目录

基础

枚举 折半枚举 搜索 模拟 打表 公式

二分 尺取 离散化

构造 贪心

快速计算斐波那契数列

// TODO: add xiaoyu's code

数据结构

分数

Numerator 分子 Denominator 分母

```
struct Fraction
{
    int num, den;
    Fraction(int num = 0, int den = 1) {
        if (den < 0) {
            num = -num, den = -den;
        }
        assert(den != 0);
        int g = gcd(abs(num), den);
        num /= g, den /= g;
        this->num = num, this->den = den;
    }
};
```

高精度整数

// 正在整理

线段树

// 正在整理

树状数组

// 正在整理

左偏树

```
// 注意编号为0的节点表示空节点
struct LeftTree
{
    const static int MXN = 100100;
    int tot = 0;
    int l[MXN], r[MXN], v[MXN], d[MXN];

    // 初始化值为x的元素
    int init(int x)
    {
        tot++;
        v[tot] = x;
        l[tot] = r[tot] = d[tot] = 0;
        return tot;
    }

    // 合并堆顶编号为x, y的堆
    int merge(int x, int y)
    {
        if (!x) return y;
        if (!y) return x;
        if (v[x] < v[y])
            swap(x, y);
        r[x] = merge(r[x], y);
        if (d[l[x]] < d[r[x]])
            swap(l[x], r[x]);
        d[x] = d[r[x]] + 1;
        return x;
    }

    // 向堆顶编号为x的堆中插入值为v的元素
    int insert(int x, int v)
    {
        return merge(x, init(v));
    }

    // 取编号为x的堆的堆顶元素
    int top(int x)
    {
        return v[x];
    }

    // 弹出编号为x的堆的堆顶元素，返回新堆顶的编号
    int pop(int x)
    {
        return merge(l[x], r[x]);
    }
} t;
```

图论

黑科技

IO 优化

```
template<typename T = int>
```

```
inline T read() {
     T val = 0, sign = 1; char ch;
     for (ch = getchar(); ch < '0' || ch > '9'; ch =
getchar())
          if (ch == '-') sign = -1;
     for (; ch >= '0' && ch <= '9'; ch = getchar())
          val = val * 10 + ch - '0';
     return sign * val;
}
```

数论

GCD/LCM

拓展欧几里得

```
LL gcd(LL a, LL b, LL &x, LL &y)
{
     if (b == 0) {
          x = 1, y = 0;
          return a;
     }
     else {
          LL r = gcd(b, a%b, y, x);
          y -= (a/b)*x;
          return r;
     }
}
```

单变元模线性方程组

```
vector<LL> line_mod_equation(LL a, LL b, LL n)
{
     LL x, y;
     LL d = gcd(a, n, x, y);

     vector<LL> ans;
     if (b%d == 0) {
          x %= n; x += n; x %= n;
          ans.push_back(x*(b/d)%(n/d));
          for (LL i=1; i<d; ++i)
               ans.push_back((ans[0]+i*(n/d))%n);
     }
     return ans;
}
```