); i++) res[i] = imag(out[i]) / (4 * n);
38     return res;
39 }
40
41 void Solve() {
42     int n, m;
43     cin >> n >> m;
44     n++, m++;
```

```

45     vd a(n), b(m);
46     for (auto &x : a) cin >> x;
47     for (auto &x : b) cin >> x;
48     auto c = conv(a, b);
49     for (auto x : c) cout << (int)(x + 0.0001) << ' ';
50 }

```

## Ver线上

```

1 // FFT_MAXN = 2^k
2 // fft_init() to precalc FFT_MAXN-th roots
3 #define rep(i, a, b) for (int i = a; i < (int)b; i++)
4 typedef long double db;
5 const int FFT_MAXN=262144;
6 const db pi=acosl(-1.);
7 struct cp{
8     db a,b;
9     cp operator+(const cp&y)const{return (cp){a+y.a,b+y.b};}
10    cp operator-(const cp&y)const{return (cp){a-y.a,b-y.b};}
11    cp operator*(const cp&y)const{return (cp){a*y.a-b*y.b,a*y.b+b*y.a};}
12    cp operator!()const{return (cp){a,-b};}
13 }nw[FFT_MAXN+1];int bitrev[FFT_MAXN];
14 void dft(cp*a,int n,int flag=1){
15     int d=0;while((1<<d)*n!=FFT_MAXN)d++;
16     rep(i,0,n)if(i<(bitrev[i]>>d))swap(a[i],a[bitrev[i]>>d]);
17     for (int l=2;l<=n;l<=1){
18         int de=FFT_MAXN/l*flag;
19         for (int i=0;i<n;i+=l){
20             cp *le=a+i,*ri=a+i+(l>>1),*w=flag==1?nw:nw+FFT_MAXN;
21             rep(k,0,l>>1){
22                 cp ne=*ri**w;
23                 *ri=*le-ne,*le=*le+ne;
24                 le++,ri++,w+=de;
25             }
26         }
27     }
28     if(flag!=1)rep(i,0,n)a[i].a/=n,a[i].b/=n;
29 }
30 void fft_init(){
31     int L=0;while((1<<L)!=FFT_MAXN)L++;
32     bitrev[0]=0;rep(i,1,FFT_MAXN)bitrev[i]=bitrev[i>>1]>>1|((i&1)<<(L-1));
33     nw[0]=nw[FFT_MAXN]=(cp){1,0};
34     rep(i,0,FFT_MAXN+1)nw[i]=(cp){cosl(2*pi/FFT_MAXN*i),sinl(2*pi/FFT_MAXN*i)}; //very
slow
35 }
36
37 void convo(db*a,int n,db*b,int m,db*c){
38     static cp f[FFT_MAXN>>1],g[FFT_MAXN>>1],t[FFT_MAXN>>1];
39     int N=2;while(N<=n+m)N<=1;
40     rep(i,0,N)
41         if(i&1){
42             f[i>>1].b=(i<=n)?a[i]:0.0;
43             g[i>>1].b=(i<=m)?b[i]:0.0;
44         }else{
45             f[i>>1].a=(i<=n)?a[i]:0.0;
46             g[i>>1].a=(i<=m)?b[i]:0.0;
47         }
48     dft(f,N>>1);dft(g,N>>1);
49     int de=FFT_MAXN/(N>>1);
50     cp qua=(cp){0,0.25},one=(cp){1,0},four=(cp){4,0},*w=nw;

```

```

51     rep(i,0,N>>1){
52         int j=i?(N>>1)-i:0;
53         t[i]=(four*(f[j]*g[j])-(f[j]-f[i])*(g[j]-g[i])*(one+w))*qua;
54         w+=del;
55     }
56     dft(t,N>>1,-1);
57     rep(i,0,n+m+1)c[i]=(i&1)?t[i>>1].a:t[i>>1].b;
58 }
59
60 void mul(int *a,int *b,int n){// n<=N, 0<=a[i],b[i]<mo
61     static cp f[N],g[N],t[N],r[N];
62     int nn=2;while(nn<=n+nn)nn<=1;
63     rep(i,0,nn){
64         f[i]=(i<=n)?(cp){(db)(a[i]>>15),(db)(a[i]&32767)}:(cp){0,0};
65         g[i]=(i<=n)?(cp){(db)(b[i]>>15),(db)(b[i]&32767)}:(cp){0,0};
66     }
67     swap(n,nn);
68     dft(f,n,1);dft(g,n,1);
69     rep(i,0,n){
70         int j=i?n-i:0;
71         t[i]=( (f[i]+!f[j])*(g[j]-g[i]) + (!f[j]-f[i])*(g[i]+!g[j]) )*(cp){0,0.25};
72         r[i]=(!f[j]-f[i])*(g[j]-g[i])*(cp){-0.25,0} + (cp){0,0.25}*(f[i]+!f[j])*
(g[i]+!g[j]);
73     }
74     dft(t,n,-1); dft(r,n,-1);
75     rep(i,0,n)a[i]=( (ll(t[i].a+0.5)%mo<<15) + ll(r[i].a+0.5) + (ll(r[i].b+0.5)%mo<<30)
)%mo;
76 }
77

```

## 3.18.2 NTT

### jly板子

Z函数下不要引入std库

```

1  constexpr int P = 998244353;
2  using i64 = long long;
3  // assume -P <= x < 2P
4  int norm(int x) {
5      if (x < 0) {
6          x += P;
7      }
8      if (x >= P) {
9          x -= P;
10     }
11     return x;
12 }
13 template<class T>
14 T power(T a, int b) {
15     T res = 1;
16     for (; b; b /= 2, a *= a) {
17         if (b % 2) {
18             res *= a;
19         }
20     }
21     return res;
22 }
23 struct Z {
24     int x;

```

```

25     Z(int x = 0) : x(norm(x)) {}
26     int val() const {
27         return x;
28     }
29     Z operator-(const Z &rhs) {
30         return Z(norm(P - x));
31     }
32     Z inv() const {
33         assert(x != 0);
34         return power(*this, P - 2);
35     }
36     Z &operator*=(const Z &rhs) {
37         x = i64(x) * rhs.x % P;
38         return *this;
39     }
40     Z &operator+=(const Z &rhs) {
41         x = norm(x + rhs.x);
42         return *this;
43     }
44     Z &operator-=(const Z &rhs) {
45         x = norm(x - rhs.x);
46         return *this;
47     }
48     Z &operator/=(const Z &rhs) {
49         return *this *= rhs.inv();
50     }
51     friend Z operator*(const Z &lhs, const Z &rhs) {
52         Z res = lhs;
53         res *= rhs;
54         return res;
55     }
56     friend Z operator+(const Z &lhs, const Z &rhs) {
57         Z res = lhs;
58         res += rhs;
59         return res;
60     }
61     friend Z operator-(const Z &lhs, const Z &rhs) {
62         Z res = lhs;
63         res -= rhs;
64         return res;
65     }
66     friend Z operator/(const Z &lhs, const Z &rhs) {
67         Z res = lhs;
68         res /= rhs;
69         return res;
70     }
71     friend std::istream &operator>>(std::istream &is, Z &a) {
72         i64 v;
73         is >> v;
74         a = Z(v);
75         return is;
76     }
77     friend std::ostream &operator<<(std::ostream &os, const Z &a) {
78         return os << a.val();
79     }
80 };
81
82 std::vector<int> rev;
83 std::vector<Z> roots{0, 1};
84 void dft(std::vector<Z> &a) {
85     int n = a.size();
86

```

```

87     if (int(rev.size()) != n) {
88         int k = __builtin_ctz(n) - 1;
89         rev.resize(n);
90         for (int i = 0; i < n; i++) {
91             rev[i] = rev[i >> 1] >> 1 | (i & 1) << k;
92         }
93     }
94
95     for (int i = 0; i < n; i++) {
96         if (rev[i] < i) {
97             std::swap(a[i], a[rev[i]]);
98         }
99     }
100     if (int(roots.size()) < n) {
101         int k = __builtin_ctz(roots.size());
102         roots.resize(n);
103         while ((1 << k) < n) {
104             Z e = power(Z(3), (P - 1) >> (k + 1));
105             for (int i = 1 << (k - 1); i < (1 << k); i++) {
106                 roots[2 * i] = roots[i];
107                 roots[2 * i + 1] = roots[i] * e;
108             }
109             k++;
110         }
111     }
112     for (int k = 1; k < n; k *= 2) {
113         for (int i = 0; i < n; i += 2 * k) {
114             for (int j = 0; j < k; j++) {
115                 Z u = a[i + j];
116                 Z v = a[i + j + k] * roots[k + j];
117                 a[i + j] = u + v;
118                 a[i + j + k] = u - v;
119             }
120         }
121     }
122 }
123 void idft(std::vector<Z> &a) {
124     int n = a.size();
125     std::reverse(a.begin() + 1, a.end());
126     dft(a);
127     Z inv = (1 - P) / n;
128     for (int i = 0; i < n; i++) {
129         a[i] *= inv;
130     }
131 }
132 struct Poly {
133     std::vector<Z> a;
134     Poly() {}
135     Poly(const std::vector<Z> &a) : a(a) {}
136     Poly(const std::initializer_list<Z> &a) : a(a) {}
137     int size() const {
138         return a.size();
139     }
140     void resize(int n) {
141         a.resize(n);
142     }
143     Z operator[](int idx) const {
144         if (idx < size()) {
145             return a[idx];
146         } else {
147             return 0;
148         }

```

```

149     }
150     Z &operator[](int idx) {
151         return a[idx];
152     }
153     Poly mulxk(int k) const {
154         auto b = a;
155         b.insert(b.begin(), k, 0);
156         return Poly(b);
157     }
158     Poly modxk(int k) const {
159         k = std::min(k, size());
160         return Poly(std::vector<Z>(a.begin(), a.begin() + k));
161     }
162     Poly divxk(int k) const {
163         if (size() <= k) {
164             return Poly();
165         }
166         return Poly(std::vector<Z>(a.begin() + k, a.end()));
167     }
168     friend Poly operator+(const Poly &a, const Poly &b) {
169         std::vector<Z> res(std::max(a.size(), b.size()));
170         for (int i = 0; i < int(res.size()); i++) {
171             res[i] = a[i] + b[i];
172         }
173         return Poly(res);
174     }
175     friend Poly operator-(const Poly &a, const Poly &b) {
176         std::vector<Z> res(std::max(a.size(), b.size()));
177         for (int i = 0; i < int(res.size()); i++) {
178             res[i] = a[i] - b[i];
179         }
180         return Poly(res);
181     }
182     friend Poly operator*(Poly a, Poly b) {
183         if (a.size() == 0 || b.size() == 0) {
184             return Poly();
185         }
186         int sz = 1, tot = a.size() + b.size() - 1;
187         while (sz < tot) {
188             sz *= 2;
189         }
190         a.a.resize(sz);
191         b.a.resize(sz);
192         dft(a.a);
193         dft(b.a);
194         for (int i = 0; i < sz; ++i) {
195             a.a[i] = a[i] * b[i];
196         }
197         idft(a.a);
198         a.resize(tot);
199         return a;
200     }
201     friend Poly operator*(Z a, Poly b) {
202         for (int i = 0; i < int(b.size()); i++) {
203             b[i] *= a;
204         }
205         return b;
206     }
207     friend Poly operator*(Poly a, Z b) {
208         for (int i = 0; i < int(a.size()); i++) {
209             a[i] *= b;
210         }

```

```

211     return a;
212 }
213 Poly &operator+=(Poly b) {
214     return (*this) = (*this) + b;
215 }
216 Poly &operator-=(Poly b) {
217     return (*this) = (*this) - b;
218 }
219 Poly &operator*=(Poly b) {
220     return (*this) = (*this) * b;
221 }
222 Poly deriv() const {
223     if (a.empty()) {
224         return Poly();
225     }
226     std::vector<Z> res(size() - 1);
227     for (int i = 0; i < size() - 1; ++i) {
228         res[i] = (i + 1) * a[i + 1];
229     }
230     return Poly(res);
231 }
232 Poly integr() const {
233     std::vector<Z> res(size() + 1);
234     for (int i = 0; i < size(); ++i) {
235         res[i + 1] = a[i] / (i + 1);
236     }
237     return Poly(res);
238 }
239 Poly inv(int m) const {
240     Poly x{a[0].inv()};
241     int k = 1;
242     while (k < m) {
243         k *= 2;
244         x = (x * (Poly{2} - modxk(k) * x)).modxk(k);
245     }
246     return x.modxk(m);
247 }
248 Poly log(int m) const {
249     return (deriv() * inv(m)).integr().modxk(m);
250 }
251 Poly exp(int m) const {
252     Poly x{1};
253     int k = 1;
254     while (k < m) {
255         k *= 2;
256         x = (x * (Poly{1} - x.log(k) + modxk(k))).modxk(k);
257     }
258     return x.modxk(m);
259 }
260 Poly pow(int k, int m) const {
261     int i = 0;
262     while (i < size() && a[i].val() == 0) {
263         i++;
264     }
265     if (i == size() || 1LL * i * k >= m) {
266         return Poly(std::vector<Z>(m));
267     }
268     Z v = a[i];
269     auto f = divxk(i) * v.inv();
270     return (f.log(m - i * k) * k).exp(m - i * k).mulxk(i * k) * power(v, k);
271 }
272 Poly sqrt(int m) const {

```



```

273     Poly x{1};
274     int k = 1;
275     while (k < m) {
276         k *= 2;
277         x = (x + (modxk(k) * x.inv(k)).modxk(k)) * ((P + 1) / 2);
278     }
279     return x.modxk(m);
280 }
281 Poly mulT(Poly b) const {
282     if (b.size() == 0) {
283         return Poly();
284     }
285     int n = b.size();
286     std::reverse(b.a.begin(), b.a.end());
287     return ((*this) * b).divxk(n - 1);
288 }
289 std::vector<Z> eval(std::vector<Z> x) const {
290     if (size() == 0) {
291         return std::vector<Z>(x.size(), 0);
292     }
293     const int n = std::max(int(x.size()), size());
294     std::vector<Poly> q(4 * n);
295     std::vector<Z> ans(x.size());
296     x.resize(n);
297     std::function<void(int, int, int)> build = [&](int p, int l, int r) {
298         if (r - l == 1) {
299             q[p] = Poly{1, -x[l]};
300         } else {
301             int m = (l + r) / 2;
302             build(2 * p, l, m);
303             build(2 * p + 1, m, r);
304             q[p] = q[2 * p] * q[2 * p + 1];
305         }
306     };
307     build(1, 0, n);
308     std::function<void(int, int, int, const Poly &)> work = [&](int p, int l, int r,
const Poly &num) {
309         if (r - l == 1) {
310             if (l < int(ans.size())) {
311                 ans[l] = num[0];
312             }
313         } else {
314             int m = (l + r) / 2;
315             work(2 * p, l, m, num.mulT(q[2 * p + 1]).modxk(m - 1));
316             work(2 * p + 1, m, r, num.mulT(q[2 * p]).modxk(r - m));
317         }
318     };
319     work(1, 0, n, mulT(q[1].inv(n)));
320     return ans;
321 }
322 };
323
324
325 int main() {
326     std::ios::sync_with_stdio(false);
327     std::cin.tie(nullptr);
328
329     int n;
330     std::cin >> n;
331
332     std::vector<Z> fac(n + 1), invfac(n + 1), p(n + 1);
333     fac[0] = p[0] = 1;

```

```

334     for (int i = 1; i <= n; i++) {
335         fac[i] = fac[i - 1] * i;
336         p[i] = p[i - 1] * 26;
337     }
338     invfac[n] = fac[n].inv();
339     for (int i = n; i > 0; i--) {
340         invfac[i - 1] = invfac[i] * i;
341     }
342
343     std::vector<Z> f(n / 3 + 1);
344     for (int i = 0; i <= n / 3; i++) {
345         f[i] = fac[n - 2 * i] * invfac[i] * invfac[n - 3 * i] * p[n - 3 * i];
346     }
347     for (int i = 0; i <= n / 3; i++) {
348         f[i] *= fac[i];
349     }
350     std::vector<Z> h(n / 3 + 1);
351     for (int i = 0; i <= n / 3; i++) {
352         h[i] = invfac[i] * (i % 2 ? -1 : 1);
353     }
354     auto g = Poly(f).mulT(Poly(h));
355
356     for (int i = 0; i <= n; i++) {
357         if (3 * i <= n) {
358             g[i] *= invfac[i];
359             std::cout << g[i];
360         } else {
361             std::cout << 0;
362         }
363         std::cout << " \n"[i == n];
364     }
365
366     return 0;
367 }
368

```

## 无模NTT

```

1 //https://ac.nowcoder.com/acm/contest/31454/E
2 using i64 = long long;
3 constexpr int NTT_N = (1 << 18) + 5;
4 constexpr i64 P = 3152519739159347311;
5 i64 mul(i64 a, i64 b) {
6     return (__int128)a * (__int128)b % P;
7     a %= P, b %= P;
8     return ((a * b - P * (i64)((long double)a / P * b + 0.5)) % P + P) % P;
9 }
10 i64 power(i64 a, i64 b) {
11     i64 ans = 1;
12     while (b) {
13         if (b & 1) ans = mul(ans, a);
14         a = mul(a, a);
15         b >>= 1;
16     }
17     return ans;
18 }
19 int ntt_n;
20 i64 omega[NTT_N], omega_inv[NTT_N];
21
22 void NTT_init(int n) {

```

```

23 ntt_n = n;
24 i64 wn = power(3, (P - 1) / n);
25 omega[0] = omega_inv[0] = 1;
26 for (int i = 1; i < n; i++)
27     omega_inv[n - i] = omega[i] = mul(omega[i - 1], wn);
28 }
29 void NTT (vector<i64> &a, int n, int tp) {
30     for (int i = 1, j = 0, k; i < n - 1; i++) {
31         k = n;
32         do {
33             j ^= (k >>= 1);
34         } while (j < k);
35         if (i < j) swap(a[i], a[j]);
36     }
37     for (int k = 2, m = ntt_n / 2; k <= n; k *= 2, m /= 2) {
38         for (int i = 0; i < n; i += k) {
39             for (int j = 0; j < k / 2; j++) {
40                 i64 w = (tp > 0 ? omega : omega_inv)[m * j];
41                 i64 u = a[i + j], v = mul(w, a[i + j + k / 2]);
42                 a[i + j] = (u + v) % P;
43                 a[i + j + k / 2] = (u - v + P) % P;
44             }
45         }
46     }
47     if (tp < 0) {
48         i64 inv = power(n, P - 2);
49         for (int i = 0; i < n; i++) a[i] = mul(a[i], inv);
50     }
51 }
52 vector<i64> convo (const vector<i64> &a, const vector<i64> &b) {
53     if (a.empty() || b.empty()) return {};
54     int s = a.size() + b.size() - 1, B = 32 - __builtin_clz(s), n = 1 << B;
55     i64 inv = power(n, P - 2);
56     vector<i64> L(a), R(b), out(n);
57     L.resize(n), R.resize(n);
58     NTT(L, n, 1), NTT(R, n, 1);
59     for (int i = 0; i < n; i++) {
60         out[i] = mul(L[i], R[i]);
61     }
62     NTT(out, n, -1);
63     return {out.begin(), out.begin() + s};
64 }
65
66 vector<i64> ans[NTT_N];
67 void Solve() {
68     int n, q;
69     cin >> n >> q;
70     vector<int> p(n + 1), vis(n + 1);
71     for (int i = 1; i <= n; i++) {
72         cin >> p[i];
73     }
74     vector<i64> h(n + 1);
75     set<int> cycLen;
76     for (int i = 1; i <= n; i++) if (!vis[i]) {
77         int u = i;
78         vector<int> cyc;
79         while (!vis[u]) {
80             vis[u] = true;
81             cyc.emplace_back(u);
82             u = p[u];
83         }
84         int m = cyc.size();

```

```

85     vector<i64> f(m), g(m);
86     for (int j = 0; j < m; j++) {
87         f[j] = cyc[j], g[j] = cyc[(m - j) % m];
88         h[j] = 0;
89     }
90     int L = 1;
91     while (L < 2 * m) L <<= 1;
92     NTT_init(L);
93     auto h2 = convo(f, g);
94     for (int j = 0; j < h2.size(); j++) {
95         h[j % m] += h2[j];
96     }
97     if (!cyclen.count(m)) {
98         cyclen.emplace(m);
99         ans[m] = vector<i64> (m, 0);
100    }
101    for (int j = 0; j < m; j++) {
102        ans[m][j] += h[j];
103    }
104 }
105
106 while (q--) {
107     int x;
108     cin >> x;
109     i64 ret = 0;
110     for (auto m : cyclen) {
111         ret += ans[m][x % m];
112     }
113     cout << ret << '\n';
114 }
115 }

```

### 3.18.3 分治NTT

```

1  std::function<Poly(int, int)> convo = [&] (int l, int r) {
2      if (r - l == 1) {
3          return Poly({l, v[l]});
4      }
5      int m = l + r >> 1;
6      return convo(l, m) * convo(m, r);
7  };

```

## 3.19 拓展中国剩余定理 excrt

```

1  constexpr int N = 1e5 + 10;
2  ll exgcd(ll a, ll b, ll &x, ll &y) {
3      if (!b) {
4          x = 1; y = 0;
5          return a;
6      }
7      ll d = exgcd(b, a % b, y, x);
8      y -= (a / b) * x;
9      return d;
10 }
11 ll qmul (ll a, ll b, ll m) {
12     ll ret = 0;
13     while (b) {
14         if (b & 1) ret = (ret + a) % m;
15         a = (a + a) % m;

```

```

16         b >= 1;
17     }
18     return ret % m;
19 }
20 ll excrt (const vector<ll> &m, const vector<ll> &r, int n) {
21     ll ans = r[0], M = m[0], t, y;
22     for (int i = 1; i < n; i++) {
23         ll mi = m[i], ret = ((r[i] - ans) % mi + mi) % mi;
24         ll g = exgcd(M, mi, t, y);
25         if (ret % g) {
26             return -1;
27         }
28         t = qmul(t, ret / g, mi);
29         ans += t * M;
30         M = mi / g * M;
31         ans = (ans % M + M) % M;
32     }
33     return ans;
34 }

```

## 3.20 Point

```

1  using T = double;
2  struct Point {
3      T x;
4      T y;
5      Point(T x = 0, T y = 0) : x(x), y(y) {}
6
7      Point &operator+=(const Point &p) {
8          x += p.x, y += p.y;
9          return *this;
10     }
11     Point &operator-=(const Point &p) {
12         x -= p.x, y -= p.y;
13         return *this;
14     }
15     Point &operator*=(const T &v) {
16         x *= v, y *= v;
17         return *this;
18     }
19     friend Point operator-(const Point &p) {
20         return Point(-p.x, -p.y);
21     }
22     friend Point operator+(Point lhs, const Point &rhs) {
23         return lhs += rhs;
24     }
25     friend Point operator-(Point lhs, const Point &rhs) {
26         return lhs -= rhs;
27     }
28     friend Point operator*(Point lhs, const T &rhs) {
29         return lhs *= rhs;
30     }
31 };
32
33 T dot(const Point &a, const Point &b) {
34     return a.x * b.x + a.y * b.y;
35 }
36
37 T cross(const Point &a, const Point &b) {
38     return a.x * b.y - a.y * b.x;

```

## 4 动态规划

### 4.1 线性DP

#### 4.1.2 最长上升子序列 $O(n \log n)$

```

1 // O(n^2)
2 int f[N], q[N]; // q原数组, f[i]表示以结尾的最长上升序列
3 int mx = 1;
4
5 for (int i = 1; i <= n; i++) {
6     f[i] = 1; // 设f[i]默认为1, 找不到前面数字小于自己的时候就为1
7     for (int j = 1; j < i; j++) {
8         if (q[i] > q[j]) f[i] = max(f[i], f[j] + 1);
9     }
10    mx = max(mx, f[i]);
11 }
12
13 // O(n log n)
14 vector<int> stk;
15 stk.push_back(q[1]);
16 for (int i = 2; i <= n; i++) {
17     if (q[i] > stk.back()) stk.push_back(q[i]); // 如果该元素大于栈顶元素, 将该元素入栈
18     else *lower_bound(stk.begin(), stk.end(), q[i]) = q[i]; // 替换掉第一个大于或者等于这个
    数字的那个数
19 }
20 cout << stk.size() << endl;

```

#### 4.1.3 最长公共子序列 $O(n^2)$

```

1 int a[N], b[N];
2 int f[N][N]; // f[i][j]表示a串前i个元素和b串前j个元素的最长公共子序列长度, 注意下标从1开始
3
4 cin >> n >> m >> a + 1 >> b + 1;
5 for (int i = 1; i <= n; i++) {
6     for (int j = 1; j <= m; j++) {
7         if (a[i] == b[j]) {
8             f[i][j] = f[i - 1][j - 1] + 1;
9         }
10        else {
11            f[i][j] = max(f[i - 1][j], f[i][j - 1]);
12        }
13    }
14 }
15
16 // LCS + 输出字符串
17 cin >> n >> m >> a + 1 >> b + 1;
18 int A = strlen(a + 1), B = strlen(b + 1); // a串b串长度
19 for (int i = A; i >= 1; i--) {
20     for (int j = B; j >= 1; j--) {
21         if (a[i] == b[j]) f[i][j] = f[i + 1][j + 1] + 1;
22         else f[i][j] = max(f[i + 1][j], f[i][j + 1]);
23     }
24 }
25
26 int i = 1, j = 1;

```

```

27 while (i <= A && j <= B) {
28     if (a[i] == b[j]) {
29         cout << a[i]; // 输出最长公共子串
30         i ++ , j ++ ;
31     }
32     else if (f[i][j] == f[i + 1][j]) i ++ ;
33     else j ++ ;
34 }

```

## 4.1.4 最长公共上升子序列 $O(n^2)$

```

1  #include <bits/stdc++.h>
2  using namespace std;
3
4  const int maxn = 3010;
5  int a[maxn], b[maxn];
6  int f[maxn][maxn]; // f[i][j] 代表所有a[1 ~ i]和b[1 ~ j]中以b[j]结尾的公共上升子序列的长度最大
   值;
7  int n;
8
9  int main()
10 {
11     cin >> n;
12     for (int i = 1; i <= n; i ++ ) cin >> a[i];
13     for (int i = 1; i <= n; i ++ ) cin >> b[i];
14
15     for (int i = 1; i <= n; i ++ ){
16         int maxv = 1;
17         for (int j = 1; j <= n; j ++ ){
18             f[i][j] = f[i - 1][j];
19             if (a[i] == b[j]) f[i][j] = max(f[i][j], maxv);
20             if (a[i] > b[j]) maxv = max(maxv, f[i - 1][j] + 1);
21         }
22     }
23
24     int ans = 0;
25     for (int i = 1; i <= n; i ++ ) ans = max(ans, f[n][i]);
26     cout << ans << endl;
27 }

```

## 4.2 背包

### 4.2.1 01背包 $O(nm)$

```

1  // 求恰好装满的最优解, f[0] = 0, f[1~m] = -inf;
2
3  int n, m;
4  int v[N], w[N];
5  int f[N];
6
7  int main()
8  {
9      cin >> n >> m; // n件物品, 背包容量为m
10
11     for (int i = 1; i <= n; i ++ ){
12         int v, w;
13         cin >> v >> w; // v质量, w价值
14         for (int j = m; j >= v; j -- ) f[j] = max(f[j], f[j - v] + w);
15     }
16 }

```

```

15     }
16
17     cout << f[m] << endl;
18
19     return 0;
20 }

```

## 4.2.2 完全背包 $O(nm)$

```

1 // 每件物品有无限个
2 int f[N];
3
4 int main()
5 {
6     cin >> n >> m; // n件物品，背包容量为m
7     for (int i = 0; i <= n; i ++ ){
8         int v, w;
9         cin >> v >> w; // v质量，w价值
10        // 正序枚举
11        for (int j = v; j <= m; j ++ ) f[j] = max(f[j], f[j - v] + w);
12    }
13
14    cout << f[m] << endl;
15    return 0;
16 }

```

## 4.2.3 多重背包

### 4.2.3.1 二进制优化 $O(NV \log s)$

```

1 // 每件物品最多有s_i个，用二进制优化后再做01背包
2 int v[N], w[N];
3 int f[M];
4
5 int main()
6 {
7     cin >> n >> m;
8
9     int cnt = 0; // 表示下标
10    for (int i = 1; i <= n; i ++ ){
11        int a, b, s; // a质量，b价值，s最多个数
12        cin >> a >> b >> s;
13        int k = 1; // 幂次
14        while (k <= s){
15            cnt ++;
16            v[cnt] = a * k;
17            w[cnt] = b * k;
18            s -= k;
19            k *= 2;
20        }
21
22        if (s > 0){ // 表示有大于幂次的部分
23            cnt ++;
24            v[cnt] = a * s;
25            w[cnt] = b * s;
26        }
27    }

```



```

28
29     //n = cnt;
30
31     for (int i = 1; i <= cnt; i ++ ){
32         for (int j = m; j >= v[i]; j -- ){
33             f[j] = max(f[j], f[j - v[i]] + w[i]);
34         }
35     }
36
37     cout << f[m] << endl;
38
39     return 0;
40 }

```

#### 4.2.3.2 单调队列优化 O (NV)

```

1  int f[N], g[N], q[N]; // fj为前i个物品, 在体积j下的最大价值, g为前i-1个物品, 在体积j下的最大价值, q
    为单调队列。
2
3  int main()
4  {
5      cin >> n >> m;
6      for (int i = 0; i < n; i ++ )
7      {
8          int v, w, s;
9          cin >> v >> w >> s;
10         memcpy(g, f, sizeof f);
11         for (int j = 0; j < v; j ++ )
12         {
13             int hh = 0, tt = -1;
14             for (int k = j; k <= m; k += v)
15             {
16                 if (hh <= tt && q[hh] < k - s * v) hh ++ ;
17                 while (hh <= tt && g[q[tt]] - (q[tt] - j) / v * w <= g[k] - (k - j) / v *
w) tt -- ;
18                 q[ ++ tt] = k;
19                 f[k] = g[q[hh]] + (k - q[hh]) / v * w;
20             }
21         }
22     }
23
24     cout << f[m] << endl;
25
26     return 0;
27 }

```

#### 4.2.4 分组背包 O (nms)

```

1  int v[N][N], w[N][N], s[N];
2  int f[N];
3
4  int main()
5  {
6      cin >> n >> m;
7
8      for (int i = 1; i <= n; i ++ ){
9          cin >> s[i];

```

```

10     for (int j = 0; j < s[i]; j ++ ){
11         cin >> v[i][j] >> w[i][j];
12     }
13 }
14
15 for (int i = 1; i <= n; i ++ ){
16     for (int j = m; j >= 0; j -- ){
17         for (int k = 0; k < s[i]; k ++ ){
18             if (v[i][k] <= j) f[j] = max(f[j], f[j - v[i][k]] + w[i][k]);
19         }
20     }
21 }
22
23 cout << f[m] << endl;
24
25 return 0;
26 }

```

## 4.2.5 超大背包

```

1 // 当物品质量过高，则将价值和重量转换
2 int f[maxn]; // f[i]为价值为i下的最小体积
3 void solve()
4 {
5     cin >> n >> m;
6     memset(f, 0x3f, sizeof f);
7     f[0] = 0;
8     for (int i = 1; i <= n; i ++ ){
9         int w, v;
10        cin >> v >> w;
11        // 用价值上限暴力枚举
12        for (int j = 100000; j >= w; j -- ) f[j] = min(f[j], f[j - w] + v);
13    }
14    int ans = 0;
15    for (int i = 1; i <= 100000; i ++ ){
16        if (f[i] <= m) ans = max(ans, i); // 暴力枚举所有价值，当体积能装下即为一组合法解
17    }
18    cout << ans << endl;
19 }

```

## 4.2.6 二维费用背包(有物品数量限制的背包) $O(nmk)$

```

1 int n, V, M;
2 int f[maxn][maxn]; // 表示不超过i的第一费用，j的第二费用的最大价值。
3
4 int main()
5 {
6     cin >> n >> V >> M; // M为第二重最大费用，或为物品数量限制
7     for (int i = 1; i <= n; i ++ ){
8         int v, m, w;
9         cin >> v >> m >> w;
10        for (int j = V; j >= v; j -- ){
11            for (int k = M; k >= m; k -- ){ // 第二重费用，或为物品数量限制
12                f[j][k] = max(f[j][k], f[j - v][k - m] + w);
13            }
14        }
15    }

```

```

16     cout << f[V][M] << endl;
17 }

```

## 4.2.7 混合背包

```

1 // 01背包，完全背包，多重背包的混合
2 // 考虑将01背包和完全背包转化为多重背包，再使用二进制优化求解多重背包
3 int n, m, v[100010], w[100010], dp[100010];
4
5 int main()
6 {
7     cin >> n >> m; // n件物品，容量为m
8     int cnt = 1;
9     for(int i = 1; i <= n; i++)
10    {
11        int a, b, s;
12        cin >> a >> b >> s; //a体积，b价值
13        int k = 1; // 幂次
14        if(s < 0) s = 1; // s<0, 01背包，只能拿一个
15        else if(s == 0) s = m / a; // s=0,完全背包，最多拿m/a个
16        while(k <= s)
17        {
18            v[cnt] = a * k;
19            w[cnt] = b * k;
20            s -= k;
21            k *= 2;
22            cnt++;
23        }
24        if(s > 0)
25        {
26            v[cnt] = s * a;
27            w[cnt] = s * b;
28            cnt++;
29        }
30    }
31    for(int i = 1; i <= cnt; i++)
32    {
33        for(int j = m; j >= v[i]; j--)
34        {
35            dp[j] = max(dp[j], dp[j - v[i]] + w[i]);
36        }
37    }
38    cout << dp[m];
39 }

```

## 4.2.6 背包问题输出方案

```

1 //用res[i]表示背包容量为i时上次选择了第几个物品。
2
3 #include <bits/stdc++.h>
4 using namespace std;
5
6 const int maxn = 101000;
7 int n, m;
8 int dp[maxn], res[maxn];
9 int v[maxn], w[maxn];
10
11 int main () {
12     scanf("%d%d", &n, &m);
13     for (int i = 1; i <= n; i++) scanf("%d%d", &v[i], &w[i]);
14 }

```

```

15     for (int i = 1; i <= n; i ++ ) {
16         for (int j = v[i]; j <= m; j ++ ) {
17             if (dp[i - 1][j] < dp[i - 1][j - v[i]] + w[i]) {
18                 res[j] = v[i];
19                 dp[j] = dp[j - v[i]] + w[i];
20             }
21         }
22     }
23
24     printf("%d\n", dp[m]);
25
26     vector<int> ans;
27     int now = dp[m];
28     while (res[now] != 0) {
29         ans.push_back(res[now]);
30         now -= res[now];
31     }
32
33     for (int i = ans.size() - 1; i >= 0; i -- ) {
34         printf("%lld ", ans[i]);
35     }
36
37 }

```

## 4.3 区间DP $O(n^3)$

```

1  int f[maxn][maxn], g[maxn][maxn]; // 合并i~j所需的最小/最大价值
2  int s[maxn]; // 原数组的前缀和
3  int w[maxn];
4
5  for (int i = 1; i <= n; i ++ ) {
6      cin >> w[i];
7      w[i + n] = w[i]; // 若有环形要求
8  }
9
10 for (int i = 1; i <= n * 2; i ++ ) s[i] = s[i - 1] + w[i];
11
12 memset(f, 0x3f, sizeof f);
13 memset(g, -0x3f, sizeof g);
14
15 for (int len = 1; len <= n; len ++ ) { // 枚举区间长度
16     for (int l = 1; l + len - 1 <= n * 2; l ++ ) { // 枚举左端点 (若无环形则到n)
17         int r = l + len - 1; // 找到右端点
18         if (l == r) f[l][r] = g[l][r] = 0;
19         else {
20             for (int k = l; k < r; k ++ ) { // 枚举分界点
21                 f[l][r] = min(f[l][r], f[l][k] + f[k + 1][r] + s[r] - s[l - 1]); //
minvalue
22                 g[l][r] = max(g[l][r], g[l][k] + g[k + 1][r] + s[r] - s[l - 1]); //
maxvalue
23             }
24         }
25     }
26 }
27
28 int mi = inf, mx = -inf;
29 for (int i = 1; i <= n; i ++ ) {
30     // 若无环形则直接f[1][n];
31     mi = min(mi, f[i][i + n - 1]);
32     mx = max(mx, g[i][i + n - 1]);

```

```

33 }
34 cout << mi << endl;
35 cout << mx << endl;
36
37 /*
38 将ax和ax+1合并为ax*ax+1, 获得(ax-ax+1)^2
39 注: 不用考虑合并的和得到的分数计算方式不相等, 合并完后就是改变ax和ax+1的值, 直接照题目计算即可
40 */
41 void solve() {
42
43     int n;
44     cin >> n;
45     for (int i = 1; i <= n; i++) cin >> a[i];
46     for (int i = 1; i <= n; i++) {
47         s[i][i] = a[i];
48         for (int j = i + 1; j <= n; j++) {
49             s[i][j] = s[i][j - 1] * a[j] % mod;
50         }
51     }
52
53     for (int len = 2; len <= n; len++) {
54         for (int l = 1; l <= n - len + 1; l++) {
55             int r = l + len - 1;
56             for (int k = l; k < r; k++) {
57                 dp[l][r] = max(dp[l][r], dp[l][k] + dp[k + 1][r] + (s[l][k] - s[k + 1]
58 [r]) * (s[l][k] - s[k + 1][r]));
59             }
60         }
61         cout << dp[1][n] << endl;
62     }
63
64 }

```

## 4.4 树形DP

```

1 // 1. 树中最长路径, 求距离型
2 int dfs(int u, int f) {
3     int dist = 0;
4     int d1 = 0, d2 = 0; // d1为最长路径, d2为次长路径
5
6     for (int i = h[u]; i != -1; i = ne[i]) {
7         int j = e[i];
8         if (j == f) continue;
9         int d = dfs(j, u) + w[i];
10        dist = max(dist, d);
11        if (d >= d1) d2 = d1, d1 = d;
12        else if (d > d2) d2 = d;
13    }
14    ans = max(ans, d1 + d2);
15    return dist;
16 }
17 int main () {
18     cin >> n;
19     memset(h, -1, sizeof h);
20     for (int i = 1; i < n; i++) {
21         int a, b, c;
22         cin >> a >> b >> c;
23         add(a, b, c), add(b, a, c);
24     }
25     dfs(1, -1);

```

```

26
27     cout << ans << endl;
28 }
29
30 //

```

## 4.5 状压DP

## 4.6 数位DP

```

1 // 统计0~N有多少位含i
2 int dgt(int n) {
3     int res = 0;
4     while (n) ++res, n /= 10;
5     return res;
6 }
7
8 int count (int n, int i) {
9     int res = 0, d = dgt(n);
10    for (int j = 1; j <= d; j++) {
11        int p = pow(10, j - 1), l = n / p / 10, r = n % p, dj = n / p % 10;
12        if (i) res += l * p;
13        if (!i && l) res += (l - 1) * p;
14        if ( (dj > i) && (i || l) ) res += p;
15        if ( (dj == i) && (i || l) ) res += r + 1;
16    }
17    return res;
18 }
19
20 // 统计数位上至少有k个某数
21 #include <bits/stdc++.h>
22 using namespace std;
23
24 const int maxn = 210;
25 int f[maxn][maxn];
26 int k, b;
27
28 void init () {
29     for (int i = 0; i < maxn; i++) {
30         for (int j = 0; j <= i; j++) {
31             if (!j) f[i][j] = 1;
32             else f[i][j] = f[i - 1][j] + f[i - 1][j - 1];
33         }
34     }
35 }
36
37 int dp (int n) {
38     if (!n) return 0;
39
40     vector<int> nums;
41     while (n) nums.push_back(n % b), n /= b;
42
43     int res = 0;
44     int ls = 0;
45
46     for (int i = nums.size() - 1; i >= 0; i--) {
47         int x = nums[i];
48         if (x) {
49             res += f[i][k - ls];

```

```

50         if (x > 1) {
51             if (k - ls - 1 >= 0) res += f[i][k - ls - 1];
52             break;
53         }
54         else {
55             ls ++ ;
56             if (ls > k) break;
57         }
58     }
59     if (!i && ls == k) res ++ ;
60 }
61
62 return res;
63 }
64
65 int main () {
66     init ();
67     int l, r;
68     cin >> l >> r >> k >> b;
69     cout << dp(r) << ' ' << dp(l - 1) << endl;
70 }
71
72
73 // 统计l~r中有多少个满足条件的数（单调增，减，不出现某数等）
74 #include <bits/stdc++.h>
75 using namespace std;
76
77 const int maxn = 15;
78
79 int f[maxn][maxn];
80
81 void init () {
82     for (int i = 0; i <= 9; i ++ ) f[1][i] = 1;
83
84     for (int i = 2; i < maxn; i ++ ) {
85         for (int j = 0; j <= 9; j ++ ) {
86             for (int k = j; k <= 9; k ++ ) {
87                 f[i][j] += f[i - 1][k];
88             }
89         }
90     }
91 }
92
93 int dp(int n) {
94     if (!n) return 1;
95
96     vector<int> nums;
97     while (n) nums.push_back(n % 10), n /= 10;
98     int res = 0;
99     int ls = 0;
100
101     for (int i = nums.size() - 1; i >= 0; i -- ) {
102         int x = nums[i];
103         for (int j = ls; j < x; j ++ ) {
104             res += f[i + 1][j];
105         }
106
107         if (x < ls) break;
108         ls = x;
109
110         if (!i) res ++ ;
111     }

```

```

112
113         return res;
114     }
115
116     int main () {
117         init();
118         int l, r;
119         while (cin >> l >> r) {
120             cout << dp(r) - dp(l - 1) << endl;
121         }
122     }

```

## 4.7 单调队列优化DP

```

1 //单调队列常用来优化：i的前m个范围内区间的最值问题。（注意，此处的范围都是一个定值
2
3 //n个物品中，连续m个中至少选出一个，总价值最小。
4 //或 不能连续k个在一起，即至少从k+1个中选出一个不选。
5 #include <bits/stdc++.h>
6 using namespace std;
7
8 const int maxn = 200010;
9 int n, m;
10 int w[maxn], dp[maxn];
11 int q[maxn];
12
13 int main () {
14     scanf("%d%d", &n, &m);
15     for (int i = 1; i <= n; i ++ ) scanf("%d", &w[i]);
16
17     int hh = 0, tt = 0;
18     for (int i = 1; i <= n; i ++ ) {
19         while (hh <= tt && i - q[hh] > m) hh ++ ;
20         dp[i] = dp[q[hh]] + w[i];
21         while (hh <= tt && dp[q[tt]] >= dp[i]) tt -- ;
22         q[ ++ tt] = i;
23     }
24
25     if (n + 1 - m > q[hh]) hh ++ ;
26     printf("%d\n", dp[q[hh]]);
27 }
28

```

## 5. 字符串

### 5.1 KMP

```

1 const int mxn = 1e6 + 100;
2 struct KMP {
3     int ne[mxn], len;
4     string t;
5     void clear() {
6         len = ne[0] = ne[1] = 0;
7     }
8     void init (string s) { /*1-bas*/
9         len = s.size() - 1;
10        t = s;
11        for (int i = 2; i <= len; i ++ ) {

```



```

12         ne[i] = ne[i - 1];
13         while (ne[i] && s[i] != s[ne[i] + 1]) ne[i] = ne[ne[i]];
14         ne[i] += (s[i] == s[ne[i] + 1]);
15     }
16 }
17 vector<int> match (string s) { /*get start_pos*/
18     int len_s = s.size() - 1;
19     vector<int> st_pos(0);
20     for (int i = 1, j = 1; i <= len_s;) {
21         while (j != 1 && s[i] != t[j]) j = ne[j - 1] + 1;
22         if (s[i] == t[j]) j ++ , i ++ ;
23         else i ++ ;
24         if (j == len + 1) {
25             st_pos.push_back(i - j + 1);
26             j = ne[len] + 1;
27         }
28     }
29     return st_pos;
30 }
31 void debug () {
32     for (int i = 0; i <= len; i ++ ) {
33         printf("[debug] nxt[%d]=%d\n", i, ne[i]);
34     }
35 }
36 /* 循环周期 形如 acaca 中 ac 是一个合法周期 */
37 vector<int> periodic() {
38     vector<int> ret;
39     int now = len;
40     while (now) {
41         now = ne[now];
42         ret.push_back(len - now);
43     }
44     return ret;
45 }
46 /* 循环节 形如 acac 中ac、acac是循环节，aca不是*/
47 vector<int> periodic_loop() {
48     vector<int> ret;
49     for (int x : periodic()) {
50         if (len % x == 0) ret.push_back(x);
51     }
52     return ret;
53 }
54 int min_periodic_loop() {
55     return periodic_loop()[0];
56 }
57 }kmp;

```

## 5.2 Manacher

```

1 struct Manacher {
2     int lc[maxn];
3     string ch;
4     int N;
5     // Manacher(string s) {init(s); manacher();}
6     /* s 1 bas */
7     void init (string s) {
8         int n = sz(s) - 1;
9         ch.resize(n * 2 + 10);
10        ch[n * 2 + 1] = '#';
11        ch[0] = '@';

```

```

12     ch[n * 2 + 2] = '\0';
13     for (int i = n; i >= 1; i -- ) {
14         ch[i * 2] = s[i]; ch[i * 2 - 1] = '#';
15     }
16     N = 2 * n + 1;
17 }
18 void work() {
19     lc[1] = 1; int k = 1;
20     for (int i = 2; i <= N; i ++ ) {
21         int p = k + lc[k] - 1;
22         if (i <= p) {
23             lc[i] = min(lc[2 * k - i], p - i + 1);
24         }
25         else lc[i] = 1;
26         while (ch[i + lc[i]] == ch[i - lc[i]]) lc[i] ++ ;
27         if (i + lc[i] > k + lc[k]) k = i;
28     }
29 }
30 void debug () {
31     puts(ch.c_str());
32     for (int i = 1; i <= N; i ++ ) {
33         printf("lc[%d]=%d\n", i, lc[i]);
34     }
35 }
36 }manacher;
37
38 void solve() {
39     string s; cin >> s;
40     s = " " + s;
41     manacher.init(s); manacher.work();
42     manacher.debug();
43 }

```

## 5.3 Hash

```

1  using ull = unsigned long long;
2  struct Hash {
3      static constexpr int sigma = 60 * 60; /* 字符集大小 */
4      static constexpr int HASH_CNT = 1; /* hash次数 */
5      ull Prime_Pool[5] = {1998585857u, 2333333333u};
6      ull Seed_Pool[5] = {146527, 911, 19260817, 91815541};
7      ull Mod_Pool[5] = {998244353, 29123, 1000000009, 4294967291u};
8      ull seed, mod;
9      vector<array<ull, HASH_CNT>> base, sum;
10     int perm[sigma];
11
12     Hash (const string &s) {
13         Init(s);
14     }
15     void Init (const string &s) {
16         int n = s.size() - 1;
17         base.resize(n + 1, {}), sum.resize(n + 1, {});
18         for (int _ = 0; _ < HASH_CNT; _++) {
19             seed = Seed_Pool[_], mod = Mod_Pool[_];
20             base[0][_] = 1;
21             for (int i = 1; i <= n; i++) {
22                 base[i][_] = base[i - 1][_] * seed % mod;
23                 sum[i][_] = (sum[i - 1][_] * seed % mod + s[i]) % mod;
24             }
25         }

```

```

26     }
27     array<ull, HASH_CNT> hash (int l, int r) {
28         array<ull, HASH_CNT> rt = {};
29         for (int _ = 0; _ < HASH_CNT; _++) {
30             int mod = Mod_Pool[_];
31             rt[_] = (sum[r][_] - sum[l - 1][_] * base[r - l + 1][_] % mod + mod) % mod;
32         }
33         return rt;
34     }
35 };

```

```

1  const int sigma = 60 * 60; /* 字符集大小 */
2  const int HASH_CNT = 2; /* hash次数 */
3  int s[maxn];
4  /* char* 1-bas
5   * sum[i] = s[i]+s[i-1]*Seed+s[i-2]*Seed^2+...+s[1]*Seed^(i-1)*/
6  ULL Prime_Pool[] = {1998585857u1, 2333333333u1};
7  ULL Seed_Pool[] = {911, 146527, 19260817, 91815541};
8  ULL Mod_Pool[] = {29123, 998244353, 1000000009, 4294967291u1};
9  struct Hash {
10     ULL seed, mod;
11     ULL base[maxn], sum[maxn];
12     int perm[sigma];
13     void init (int seedindex, int modindex) {
14         seed = Seed_Pool[seedindex], mod = Mod_Pool[modindex];
15         base[0] = 1;
16         for (int i = 1; i <= n; i++) {
17             base[i] = base[i - 1] * seed % mod;
18         }
19         for (int i = 1; i <= n; i++) {
20             sum[i] = (sum[i - 1] * seed % mod + s[i]) % mod;
21         }
22     }
23     /*random_shuffle 离散化id, 防止kill_hash*/
24     void index_init(int seedindex, int modindex) {
25         seed = Seed_Pool[seedindex], mod = Mod_Pool[modindex];
26         base[0] = 1;
27         for (int i = 1; i <= n; i++) {
28             base[i] = base[i - 1] * seed % mod;
29         }
30         iota (perm + 1, perm + sigma + 1, 1);
31         random_shuffle(perm + 1, perm + sigma + 1);
32         for (int i = 1; i <= n; i++) {
33             sum[i] = (sum[i - 1] * seed % mod + perm[s[i]]) % mod;
34         }
35     }
36     ULL gethash (int l, int r) {
37         return (sum[r] - sum[l - 1] * base[r - l + 1] % mod + mod) % mod;
38     }
39 }hasher[HASH_CNT];
40 pair<ULL, ULL> hashrange(int l, int r) {
41     return mkp(hasher[0].gethash(l, r), hasher[1].gethash(l, r));
42 }
43 map<char, int> id; int idcnt;
44
45 void solve() {
46     read(n), read(m);
47     string a; cin >> a;
48     for (int i = 0; i < n; i++) {
49         if (!id.count(a[i])) id[a[i]] = ++ idcnt;

```

```

50     s[i + 1] = id[a[i]];
51 }
52 for (int i = 0; i < HASH_CNT; i++) hasher[i].index_init(i, i);
53
54 while (m--) {
55     int l1, r1, l2, r2;
56     read(l1), read(r1), read(l2), read(r2);
57     if (hashrange(l1, r1) == hashrange(l2, r2)) puts("Yes");
58     else puts("No");
59 }
60 }

```

## 二维哈希

```

1  constexpr int N = 2e3 + 10;
2  int a[N + 5][N + 5], n, m;
3  struct Hash1 {
4      const int p1 = 31, p2 = 1331, mod = 1e9 + 9;
5      void add (int& x, int y) {
6          if ((x += y) >= mod)
7              x -= mod;
8      }
9      void sub (int& x, int y) {
10         if ((x -= y) < 0)
11             x += mod;
12     }
13     int invth1[N + 5], invth2[N + 5];
14     int Pow1[N + 5], Pow2[N + 5];
15     int sum1[N + 5][N + 5], sum2[N + 5][N + 5];
16     int power (int a, int x) {
17         int ans = 1;
18         while (x) {
19             if (x & 1)
20                 ans = 1ll * ans * a % mod;
21             a = 1ll * a * a % mod;
22             x >>= 1;
23         }
24         return ans;
25     }
26     int inv (int a) {
27         return power (a, mod - 2);
28     }
29     void init () {
30         Pow1[0] = Pow2[0] = invth1[0] = invth2[0] = 1;
31         int invp1 = inv (p1), invp2 = inv (p2);
32         for (int i = 1; i < N + 5; i++) {
33             Pow1[i] = 1ll * Pow1[i - 1] * p1 % mod;
34             Pow2[i] = 1ll * Pow2[i - 1] * p2 % mod;
35             invth1[i] = 1ll * invth1[i - 1] * invp1 % mod;
36             invth2[i] = 1ll * invth2[i - 1] * invp2 % mod;
37         }
38         for (int i = 1; i <= n; i++)
39             for (int j = 1; j <= m; j++) {
40                 sum1[i][j] = 1ll * Pow1[i] * Pow2[j] % mod * a[i][j] % mod;
41                 sum2[i][j] = 1ll * Pow1[n - i + 1] * Pow2[m - j + 1] % mod * a[i][j] %
mod;
42             }
43         for (int i = 1; i <= n; i++)
44             for (int j = 1; j <= m; j++) {
45                 add (sum1[i][j], sum1[i - 1][j]);
46                 add (sum1[i][j], (sum1[i][j - 1] - sum1[i - 1][j - 1] + mod) % mod);

```

```

47         add (sum2[i][j], sum2[i - 1][j]);
48         add (sum2[i][j], (sum2[i][j - 1] - sum2[i - 1][j - 1] + mod) % mod);
49     }
50 }
51 bool check (int l, int r, int u, int d) {
52     int g1 = 0;
53     add (g1, sum1[u][d]);
54     sub (g1, sum1[l - 1][d]);
55     sub (g1, sum1[u][r - 1]);
56     add (g1, sum1[l - 1][r - 1]);
57     int g2 = 0;
58     add (g2, sum2[u][d]);
59     sub (g2, sum2[l - 1][d]);
60     sub (g2, sum2[u][r - 1]);
61     add (g2, sum2[l - 1][r - 1]);
62     g1 = 111 * g1 * invth1[l - 1] % mod * invth2[r - 1] % mod;
63     g2 = 111 * g2 * invth1[n - u] % mod * invth2[m - d] % mod;
64     return g1 == g2;
65 }
66 };
67 Hash1 h1;
68 struct Hash2 {
69     const int p1 = 131, p2 = 911, mod = 1e9 + 9;
70     void add (int& x, int y) {
71         if ((x += y) >= mod)
72             x -= mod;
73     }
74     void sub (int& x, int y) {
75         if ((x -= y) < 0)
76             x += mod;
77     }
78     int invth1[N + 5], invth2[N + 5];
79     int Pow1[N + 5], Pow2[N + 5];
80     int sum1[N + 5][N + 5], sum2[N + 5][N + 5];
81     int power (int a, int x) {
82         int ans = 1;
83         while (x) {
84             if (x & 1)
85                 ans = 111 * ans * a % mod;
86             a = 111 * a * a % mod;
87             x >>= 1;
88         }
89         return ans;
90     }
91     int inv (int a) {
92         return power (a, mod - 2);
93     }
94     void init () {
95         Pow1[0] = Pow2[0] = invth1[0] = invth2[0] = 1;
96         int invp1 = inv (p1), invp2 = inv (p2);
97         for (int i = 1; i < N + 5; i++) {
98             Pow1[i] = 111 * Pow1[i - 1] * p1 % mod;
99             Pow2[i] = 111 * Pow2[i - 1] * p2 % mod;
100             invth1[i] = 111 * invth1[i - 1] * invp1 % mod;
101             invth2[i] = 111 * invth2[i - 1] * invp2 % mod;
102         }
103         for (int i = 1; i <= n; i++)
104             for (int j = 1; j <= m; j++) {
105                 sum1[i][j] = 111 * Pow1[i] * Pow2[j] % mod * a[i][j] % mod;
106                 sum2[i][j] = 111 * Pow1[n - i + 1] * Pow2[m - j + 1] % mod * a[i][j] %
mod;
107             }

```

```

108     for (int i = 1; i <= n; i++)
109         for (int j = 1; j <= m; j++) {
110             add (sum1[i][j], sum1[i - 1][j]);
111             add (sum1[i][j], (sum1[i][j - 1] - sum1[i - 1][j - 1] + mod) % mod);
112             add (sum2[i][j], sum2[i - 1][j]);
113             add (sum2[i][j], (sum2[i][j - 1] - sum2[i - 1][j - 1] + mod) % mod);
114         }
115     }
116     bool check (int l, int r, int u, int d) {
117         int g1 = 0;
118         add (g1, sum1[u][d]);
119         sub (g1, sum1[l - 1][d]);
120         sub (g1, sum1[u][r - 1]);
121         add (g1, sum1[l - 1][r - 1]);
122         int g2 = 0;
123         add (g2, sum2[u][d]);
124         sub (g2, sum2[l - 1][d]);
125         sub (g2, sum2[u][r - 1]);
126         add (g2, sum2[l - 1][r - 1]);
127         g1 = 111 * g1 * invth1[l - 1] % mod * invth2[r - 1] % mod;
128         g2 = 111 * g2 * invth1[n - u] % mod * invth2[m - d] % mod;
129         return g1 == g2;
130     }
131 };
132 Hash2 h2;
133 bool check (int a, int b, int c, int d) {
134     if (!(a >= 1 && a <= n && b >= 1 && b <= m && c >= 1 && c <= n && d >= 1 && d <= m
&& a <= c && b <= d))
135         return false;
136     return h1.check (a, b, c, d) && h2.check (a, b, c, d);
137 }

```

## 5.4 ACAM

```

1  const int maxn = 300010;
2  queue<int> q;
3  //int vis[maxn]; 用于做出现次数
4  struct Aho_Corasick_Automaton {
5      int c[maxn][26], fail[maxn], cnt, val[maxn];
6      // int flag[maxn], in[maxn], ans[maxn]; // 用于做出现次数
7
8      void ins (string s) { // 不加x,即为出现个数
9          int len = s.size();
10         int now = 0;
11         for (int i = 0; i < len; i++) {
12             int v = s[i] - 'a';
13             if (!c[now][v]) c[now][v] = ++ cnt;
14             now = c[now][v];
15         }
16         // if (!flag[now]) flag[now] = x; // 用于做出现次数
17         // val[x] = flag[now];
18
19         // val[now]++; // 用于做出现个数
20
21     }
22
23     void build() {
24         for (int i = 0; i < 26; i++) {
25             if (c[0][i]) {
26                 fail[c[0][i]] = 0;

```

```

27     q.push(c[0][i]);
28     }
29 }
30 while (q.size()) {
31     int t = q.front(); q.pop();
32     for (int i = 0; i < 26; i ++ ) {
33         if (c[t][i]) {
34             fail[c[t][i]] = c[fail[t]][i];
35             // in[fail[c[t][i]]] ++ ; // 用于做出现次数
36             q.push(c[t][i]);
37         }
38         else c[t][i] = c[fail[t]][i];
39     }
40 }
41 }
42 void query (char *s) {
43     int len = strlen(s);
44     int now = 0;
45     // int ans = 0;
46     for (int i = 0; i < len; i ++ ) {
47         now = c[now][s[i] - 'a'];
48         // ans[now] ++ ; // 用于做出现次数
49         // for (int t = now; t && ~val[t]; t = fail[t]) { // 用于做出现个数
50         //     ans += val[t], val[t] = -1;
51         // }
52     }
53     // return ans; 最终出现个数
54 }
55
56 // void topsort() { // 用于做出现次数
57 //     for (int i = 1; i <= cnt; i ++ ) {
58 //         if (!in[i]) q.push(i);
59 //     }
60 //     while (q.size()) {
61 //         int t = q.front(); q.pop();
62 //         vis[flag[t]] = ans[t];
63 //         int v = fail[t];
64 //         in[v] -- ;
65 //         ans[v] += ans[t];
66 //         if (!in[v]) q.push(v);
67 //     }
68 // }
69
70 void clear () {
71     cnt = 0;
72     memset(c, 0, sizeof c);
73     memset(val, 0, sizeof val);
74     memset(fail, 0, sizeof fail);
75     // memset(flag, 0, sizeof flag); // 用于做出现次数
76     // memset(ans, 0, sizeof ans);
77     // memset(in, 0, sizeof in);
78 }
79 }acam;
80 int n;
81 string p[200];
82 char t[1000010];
83
84 // vis[AC.val[i]] 第i个串的出现个数.
85 int main () {
86     while (scanf("%d", &n), n) {
87         acam.clear();
88         for (int i = 1; i <= n; i ++ ) {

```

```

89     cin >> p[i];
90     acam.ins(p[i], i);
91 }
92 acam.build();
93 scanf("%s", t);
94 acam.query(t);
95 acam.topsort();
96
97 int mx = 0;
98 for (int i = 1; i <= n; i ++ ) {
99     mx = max(mx, vis[acam.val[i]]);
100    //      printf("%d\n", vis[AC.val[i]]);
101 }
102
103 printf("%d\n", mx);
104 for (int i = 1; i <= n; i ++ ) {
105     if (vis[acam.val[i]] == mx) cout << p[i] << endl;
106 }
107 }
108 }

```

## 5.5 最小表示法

```

1  #include <bits/stdc++.h>
2  using namespace std;
3
4  char s[1000];
5
6  int main () {
7      scanf("%s", s + 1);
8      int n = strlen(s + 1);
9      for (int i = 1; i <= n; i ++ ) {
10         s[n + i] = s[i];
11     }
12     int i = 1, j = 2, k;
13     while (i <= n && j <= n) {
14         for (k = 0; k < n && s[i + k] == s[j + k]; k ++ );
15         if (k == n) break;
16         if (s[i + k] > s[j + k]) {
17             i += k + 1;
18             if (i == j) i ++ ;
19         }
20         else {
21             j += k + 1;
22             if (i == j) j ++ ;
23         }
24     }
25     int ans = min(i, j);
26     for (int i = ans; i <= ans + n - 1; i ++ ) printf("%c", s[i]);
27 }

```

## 5.6 Trie

```

1  // 字典树
2  const int mxn = 1e6 + 100;
3  struct Trie {
4      int nxt[mxn << 1][26], cnt[mxn << 1];
5      int c = 0;
6      void clear() {
7          c = 0;

```



```

8         ms(nxt[0], 0);
9         ms(cnt, 0);
10    }
11
12    inline int newnode () {
13        c ++ ;
14        return c;
15    }
16
17    void insert (string s) {
18        int u = 0, now = 0;
19        while (now < sz(s)) {
20            u = insert(u, s[now] - 'a');
21            now ++ ;
22        }
23        cnt[u] ++ ;
24    }
25    inline int insert (int pre, int ch) {
26        return nxt[pre][ch] ? nxt[pre][ch] : nxt[pre][ch] = newnode();
27    }
28
29    inline int query (string s) {
30        int u = 0, now = 0;
31        while (now < sz(s)) {
32            if (!nxt[u][s[now] - 'a']) return 0;
33            else u = nxt[u][s[now] - 'a'];
34            now ++ ;
35        }
36        return cnt[u];
37    }
38 }trie;
39
40 void solve() {
41     read(n);
42     for (int i = 1; i <= n; i ++ ) {
43         string op, a;
44         cin >> op >> a;
45         if (op == "I") trie.insert(a);
46         else printf("%lld\n", trie.query(a));
47     }
48 }
49
50
51 // 01trie 最大区间异或和
52 const int mxn = 5e5 + 100;
53 struct Trie {
54     int nxt[mxn << 2][2], l[mxn << 2];
55     int cnt, ans1, ansr, ansv;
56
57     inline void init () {
58         cnt = 0; ansv = -1;
59         ms(nxt[0], 0);
60         ms(l, 0x3f);
61     }
62
63     inline int create () {
64         cnt ++ ;
65         ms(nxt[cnt], 0);
66         return cnt;
67     }
68
69     inline void insert (int id, int x) {

```

```

70     int u = 0;
71     for (int i = 31; i >= 0; i -- ) {
72         int t = ((x >> i) & 1);
73         if (!nxt[u][t]) nxt[u][t] = create();
74         u = nxt[u][t];
75     }
76     l[u] = id;
77     // l[u] = min(l[u], id);
78 }
79 inline void query (int id, int x) {
80     int u = 0, res = 0;
81     // de(ansv); de(x);
82     for (int i = 31; i >= 0; i -- ) {
83         int t = ((x >> i) & 1);
84         if (nxt[u][!t]) {
85             u = nxt[u][!t];
86             res += 1ll << i;
87         }
88         else u = nxt[u][t];
89     }
90     // de(id); de(res);
91     // if (res == ansv) {
92     //     if (l[u] < ans1) {
93     //         ans1 = l[u]; ansr = id;
94     //     }
95     // }
96     if (res > ansv) {
97         ansv = res; ans1 = l[u]; ansr = id;
98     }
99 }
100 /*
101 带删除的操作，需要另开tag数组
102 查询时需额外判断 tag[nxt[u][!t]]
103 inline void erase (int x) {
104     int u = 0;
105     for (int i = 31; i >= 0; i -- ) {
106         int t = ((x >> i) & 1);
107         u = nxt[u][t];
108         tag[u] -- ;
109     }
110 }
111 */
112 }trie;
113
114 void solve() {
115     read(n);
116     trie.init(); trie.insert(0, 0);
117     int sum = 0;
118     for (int i = 1; i <= n; i ++ ) {
119         int x; read(x); sum ^= x;
120         trie.query(i, sum); trie.insert(i, sum);
121     }
122     printf("%lld %lld %lld\n", trie.ansv, trie.ans1 + 1, trie.ansr);
123 }
124
125
126 // 可持久化01trie
127 const int mxn = 6e5 + 100;
128 struct Tire {
129     int nxt[mxn * 25][2], sum[mxn * 25];
130     int root[mxn * 25];
131     int cnt;

```

```

132
133     inline void init () {
134         cnt = 0;
135         ms(nxt[0], 0);
136         root[0] = root[1] = sum[0] = sum[1] = 0;
137     }
138
139     inline int create () {
140         cnt ++ ;
141         ms(nxt[cnt], 0);
142         return cnt;
143     }
144
145     inline int insert (int x, int pre) {
146         int u = ++ cnt, t = u;
147         for (int i = 30; i >= 0; i -- ) {
148             int t = ((x >> i) & 1);
149             nxt[u][0] = nxt[pre][0], nxt[u][1] = nxt[pre][1];
150             sum[u] = sum[pre] + 1;
151             nxt[u][t] = ++ cnt;
152             u = nxt[u][t], pre = nxt[pre][t];
153         }
154         sum[u] = sum[pre] + 1;
155         return t;
156     }
157
158     inline int query (int x, int l, int r) {
159         int res = 0;
160         for (int i = 30; i >= 0; i -- ) {
161             int t = !((x >> i) & 1);
162             if (sum[nxt[r][t]] - sum[nxt[l][t]] > 0) {
163                 res |= (1ll << i);
164                 l = nxt[l][t], r = nxt[r][t];
165             }
166             else l = nxt[l][!t], r = nxt[r][!t];
167         }
168         return res;
169     }
170 }trie;
171
172 void solve() {
173     read(n), read(m);
174     int now = 0;
175     n ++ ;
176     trie.root[1] = trie.insert(now, trie.root[0]);
177     for (int i = 2; i <= n; i ++ ) {
178         int x; read(x);
179         now ^= x;
180         trie.root[i] = trie.insert(now, trie.root[i - 1]);
181     }
182
183     while (m -- ) {
184         char op[2];
185         scanf("%s", op);
186         if (*op == 'A') {
187             int x; read(x); now ^= x;
188             n ++ ;
189             trie.root[n] = trie.insert(now, trie.root[n - 1]);
190         }
191         else {
192             int l, r, x;
193             read(l), read(r), read(x);

```

```

194         int tmp = now ^ x;
195         ll ans = trie.query(tmp, trie.root[l - 1], trie.root[r]);
196         printf("%lld\n", ans);
197     }
198 }
199 }

```

## 5.7 PAM

```

1 //max(len(t)*cnt(t)) t为s回文子串, cnt(t)=t出现次数
2 const int maxn = 300010;
3 struct PAM {
4     int s[maxn], now;
5     int nxt[maxn][26], fail[maxn], l[maxn], last, tot;
6     int num[maxn]; /*节点代表的所有回文串出现次数*/
7     void clear() {
8         s[0] = l[1] = -1;
9         fail[0] = tot = now = 1;
10        last = l[0] = 0;
11        ms(nxt[0], 0); ms(nxt[1], 0);
12    }
13    PAM () { clear(); }
14    int new_node (int ll) {
15        tot ++ ;
16        ms(nxt[tot], 0);
17        fail[tot] = num[tot] = 0;
18        l[tot] = ll;
19        return tot;
20    }
21    int get_fail(int x) {
22        while (s[now - l[x] - 2] != s[now - 1]) x = fail[x];
23        return x;
24    }
25    void add (int ch) {
26        s[now ++ ] = ch;
27        int cur = get_fail(last);
28        if (!nxt[cur][ch]) {
29            int tt = new_node(l[cur] + 2);
30            fail[tt] = nxt[get_fail(fail[cur])][ch];
31            nxt[cur][ch] = tt;
32        }
33        last = nxt[cur][ch]; num[last] ++ ;
34    }
35    void build () {
36        for (int i = tot; i >= 2; i -- ) {
37            num[fail[i]] += num[i];
38        }
39        num[0] = num[1] = 0;
40    }
41    void init (char *ss) {
42        while (*ss) {
43            add(*ss - 'a'); ss ++ ;
44        }
45    }
46    void init (string ss) {
47        for (auto x : ss) add(x - 'a');
48    }
49    int query();
50 };
51 int PAM::query () {
52     int res = 1;

```

```

53     for (int i = 2; i <= tot; i++) res = max(res, num[i] * l[i]);
54     return res;
55 }
56 PAM p;
57 void solve() {
58     cin >> n;
59     string s; cin >> s;
60     p.init(s); p.build();
61     printf("%lld\n", p.query());
62 }

```

## 5.8 Z

```

1 //result: ext[i] = LCP(S[i,lens],T)
2 //require: z[i] = LCP(T[i,lent],T)
3 int z[maxn], ext[maxn];
4 inline void exkmp (string s, int lens, string t, int lent, int *ext, int *z) {
5     ext[0] = 0;
6     for (int i = 1, l = 0, r = 0; i <= lens; i++) {
7         ext[i] = (i <= r) ? min(z[i - l + 1], r - i + 1) : 0;
8         while (i + ext[i] <= lens && ext[i] < lent && s[i + ext[i]] == t[ext[i] +
9             1]) ext[i]++;
10        if (i + ext[i] - 1 >= r && i != 1) l = i, r = i + ext[i] - 1;
11    }
12 }
13 void solve() {
14     string a, b; cin >> a >> b;
15     a = " " + a, b = " " + b;
16     exkmp(b, sz(b) - 1, b, sz(b) - 1, z, z);
17     exkmp(a, sz(a) - 1, b, sz(b) - 1, ext, z);
18 }

```

## 5.9 序列自动机

$nxt[i][j]$ 表示 $i$ 后的第一个字符 $j$ 的位置

```

1 vector<array<int, 27>> nxt(n + 2, {0});
2 for (int j = 1; j <= 26; j++) {
3     nxt[n][j] = nxt[n + 1][j] = n + 1;
4 }
5 for (int i = n - 1; i >= 0; i--) {
6     for (int j = 1; j <= 26; j++) {
7         nxt[i][j] = nxt[i + 1][j];
8     }
9     nxt[i][s[i + 1] - 'a' + 1] = i + 1;
10 }

```

## 6 杂项

### 6.1 高精度

`__int128`使用 (kuangbin模板) (范围:  $-2^{127} \sim 2^{127}$ , 约 $10^{38}$ , `longlong`范围 $10^{19}$ )

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 inline __int128 read(){

```

```

4     __int128 x = 0, f = 1;
5     char ch = getchar();
6     while(ch < '0' || ch > '9'){
7         if(ch == '-') f = -1;
8         ch = getchar();
9     }
10    while(ch >= '0' && ch <= '9'){
11        x = x * 10 + ch - '0';
12        ch = getchar();
13    }
14    return x * f;
15 }
16 inline void print(__int128 x){
17     if(x < 0){
18         putchar('-'); x = -x;
19     }
20     if(x > 9) print(x / 10);
21     putchar(x % 10 + '0');
22 }
23 int main(void){
24     __int128 a = read();
25     __int128 b = read();
26     print(a + b);
27     cout << endl;
28     return 0;
29 }

```

## 大整数类（压9位）

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  typedef long long ll;
4  struct Wint:vector<ll>
5  {
6      const static ll BIT=1e9;
7      Wint(ll n=0) {push_back(n);check();}
8      Wint& operator=(const char* num)
9      {
10         int Len=strlen(num)-1; clear();
11         for(int i=Len;i>=0;i-=9)
12         {
13             push_back(0); ll w=1;
14             for(int j=i;j>i-9&&j>=0;--j)
15                 back()+=(num[j]^48)*w,w*=10;
16         }
17         return *this;
18     }
19     Wint& check()
20     {
21         while(!empty()&&!back()) pop_back();
22         if(empty()) return *this;
23         for(int i=1;i<size();++i)
24             (*this)[i]+=(*this)[i-1]/BIT,
25             (*this)[i-1]%=BIT;
26         while(back()>=BIT)
27         {
28             push_back(back()/BIT);
29             (*this)[size()-2]%=BIT;
30         }
31         return *this;

```

```

32     }
33 };
34 bool operator<(Wint a,wint b)
35 {
36     if(a.size()!=b.size()) return a.size()<b.size();
37     for(int i=a.size()-1;i>=0;--i)
38         if(a[i]!=b[i]) return a[i]<b[i];
39     return 0;
40 }
41 bool operator>(Wint a,wint b) {return b<a;}
42 bool operator<=(wint a,wint b) {return !(a>b);}
43 bool operator>=(wint a,wint b) {return !(a<b);}
44 bool operator!=(wint a,wint b) {return a<b||b<a;}
45 bool operator==(wint a,wint b) {return !(a<b)&&!(b<a);}
46 Wint& operator+=(Wint &a,Wint b)
47 {
48     if(a.size()<b.size()) a.resize(b.size());
49     for(int i=0;i<b.size();++i) a[i]+=b[i];
50     return a.check();
51 }
52 Wint operator+(Wint a,Wint b) {return a+=b;}
53 Wint& operator-=(Wint &a,Wint b)
54 {
55     for(int i=0;i<b.size();a[i]-=b[i],++i)
56         if(a[i]<b[i])
57         {
58             int j=i+1;
59             while(!a[j]) ++j;
60             while(j>i) --a[j],a[--j]+=Wint::BIT;
61         }
62     return a.check();
63 }
64 Wint operator-(Wint a,Wint b) {return a-=b;}
65 Wint operator*(Wint a,Wint b)
66 {
67     if(a.empty()&&b.empty()) return a;
68     Wint n; n.assign(a.size()+b.size()-1,0);
69     for(int i=0;i<a.size();++i)
70         for(int j=0;j<b.size();++j)
71             n[i+j]+=a[i]*b[j];
72     return n.check();
73 }
74 Wint& operator*=(Wint &a,Wint b) {return a=a*b;}
75 Wint operator/(Wint a,int b)
76 {
77     Wint n; bool wp=0; ll t=0;
78     for(int i=a.size()-1;i>=0;--i)
79     {
80         t=t*Wint::BIT+a[i];
81         if(wp||t/b) wp=1,n.push_back(t/b);
82         t%=b;
83     }
84     reverse(n.begin(),n.end());
85     return n;
86 }
87 Wint& operator/=(Wint &a,int b) {return a=a/b;}
88 void readX(Wint &n) {char s[100010]; scanf("%s",s); n=s;}
89 void writeX(Wint n)
90 {
91     if(n.empty()) {putchar('0'); return;}
92     int Len=n.size()-1; printf("%11d",n[Len]);
93     for(int i=Len-1;i>=0;--i) printf("%0911d",n[i]);

```

## 6.2 二维前缀和

```

1 //构造前缀和矩阵
2 a[i][j] += a[i][j-1] + a[i-1][j] - a[i-1][j-1];
3
4 //求x1, y1, x2, y2为边界的子矩阵之和
5 a[x2][y2] - a[x1-1][y2] - a[x2][y1-1] + a[x1-1][y1-1];

```

## 6.3 双指针

## 6.4 ST表

```

1 vector<vector<int>> dp(32, vector<int> (n + 1));
2 vector<int> logn(n + 1);
3 logn[1] = 0, logn[2] = 1;
4 for (int i = 3; i <= n; i++) {
5     logn[i] = logn[i / 2] + 1;
6 }
7 for (int i = 0; i <= 25; i++) {
8     for (int j = 1; j + (1 << i) - 1 <= n; j++) {
9         if (!i) dp[i][j] = a[j];
10        else dp[i][j] = max(dp[i - 1][j], dp[i - 1][j + (1 << i - 1)]);
11    }
12 }
13 auto qry = [&] (int l, int r) {
14     int k = logn[r - l + 1];
15     return max(dp[k][l], dp[k][r - (1 << k) + 1]);
16 };

```

### 二维ST表

```

1 int val[N][N];
2 int f[N][N][15][15];
3 int n, m, k;
4
5 void prework () {
6     for (int i = 0; i < 15; i++) {
7         for (int j = 0; j < 15; j++) {
8             if (i == 0 and j == 0) continue;
9             for (int k = 1; k <= n - (1 << i) + 1; k++) {
10                for (int p = 1; p <= m - (1 << j) + 1; p++) {
11                    if (i == 0)
12                        f[k][p][i][j] = std::max (f[k][p][i][j - 1], f[k][p + (1 << j - 1)][i][j - 1]);
13                    else
14                        f[k][p][i][j] = std::max (f[k][p][i - 1][j], f[k + (1 << i - 1)][p][i - 1][j]);
15                }
16            }
17        }
18    }
19 }
20
21 int query (int r1, int c1, int r2, int c2) {
22     int k1 = log2 (r2 - r1 + 1);
23     int k2 = log2 (c2 - c1 + 1);

```



```

24     return std::min (f[r1][c1][k1][k2], std::max (f[r2 - (1 << k1) + 1][c1][k1][k2],
std::min (f[r1][c2 - (1 << k2) + 1][k1][k2], f[r2 - (1 << k1) + 1][c2 - (1 << k2) + 1]
[k1][k2])));
25 }

```

## 6.5 快读

```

1  // 仅支持整型
2  template <typename _T>
3  inline void read(_T &f) {
4      f = 0; _T fu = 1; char c = getchar();
5      while (c < '0' || c > '9') { if (c == '-') { fu = -1; } c = getchar(); }
6      while (c >= '0' && c <= '9') { f = (f << 3) + (f << 1) + (c & 15); c = getchar(); }
7      f *= fu;
8  }
9
10 //重载流输入型
11 struct IOS{
12     template<typename ATP>IOS& operator >> (ATP &x){
13         x = 0; int f = 1; char c;
14         for(c = getchar(); c < '0' || c > '9'; c = getchar()) if(c == '-') f = -1;
15         while(c >= '0' && c <= '9') x = x * 10 + (c ^ '0'), c = getchar();
16         x *= f;
17         return *this;
18     }
19 }io;
20 io >> n;
21

```

## 6.6 模拟退火

```

1  /*
2  求费马点型
3
4  在二维平面上有 n 个点，第 i 个点的坐标为 (xi,yi)。
5
6  请你找出一个点，使得该点到这 n 个点的距离之和最小。
7
8  该点可以选择在平面中的任意位置，甚至与这 n 个点的位置重合。
9  */
10
11 #define x first
12 #define y second
13 typedef pair<double, double> PII;
14 const int maxn = 110;
15 int n;
16 PII q[maxn];
17 double ans = 1e8;
18
19 double rand (double l, double r) {
20     return (double)rand() / RAND_MAX * (r - l) + l;
21 }
22
23 double dis (PII a, PII b) {
24     return sqrt((a.x - b.x) * (a.x - b.x) + (a.y - b.y) * (a.y - b.y));
25 }
26

```

```

27 double calc (PII p) {
28     double res = 0;
29     for (int i = 1; i <= n; i ++ ) {
30         res += dis(p, q[i]);
31     }
32     ans = min(ans, res);
33     return res;
34 }
35
36 void simulate_anneal () {
37     PII cur(rand(0, 10000), rand(0, 10000));
38     for (double t = 1e4; t > 1e-4; t *= 0.9) {
39         PII np(rand(cur.x - t, cur.x + t), rand(cur.y - t, cur.y + t));
40         double d = calc(np) - calc(cur);
41         if (exp(-d / t) > rand(0, 1)) cur = np;
42     }
43 }
44
45 int main () {
46     srand(time(NULL));
47     scanf("%d", &n);
48     for (int i = 1; i <= n; i ++ ) scanf("%lf%lf", &q[i].first, &q[i].second);
49
50     // 或 while ((double)clock()/CLOCKS_PER_SEC<0.8)
51     for (int i = 1; i <= 100; i ++ )
52         simulate_anneal();
53
54     printf("%.01f\n", ans);
55 }

```

## 6.7 STL

```

1  vector, 变长数组, 倍增的思想
2      size()  返回元素个数
3      empty()  返回是否为空
4      clear()  清空
5      front()/back()
6      push_back()/pop_back()
7      begin()/end()
8      []
9      支持比较运算, 按字典序
10
11  pair<int, int>
12      first, 第一个元素
13      second, 第二个元素
14      支持比较运算, 以first为第一关键字, 以second为第二关键字 (字典序)
15
16  string, 字符串
17      int has = s.find('xxx');
18      if (has != string::npos) 含xxx字符
19      size()/length()  返回字符串长度
20      empty()
21      clear()
22      substr(起始下标, (子串长度))  返回子串
23      c_str()  返回字符串所在字符数组的起始地址 代表可以用printf输出string类
24
25  queue, 队列
26      size()
27      empty()
28      push()  向队尾插入一个元素

```

```

29     front()    返回队头元素
30     back()    返回队尾元素
31     pop()     弹出队头元素
32
33     priority_queue, 优先队列, 默认是大根堆
34     push()    插入一个元素
35     top()     返回堆顶元素
36     pop()     弹出堆顶元素
37     定义成小根堆的方式: priority_queue<int, vector<int>, greater<int>> q;
38
39     stack, 栈
40     size()
41     empty()
42     push()    向栈顶插入一个元素
43     top()     返回栈顶元素
44     pop()     弹出栈顶元素
45
46     deque, 双端队列
47     size()
48     empty()
49     clear()
50     front()/back()
51     push_back()/pop_back()
52     push_front()/pop_front()
53     begin()/end()
54     []
55
56     set, map, multiset, multimap, 基于平衡二叉树(红黑树), 动态维护有序序列
57     size()
58     empty()
59     clear()
60     begin()/end()
61     ++, -- 返回前驱和后继, 时间复杂度  $O(\log n)$ 
62
63     set/multiset
64     insert()  插入一个数
65     find()    查找一个数
66     count()   返回某一个数的个数
67     erase()
68         (1) 输入是一个数x, 删除所有x  $O(k + \log n)$ 
69         (2) 输入一个迭代器, 删除这个迭代器
70     lower_bound()/upper_bound()
71         lower_bound(x)  返回大于等于x的最小的数的迭代器
72         upper_bound(x)  返回大于x的最小的数的迭代器
73     map/multimap
74     insert()  插入的数是一个pair
75     erase()   输入的参数是pair或者迭代器
76     find()
77     []       时间复杂度是  $O(\log n)$ 
78     lower_bound()/upper_bound()
79
80     unordered_set, unordered_map, unordered_multiset, unordered_multimap, 哈希表
81     和上面类似, 增删改查的时间复杂度是  $O(1)$ 
82     不支持 lower_bound()/upper_bound(), 迭代器的++, --
83
84
85
86     bit.size()    返回大小(位数)
87     bit.count()   返回1的个数
88     bit.any()     返回是否有1
89     bit.none()    返回是否没有1
90     bit.set()     全都变成1

```

```

91 bit.set(p)      将第p + 1位变成1 (bitset是从第0位开始的!)
92 bit.set(p, x)   将第p + 1位变成x
93 bit.reset()     全都变成0
94 bit.reset(p)    将第p + 1位变成0
95 bit.flip()      全都取反
96 bit.flip(p)     将第p + 1位取反
97 bit.to_ulong()  返回它转换为unsigned long的结果, 如果超出范围则报错
98 bit.to_ullong() 返回它转换为unsigned long long的结果, 如果超出范围则报错
99 bit.to_string() 返回它转换为string的结果

```

## 6.8 离散化

```

1  vector<int> v;
2      for (int i = 1; i <= n; i ++ ) {
3          read(q[i]); v.pb(q[i]);
4      }
5      sort(all(v));
6      v.erase(unique(all(v)), v.end());
7      for (int i = 1; i <= n; i ++ ) {
8          q[i] = lower_bound(all(v), q[i]) - v.begin() + 1;
9      }

```

## 6.9 随机数

```

1  mt19937 sed(time(nullptr));
2  uniform_int_distribution<int> range(1, r); // 1到r内随机数
3  cout << range(sed) << endl;

```

## 6.10 ModInt

### Z

不要引入std

```

1  constexpr int P = 998244353;
2  using i64 = long long;
3  // assume -P <= x < 2P
4  int norm(int x) {
5      if (x < 0) {
6          x += P;
7      }
8      if (x >= P) {
9          x -= P;
10     }
11     return x;
12 }
13 template<class T>
14 T power(T a, i64 b) {
15     T res = 1;
16     for (; b; b /= 2, a *= a) {
17         if (b % 2) {
18             res *= a;
19         }
20     }
21     return res;
22 }

```

```

23 struct Z {
24     int x;
25     Z(int x = 0) : x(norm(x)) {}
26     Z(i64 x) : x(norm(x % P)) {}
27     int val() const {
28         return x;
29     }
30     Z operator-() const {
31         return Z(norm(P - x));
32     }
33     Z inv() const {
34         assert(x != 0);
35         return power(*this, P - 2);
36     }
37     Z &operator*=(const Z &rhs) {
38         x = i64(x) * rhs.x % P;
39         return *this;
40     }
41     Z &operator+=(const Z &rhs) {
42         x = norm(x + rhs.x);
43         return *this;
44     }
45     Z &operator-=(const Z &rhs) {
46         x = norm(x - rhs.x);
47         return *this;
48     }
49     Z &operator/=(const Z &rhs) {
50         return *this *= rhs.inv();
51     }
52     friend Z operator*(const Z &lhs, const Z &rhs) {
53         Z res = lhs;
54         res *= rhs;
55         return res;
56     }
57     friend Z operator+(const Z &lhs, const Z &rhs) {
58         Z res = lhs;
59         res += rhs;
60         return res;
61     }
62     friend Z operator-(const Z &lhs, const Z &rhs) {
63         Z res = lhs;
64         res -= rhs;
65         return res;
66     }
67     friend Z operator/(const Z &lhs, const Z &rhs) {
68         Z res = lhs;
69         res /= rhs;
70         return res;
71     }
72     friend std::istream &operator>>(std::istream &is, Z &a) {
73         i64 v;
74         is >> v;
75         a = Z(v);
76         return is;
77     }
78     friend std::ostream &operator<<(std::ostream &os, const Z &a) {
79         return os << a.val();
80     }
81 };
82
83

```

## mint

```
1 constexpr int P = 998244353;
2 template<const int T>
3 struct ModInt {
4     const static int mod = T;
5     int x;
6     ModInt (int x = 0) : x (x% mod) {}
7     int val () { return x; }
8     constexpr int moded (int x) const {
9         if (x < 0) {
10             x += P;
11         }
12         if (x >= P) {
13             x -= P;
14         }
15         return x;
16     }
17     ModInt operator + (const ModInt& a) const { int x0 = x; x0 += a.x; return ModInt
(x0); }
18     ModInt operator - (const ModInt& a) const { int x0 = x - a.x; return ModInt (x0 < mod
? x0 + mod : x0); }
19     ModInt operator * (const ModInt& a) const { return ModInt (1LL * x * a.x % mod); }
20     ModInt operator / (const ModInt& a) const { return *this * a.inv (); }
21     void operator += (const ModInt& a) { x = moded (x + a.x); if (x >= mod) x -= mod; }
22     void operator -= (const ModInt& a) { x -= a.x; if (x < 0) x += mod; }
23     void operator *= (const ModInt& a) { x = 1LL * x * a.x % mod; }
24     void operator /= (const ModInt& a) { *this = *this / a; }
25     friend ostream& operator<<(ostream& os, const ModInt& a) { return os << a.x; }
26     ModInt pow (int64_t n) const {
27         ModInt res (1), mul (x);
28         while (n) {
29             if (n & 1) res *= mul;
30             mul *= mul;
31             n >>= 1;
32         }
33         return res;
34     }
35     ModInt inv () const {
36         int a = x, b = mod, u = 1, v = 0;
37         while (b) {
38             int t = a / b;
39             a -= t * b; swap (a, b);
40             u -= t * v; swap (u, v);
41         }
42         if (u < 0) u += mod;
43         return u;
44     }
45 };
46 typedef ModInt<P> mint; // mod1 || mod2
```

## 6.11 重载哈希

```
1 struct custom_hash {
2     static uint64_t splitmix64(uint64_t x) {
3         // http://xorshift.di.unimi.it/splitmix64.c
4         x += 0x9e3779b97f4a7c15;
5         x = (x ^ (x >> 30)) * 0xbf58476d1ce4e5b9;
6         x = (x ^ (x >> 27)) * 0x94d049bb133111eb;
```

```

7         return x ^ (x >> 31);
8     }
9
10    size_t operator()(uint64_t x) const {
11        static const uint64_t FIXED_RANDOM =
12        chrono::steady_clock::now().time_since_epoch().count();
13        return splitmix64(x + FIXED_RANDOM);
14    }
15 };
16 unordered_map<LL, LL, custom_hash> mp;

```

## 6.12 取模

```

1 int mol(int x) {
2     if (x < 0) return x + P;
3     if (x >= P) return x - P;
4     return x;
5 }

```

```

1 #define mol(x) ((x) < 0 ? (x) + P : (x) >= P ? (x) - P : (x))

```

## 6.13 枚举子集

```

1 for (int i = 0; i < (1 << n); i++) {
2     for (int j = i; j; j = (j - 1) & i) {
3         cout << i << ' ' << j << '\n';
4     }
5 }

```