# Platelet

Team Reference Material

凌皓煜 陈彤 顾逸

2018

# Contents

# Chapter 1

# Graph Theory

## 1.1 2-SAT

## 1.2 双连通分量

### 1.2.1 点双连通分量

### 1.2.2 边双连通分量

## 1.3 K 短路 (lhy)

```
const int MAXNODE = MAXN + MAXM * 2;

bool used[MAXN];
int n, m, cnt, S, T, Kth, N, TT;
int rt[MAXN], seq[MAXN], adj[MAXN], from[MAXN], dep[MAXN];
LL dist[MAXN], w[MAXM], ans[MAXK];

struct GivenEdge{
    int u, v, w;
    GivenEdge() {};
    GivenEdge(int _u, int _v, int _w) : u(_u), v(_v), w(_w){};
}edge[MAXM];

struct Edge{
    int v, nxt, w;
    Edge() {};
    Edge(int _v, int _nxt, int _w) : v(_v), nxt(_nxt), w(_w) {};
}e[MAXM];

inline void addedge(int u, int v, int w)
{
    e[++cnt] = Edge(v, adj[u], w); adj[u] = cnt;
}

void dij(int S)
{
    for(int i = 1; i <= N; i++)
    {
        dist[i] = INF;
        dep[i] = 0x3f3f3f3f;
        used[i] = false;
        from[i] = 0;
    }
```

```
    static priority_queue<pair<LL, int>, vector<pair<LL, int> >, greater<pair<LL, int> > > hp;
    while(!hp.empty())hp.pop();
    hp.push(make_pair(dist[S] = 0, S));
    dep[S] = 1;
    while(!hp.empty())
    {
        pair<LL, int> now = hp.top();
        hp.pop();
        int u = now.second;
        if(used[u])continue;
        else used[u] = true;
        for(int p = adj[u]; p; p = e[p].nxt)
        {
            int v = e[p].v;
            if(dist[u] + e[p].w < dist[v])
            {
                dist[v] = dist[u] + e[p].w;
                dep[v] = dep[u] + 1;
                from[v] = p;
                hp.push(make_pair(dist[v], v));
            }
        }
    }
    for(int i = 1; i <= m; i++)    w[i] = 0;
    for(int i = 1; i <= N; i++)
        if(from[i])w[from[i]] = -1;
    for(int i = 1; i <= m; i++)
    {
        if(~w[i] && dist[edge[i].u] < INF && dist[edge[i].v] < INF)
        {
            w[i] = -dist[edge[i].u] + (dist[edge[i].v] + edge[i].w);
        }
        else
        {
            w[i] = -1;
        }
    }
}

inline bool cmp_dep(int p, int q)
{
    return dep[p] < dep[q];
}

struct Heap{
    LL key;
    int id, lc, rc, dist;
    Heap() {};
    Heap(LL k, int i, int l, int r, int d) : key(k), id(i), lc(l), rc(r), dist(d) {};
    inline void clear()
    {
        key = 0;
        id = lc = rc = dist = 0;
    }
}hp[MAXNODE];

inline int merge_simple(int u, int v)
{
    if(!u)return v;
    if(!v)return u;
    if(hp[u].key > hp[v].key)
```

```
87      {
88          swap(u, v);
89      }
90      hp[u].rc = merge_simple(hp[u].rc, v);
91      if(hp[hp[u].lc].dist < hp[hp[u].rc].dist)
92      {
93          swap(hp[u].lc, hp[u].rc);
94      }
95      hp[u].dist = hp[hp[u].rc].dist + 1;
96      return u;
97  }

98  inline int merge_full(int u, int v)
99  {
100     if(!u)return v;
101     if(!v)return u;
102     if(hp[u].key > hp[v].key)
103     {
104         swap(u, v);
105     }
106     int nownode = ++cnt;
107     hp[nownode] = hp[u];
108     hp[nownode].rc = merge_full(hp[nownode].rc, v);
109     if(hp[hp[nownode].lc].dist < hp[hp[nownode].rc].dist)
110     {
111         swap(hp[nownode].lc, hp[nownode].rc);
112     }
113     hp[nownode].dist = hp[hp[nownode].rc].dist + 1;
114     return nownode;
115 }

116 priority_queue<pair<LL, int>, vector<pair<LL, int> >, greater<pair<LL, int> > > Q;

117 int main()
118 {
119     while(scanf("%d%d", &n, &m) != EOF)
120     {
121         scanf("%d%d%d%d", &S, &T, &Kth, &TT);
122         for(int i = 1; i <= m; i++)
123         {
124             int u, v, w;
125             scanf("%d%d%d", &u, &v, &w);
126             edge[i] = {u, v, w};
127         }
128         N = n;
129         memset(adj, 0, sizeof(*adj) * (N + 1));
130         cnt = 0;
131         for(int i = 1; i <= m; i++)
132             addedge(edge[i].v, edge[i].u, edge[i].w);
133         dij(T);
134         if(dist[S] > TT)
135         {
136             puts("Whitesnake!");
137             continue;
138         }
139         for(int i = 1; i <= N; i++)
140             seq[i] = i;
141         sort(seq + 1, seq + N + 1, cmp_dep);

142         cnt = 0;
143         memset(adj, 0, sizeof(*adj) * (N + 1));
```

```
144         memset(rt, 0, sizeof(*rt) * (N + 1));
145         for(int i = 1; i <= m; i++)
146             addedge(edge[i].u, edge[i].v, edge[i].w);
147         rt[T] = cnt = 0;
148         hp[0].dist = -1;
149         for(int i = 1; i <= N; i++)
150         {
151             int u = seq[i], v = edge[from[u]].v;
152             rt[u] = 0;
153             for(int p = adj[u]; p; p = e[p].nxt)
154             {
155                 if(~w[p])
156                 {
157                     hp[++cnt] = Heap(w[p], p, 0, 0, 0);
158                     rt[u] = merge_simple(rt[u], cnt);
159                 }
160             }
161             if(i == 1)continue;
162             rt[u] = merge_full(rt[u], rt[v]);
163         }
164         while(!Q.empty())Q.pop();
165         Q.push(make_pair(dist[S], 0));
166         edge[0].v = S;
167         for(int kth = 1; kth <= Kth; kth++)
168         {
169             if(Q.empty())
170             {
171                 ans[kth] = -1;
172                 continue;
173             }
174             pair<LL, int> now = Q.top(); Q.pop();
175             ans[kth] = now.first;
176             int p = now.second;
177             if(hp[p].lc)
178             {
179                 Q.push(make_pair(+hp[hp[p].lc].key + now.first - hp[p].key, hp[p].lc));
180             }
181             if(hp[p].rc)
182             {
183                 Q.push(make_pair(+hp[hp[p].rc].key + now.first - hp[p].key, hp[p].rc));
184             }
185             if(rt[edge[hp[p].id].v])
186             {
187                 Q.push(make_pair(hp[rt[edge[hp[p].id].v]].key + now.first, rt[edge[hp[p].id].v]));
188             }
189         }
190         if(ans[Kth] == -1 || ans[Kth] > TT)
191         {
192             puts("Whitesnake!");
193         }
194         else
195         {
196             puts("yareyaredawa");
197         }
198     }
199 }
```

## 1.4 最大团

## 1.5 一般图最大匹配

## 1.6 树

### 1.6.1 虚树

### 1.6.2 矩阵树定理

### 1.6.3 点分治

### 1.6.4 Prufer 编码

### 1.6.5 Link-Cut Tree (ct)

```
struct Node *null;
struct Node {
    Node *ch[2], *fa, *pos;
    int val, mn, l, len; bool rev;
    // min_val in chain
    inline bool type()
    {
        return fa -> ch[1] == this;
    }
    inline bool check()
    {
        return fa -> ch[type()] == this;
    }
    inline void pushup()
    {
        pos = this; mn = val;
        ch[0] -> mn < mn ? mn = ch[0] -> mn, pos = ch[0] -> pos : 0;
        ch[1] -> mn < mn ? mn = ch[1] -> mn, pos = ch[1] -> pos : 0;
        len = ch[0] -> len + ch[1] -> len + l;
    }
    inline void pushdown()
    {
        if (rev)
        {
            ch[0] -> rev ^= 1;
            ch[1] -> rev ^= 1;
            std::swap(ch[0], ch[1]);
            rev ^= 1;
        }
    }
    inline void pushdownall()
    {
        if (check()) fa -> pushdownall();
        pushdown();
    }
    inline void rotate()
    {
        bool d = type(); Node *f = fa, *gf = f -> fa;
        (fa = gf, f -> check()) ? fa -> ch[f -> type()] = this : 0;
        (f -> ch[d] = ch[!d]) != null ? ch[!d] -> fa = f : 0;
        (ch[!d] = f) -> fa = this;
        f -> pushup();
    }
    inline void splay(bool need = 1)
```

```
45      {
46          if (need) pushdownall();
47          for (; check(); rotate())
48              if (fa -> check())
49                  (type() == fa -> type() ? fa : this) -> rotate();
50          pushup();
51      }
52      inline Node *access()
53      {
54          Node *i = this, *j = null;
55          for (; i != null; i = (j = i) -> fa)
56          {
57              i -> splay();
58              i -> ch[1] = j;
59              i -> pushup();
60          }
61          return j;
62      }
63      inline void make_root()
64      {
65          access();
66          splay();
67          rev ^= 1;
68      }
69      inline void link(Node *that)
70      {
71          make_root();
72          fa = that;
73          splay(0);
74      }
75      inline void cut(Node *that)
76      {
77          make_root();
78          that -> access();
79          that -> splay(0);
80          that -> ch[0] = fa = null;
81          that -> pushup();
82      }
83  } mem[maxn];
84  inline Node *query(Node *a, Node *b)
85  {
86      a -> make_root(); b -> access(); b -> splay(0);
87      return b -> pos;
88  }
89  inline int dist(Node *a, Node *b)
90  {
91      a -> make_root(); b -> access(); b -> splay(0);
92      return b -> len;
93  }
```

# Chapter 2

# Math

## 2.1 int64 相乘取模 (Durandal)

```cpp
int64_t mul(int64_t x, int64_t y, int64_t p) {
    int64_t t = (x * y - (int64_t) ((long double) x / p * y + 1e-3) * p) % p;
    return t < 0 ? t + p : t;
}
```

## 2.2 扩展欧几里得 (gy)

```cpp
// return gcd(a, b)
// ax+by=gcd(a,b)
int extend_gcd(int a, int b, int &x, int &y) {
    if (b == 0) {
        x = 1, y = 0;
        return a;
    }
    int res = extend_gcd(b, a % b, x, y);
    int t = y;
    y = x - a / b * y;
    x = t;
    return res;
}

// return minimal positive integer x so that ax+by=c
// or -1 if such x does not exist
int solve_equ(int a, int b, int c) {
    int x, y, d;
    d = extend_gcd(a, b, x, y);
    if (c % d)
        return -1;
    int t = c / d;
    x *= t;
    y *= t;
    int k = b / d;
    x = (x % k + k) % k;
    return x;
}

// return minimal positive integer x so that ax==b(mod p)
// or -1 if such x does not exist
int solve(int a, int b, int p) {
    a = (a % p + p) % p;
    b = (b % p + p) % p;
```

```
    return solve_equ(a, p, b);
}
```

## 2.3 中国剩余定理 (Durandal)

返回是否可行，余数和模数结果为 $r_1, m_1$

```
bool CRT(int &r1, int &m1, int r2, int m2) {
    int x, y, g = extend_gcd(m1, m2, x, y);
    if ((r2 - r1) % g != 0) return false;
    x = 1ll * (r2 - r1) * x % m2;
    if (x < 0) x += m2;
    x /= g;
    r1 += m1 * x;
    m1 *= m2 / g;
    return true;
}
```

## 2.4 线性同余不等式 (Durandal)

必须满足 $0 \le d < m, 0 \le l \le r < m$，返回 $\min\{x \ge 0 \mid l \le x \cdot d \bmod m \le r\}$，无解返回 $-1$

```
int64_t calc(int64_t d, int64_t m, int64_t l, int64_t r) {
    if (l == 0) return 0;
    if (d == 0) return -1;
    if (d * 2 > m) return calc(m - d, m, m - r, m - l);
    if ((l - 1) / d < r / d) return (l - 1) / d + 1;
    int64_t k = calc((-m % d + d) % d, d, l % d, r % d);
    if (k == -1) return -1;
    return (k * m + l - 1) / d + 1;
}
```

## 2.5 组合数

## 2.6 高斯消元

## 2.7 Miller Rabin & Pollard Rho (gy)

```
/*
 * In Java, use BigInteger.isProbablePrime(int certainty) to replace miller_rabin(BigInteger
  ↪ number)
 * Test Set / First Wrong Answer
 * 2 / 2,047
 * 2, 3 / 1,373,653
 * 31, 73 / 9,080,191
 * 2, 3, 5 / 25,326,001
 * 2, 3, 5, 7 / 3,215,031,751 (> Int.MAX_VALUE)
 * 2, 7, 61 / 4,759,123,141
 * 2, 13, 23, 1662803 / 1,122,004,669,633
 * 2, 3, 5, 7, 11 / 2,152,302,898,747
 * 2, 3, 5, 7, 11, 13 / 3,474,749,660,383
 * 2, 3, 5, 7, 11, 13, 17 / 341,550,071,728,321
 * 2, 3, 5, 7, 11, 13, 17, 19, 23 / 3,825,123,056,546,413,051
 * 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37 / 318,665,857,834,031,151,167,461 (> Long.MAX_VALUE)
 * 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41 / 3,317,044,064,679,887,385,961,981
```

```cpp
 */
const int test_case_size = 12;
const int test_cases[test_case_size] = {2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37};

int64_t multiply_mod(int64_t x, int64_t y, int64_t p) {
    int64_t t = (x * y - (int64_t) ((long double) x / p * y + 1e-3) * p) % p;
    return t < 0 ? t + p : t;
}

int64_t add_mod(int64_t x, int64_t y, int64_t p) {
    return (0ull + x + y) % p;
}

int64_t power_mod(int64_t x, int64_t exp, int64_t p) {
    int64_t ans = 1;
    while (exp) {
        if (exp & 1)
            ans = multiply_mod(ans, x, p);
        x = multiply_mod(x, x, p);
        exp >>= 1;
    }
    return ans;
}

bool miller_rabin_check(int64_t prime, int64_t base) {
    int64_t number = prime - 1;
    for (; ~number & 1; number >>= 1)
        continue;
    int64_t result = power_mod(base, number, prime);
    for (; number != prime - 1 && result != 1 && result != prime - 1; number <<= 1)
        result = multiply_mod(result, result, prime);
    return result == prime - 1 || (number & 1) == 1;
}

bool miller_rabin(int64_t number) {
    if (number < 2)
        return false;
    if (number < 4)
        return true;
    if (~number & 1)
        return false;
    for (int i = 0; i < test_case_size && test_cases[i] < number; i++)
        if (!miller_rabin_check(number, test_cases[i]))
            return false;
    return true;
}

int64_t gcd(int64_t x, int64_t y) {
    return y == 0 ? x : gcd(y, x % y);
}

int64_t pollard_rho_test(int64_t number, int64_t seed) {
    int64_t x = rand() % (number - 1) + 1, y = x;
    int head = 1, tail = 2;
    while (true) {
        x = multiply_mod(x, x, number);
        x = add_mod(x, seed, number);
        if (x == y)
            return number;
        int64_t answer = gcd(std::abs(x - y), number);
        if (answer > 1 && answer < number)
```

```
            return answer;
        if (++head == tail) {
            y = x;
            tail <<= 1;
        }
    }
}

void factorize(int64_t number, std::vector<int64_t> &divisor) {
    if (number > 1) {
        if (miller_rabin(number)) {
            divisor.push_back(number);
        } else {
            int64_t factor = number;
            while (factor >= number)
                factor = pollard_rho_test(number, rand() % (number - 1) + 1);
            factorize(number / factor, divisor);
            factorize(factor, divisor);
        }
    }
}
```

## 2.8 $O(m^2 \log n)$ 线性递推

## 2.9 Polynomial

### 2.9.1 FFT

### 2.9.2 NTT & 多项式求逆

## 2.10 拉格朗日插值

## 2.11 杜教筛

## 2.12 BSGS

### 2.12.1 BSGS

### 2.12.2 扩展 BSGS

## 2.13 直线下整点个数 (gy)

必须满足 $a \geq 0, b \geq 0, m > 0$，返回 $\sum_{i=0}^{n-1} \frac{a+bi}{m}$

```
int64_t count(int64_t n, int64_t a, int64_t b, int64_t m) {
    if (b == 0)
        return n * (a / m);
    if (a >= m)
        return n * (a / m) + count(n, a % m, b, m);
    if (b >= m)
        return (n - 1) * n / 2 * (b / m) + count(n, a, b % m, m);
    return count((a + b * n) / m, (a + b * n) % m, m, b);
}
```

## 2.14 单纯形

## 2.15 辛普森积分

## 2.16 常用数列定理 (gy)

- 第一类 Stirling number
  用 $s(n,k)=(-1)^{n-k}\begin{bmatrix}n\\k\end{bmatrix}$ 表示第一类 Stirling number
  $\begin{bmatrix}n+1\\k\end{bmatrix}=n\begin{bmatrix}n\\k\end{bmatrix}+\begin{bmatrix}n\\k-1\end{bmatrix},k>0$
  $\begin{bmatrix}0\\0\end{bmatrix}=1,\begin{bmatrix}n\\0\end{bmatrix}=\begin{bmatrix}0\\n\end{bmatrix}=0,n>0$
  $\begin{bmatrix}n\\k\end{bmatrix}$ 为将 $n$ 个元素分成 $k$ 个环的方案数
- 第二类 Stirling number
  用 $S(n,k)=\begin{Bmatrix}n\\k\end{Bmatrix}$ 表示第二类 Stirling number
  $\begin{Bmatrix}n+1\\k\end{Bmatrix}=k\begin{Bmatrix}n\\k\end{Bmatrix}+\begin{Bmatrix}n\\k-1\end{Bmatrix},k>0$
  $\begin{Bmatrix}0\\0\end{Bmatrix}=1,\begin{Bmatrix}n\\0\end{Bmatrix}=\begin{Bmatrix}0\\n\end{Bmatrix}=0,n>0$
  $\begin{Bmatrix}n\\k\end{Bmatrix}=\frac{1}{k!}\sum\limits_{j=0}^{k}(-1)^{k-j}\binom{k}{j}j^n$
  $\begin{Bmatrix}n\\k\end{Bmatrix}$ 为将 $n$ 个元素划分成 $k$ 个非空集合的方案数
- Catalan Number
  $c_n$ 表示长度为 $2n$ 的合法括号序的数量
  $c_1=1,\ c_{n+1}=\sum\limits_{i=1}^{n}c_i\times c_{n+1-i}$
  $c_n=\frac{\binom{2n}{n}}{n+1}$
- Bell number
  $B_n$ 表示基数为 $n$ 的集合的划分方案数
  $B_i=\begin{cases}1 & i=0\\ \sum\limits_{k=0}^{n}\binom{n}{k}B_k & i>0\end{cases}$
  $B_n=\sum\limits_{k=0}^{n}\begin{Bmatrix}n\\k\end{Bmatrix}$
- Bernoulli number
  $\sum\limits_{j=0}^{m}\binom{m+1}{j}B_j=0,m>0$
  $B_i=\begin{cases}1 & i=0\\ -\frac{\sum\limits_{j=0}^{i-1}\binom{i+1}{j}}{i+1} & i>0\end{cases}$
  $\sum\limits_{k=1}^{n}k^m=\frac{1}{m+1}\sum\limits_{k=0}^{m}\binom{m+1}{k}B_kn^{m+1-k}$
- Lagrange polynomial
  给定次数为 $n$ 的多项式函数 $L(x)$ 上的 $n+1$ 个点 $(x_0,y_0),(x_1,y_1),\ldots,(x_n,y_n)$
  则 $L(x)=\sum\limits_{j=0}^{n}y_j\prod\limits_{0\le m\le n,m\ne j}\frac{x-x_m}{x_j-x_m}$

# Chapter 3

# Geometry

## 3.1 点、直线、圆 (gy)

```cpp
using number = long double;
const number eps = 1e-8;

number _sqrt(number x) {
    return std::sqrt(std::max(x, (number) 0));
}
number _asin(number x) {
    x = std::min(x, (number) 1), x = std::max(x, (number) -1);
    return std::asin(x);
}
number _acos(number x) {
    x = std::min(x, (number) 1), x = std::max(x, (number) -1);
    return std::acos(x);
}

int sgn(number x) {
    return (x > eps) - (x < -eps);
}
int cmp(number x, number y) {
    return sgn(x - y);
}

struct point {
    number x, y;
    point() {}
    point(number x, number y) : x(x), y(y) {}

    number len2() const {
        return x * x + y * y;
    }
    number len() const {
        return _sqrt(len2());
    }
    point unit() const {
        return point(x / len(), y / len());
    }
    point rotate90() const {
        return point(-y, x);
    }

    friend point operator+(const point &a, const point &b) {
        return point(a.x + b.x, a.y + b.y);
```

```
    }
    friend point operator-(const point &a, const point &b) {
        return point(a.x - b.x, a.y - b.y);
    }
    friend point operator*(const point &a, number b) {
        return point(a.x * b, a.y * b);
    }
    friend point operator/(const point &a, number b) {
        return point(a.x / b, a.y / b);
    }
    friend number dot(const point &a, const point &b) {
        return a.x * b.x + a.y * b.y;
    }
    friend number det(const point &a, const point &b) {
        return a.x * b.y - a.y * b.x;
    }
    friend number operator==(const point &a, const point &b) {
        return cmp(a.x, b.x) == 0 && cmp(a.y, b.y) == 0;
    }
};

number dis2(const point &a, const point &b) {
    return (a - b).len2();
}
number dis(const point &a, const point &b) {
    return (a - b).len();
}

struct line {
    point a, b;
    line() {}
    line(point a, point b) : a(a), b(b) {}
    point value() const {
        return b - a;
    }
};

bool point_on_line(const point &p, const line &l) {
    return sgn(det(p - l.a, p - l.b)) == 0;
}
// including endpoint
bool point_on_ray(const point &p, const line &l) {
    return sgn(det(p - l.a, p - l.b)) == 0 &&
        sgn(dot(p - l.a, l.b - l.a)) >= 0;
}
// including endpoints
bool point_on_seg(const point &p, const line &l) {
    return sgn(det(p - l.a, p - l.b)) == 0 &&
        sgn(dot(p - l.a, l.b - l.a)) >= 0 &&
        sgn(dot(p - l.b, l.a - l.b)) >= 0;
}
bool seg_has_intersection(const line &a, const line &b) {
    if (point_on_seg(a.a, b) || point_on_seg(a.b, b) ||
            point_on_seg(b.a, a) || point_on_seg(b.b, a))
        return /* including endpoints */ true;
    return sgn(det(a.a - b.a, b.b - b.a)) * sgn(det(a.b - b.a, b.b - b.a)) < 0
        && sgn(det(b.a - a.a, a.b - a.a)) * sgn(det(b.b - a.a, a.b - a.a)) < 0;
}
point intersect(const line &a, const line &b) {
    number s1 = det(a.b - a.a, b.a - a.a);
    number s2 = det(a.b - a.a, b.b - a.a);
```

```
96      return (b.a * s2 - b.b * s1) / (s2 - s1);
97  }
98  point projection(const point &p, const line &l) {
99      return l.a + (l.b - l.a) * dot(p - l.a, l.b - l.a) / (l.b - l.a).len2();
100 }
101 number dis(const point &p, const line &l) {
102     return std::abs(dot(p - l.a, l.b - l.a)) / (l.b - l.a).len();
103 }
104 point symmetry_point(const point &a, const point &o) {
105     return o + o - a;
106 }
107 point reflection(const point &p, const line &l) {
108     return symmetry_point(p, projection(p, l));
109 }

110 struct circle {
111     point o;
112     number r;
113     circle() {}
114     circle(point o, number r) : o(o), r(r) {}
115 };

116 bool intersect(const line &l, const circle &a, point &p1, point &p2) {
117     number x = dot(l.a - a.o, l.b - l.a);
118     number y = (l.b - l.a).len2();
119     number d = x * x - y * ((l.a - a.o).len2() - a.r * a.r);
120     if (sgn(d) < 0) return false;
121     point p = l.a - (l.b - l.a) * (x / y), delta = (l.b - l.a) * (_sqrt(d) / y);
122     p1 = p + delta, p2 = p - delta;
123     return true;
124 }
125 bool intersect(const circle &a, const circle &b, point &p1, point &p2) {
126     if (a.o == b.o && cmp(a.r, b.r) == 0)
127         return /* value for coincident circles */ false;
128     number s1 = (b.o - a.o).len();
129     if (cmp(s1, a.r + b.r) > 0 || cmp(s1, std::abs(a.r - b.r)) < 0)
130         return false;
131     number s2 = (a.r * a.r - b.r * b.r) / s1;
132     number aa = (s1 + s2) / 2, bb = (s1 - s2) / 2;
133     point p = (b.o - a.o) * (aa / (aa + bb)) + a.o;
134     point delta = (b.o - a.o).unit().rotate90() * _sqrt(a.r * a.r - aa * aa);
135     p1 = p + delta, p2 = p - delta;
136     return true;
137 }
138 bool tangent(const point &p0, const circle &c, point &p1, point &p2) {
139     number x = (p0 - c.o).len2();
140     number d = x - c.r * c.r;
141     if (sgn(d) < 0) return false;
142     if (sgn(d) == 0)
143         return /* value for point_on_line */ false;
144     point p = (p0 - c.o) * (c.r * c.r / x);
145     point delta = ((p0 - c.o) * (-c.r * _sqrt(d) / x)).rotate90();
146     p1 = c.o + p + delta;
147     p2 = c.o + p - delta;
148     return true;
149 }
150 bool ex_tangent(const circle &a, const circle &b, line &l1, line &l2) {
151     if (cmp(std::abs(a.r - b.r), (b.o - a.o).len()) == 0) {
152         point p1, p2;
153         intersect(a, b, p1, p2);
154         l1 = l2 = line(p1, p1 + (a.o - p1).rotate90());
```

```
        return true;
    } else if (cmp(a.r, b.r) == 0) {
        point dir = b.o - a.o;
        dir = (dir * (a.r / dir.len())).rotate90();
        l1 = line(a.o + dir, b.o + dir);
        l2 = line(a.o - dir, b.o - dir);
        return true;
    } else {
        point p = (b.o * a.r - a.o * b.r) / (a.r - b.r);
        point p1, p2, q1, q2;
        if (tangent(p, a, p1, p2) && tangent(p, b, q1, q2)) {
            l1 = line(p1, q1);
            l2 = line(p2, q2);
            return true;
        } else {
            return false;
        }
    }
}
bool in_tangent(const circle &a, const circle &b, line &l1, line &l2) {
    if (cmp(a.r + b.r, (b.o - a.o).len()) == 0) {
        point p1, p2;
        intersect(a, b, p1, p2);
        l1 = l2 = line(p1, p1 + (a.o - p1).rotate90());
        return true;
    } else {
        point p = (b.o * a.r + a.o * b.r) / (a.r + b.r);
        point p1, p2, q1, q2;
        if (tangent(p, a, p1, p2) && tangent(p, b, q1, q2)) {
            l1 = line(p1, q1);
            l2 = line(p2, q2);
            return true;
        } else {
            return false;
        }
    }
}
```

# Chapter 4

# String

# Chapter 5

# Data Structure

## 5.1 莫队 (ct)

```
int size;
struct Query {
    int l, r, id;
    inline bool operator < (const Queuy &that) const {return l / size != that.l / size ? l < that.l
      ↪ : ((l / size) & 1 ? r < that.r : r > that.r);}
} q[maxn];
int main()
{
    size = (int) sqrt(n * 1.0);
    std::sort(q + 1, q + m + 1);
    int l = 1, r = 0;
    for (int i = 1; i <= m; ++i)
    {
        for (; r < q[i].r; ) add(++r);
        for (; r > q[i].r; ) del(r--);
        for (; l < q[i].l; ) del(l++);
        for (; l > q[i].l; ) add(--l);
        /*
            write your code here.
        */
    }
    return 0;
}
```

## 5.2 ST 表 (ct)

```
int a[maxn], f[20][maxn], n;
int Log[maxn];

void build()
{
    for (int i = 1; i <= n; ++i) f[0][i] = a[i];

    int lim = Log[n];
    for (int j = 1; j <= lim; ++j)
    {
        int *fj = f[j], *fj1 = f[j - 1];
        for (int i = 1; i <= n - (1 << j) + 1; ++i)
            fj[i] = dmax(fj1[i], fj1[i + (1 << (j - 1))]);
    }
}
```

```cpp
int Query(int l, int r)
{
    int k = Log[r - l + 1];
    return dmax(f[k][l], f[k][r - (1 << k) + 1]);
}
int main()
{
    scanf("%d", &n);
    Log[0] = -1;
    for (int i = 1; i <= n; ++i)
    {
        scanf("%d", &a[i]);
        Log[i] = Log[i >> 1] + 1;
    }
    build();
    int q;
    scanf("%d", &q);
    for (; q; --q)
    {
        int l, r; scanf("%d%d", &l, &r);
        printf("%d\n", Query(l, r) );
    }
}
```

## 5.3 可并堆 (ct)

```cpp
struct Node {
    Node *ch[2];
    ll val; int size;
    inline void update()
    {
        size = ch[0] -> size + ch[1] -> size + 1;
    }
} mem[maxn], *rt[maxn];
Node *merge(Node *a, Node *b)
{
    if (a == mem) return b;
    if (b == mem) return a;
    if (a -> val < b -> val) std::swap(a, b);
    // a -> pushdown();
    std::swap(a -> ch[0], a -> ch[1]);
    a -> ch[1] = merge(a -> ch[1], b);
    a -> update();
    return a;
}
```

## 5.4 zkw 线段树 (ct)

```cpp
// must be 0-based !
inline void build()
{
    for (int i = M - 1; i; --i) tr[i] = dmax(tr[i << 1], tr[i << 1 | 1]);
}
inline void Change(int x, int v)
{
    x += M; tr[x] = v; x >>= 1;
    for (; x; x >>= 1) tr[x] = dmax(tr[x << 1], tr[x << 1 | 1]);
}
```

```
inline int Query(int s, int t)
{
    int ret = -0x7fffffff;
    for (s = s + M - 1, t = t + M + 1; s ^ t ^ 1; s >>= 1, t >>= 1)
    {
        if (~s & 1) cmax(ret, tr[s ^ 1]);
        if (t & 1) cmax(ret, tr[t ^ 1]);
    }
    return ret;
}
int main()
{
    int n; scanf("%d", &n);
    for (M = 1; M < n; M <<= 1) ;
    for (int i = 0; i < n; ++i)
        scanf("%d", &tr[i + M]);
    for (int i = n; i < M; ++i) tr[i + M] = -0x7fffffff;
    build();
    int q; scanf("%d", &q);
    for (; q; --q)
    {
        int l, r; scanf("%d%d", &l, &r); --l, --r;
        printf("%d\n", Query(l, r));
    }
    return 0;
}
```

## 5.5 主席树

## 5.6 Splay (ct)

指针版

```
struct Node *null;
struct Node {
    Node *ch[2], *fa;
    int val; bool rev;
    inline bool type()
    {
        return fa -> ch[1] == this;
    }
    inline void pushup()
    {
    }
    inline void pushdown()
    {
        if (rev)
        {
            ch[0] -> rev ^= 1;
            ch[1] -> rev ^= 1;
            std::swap(ch[0], ch[1]);
            rev ^= 1;
        }
    }
    inline void rotate()
    {
        bool d = type(); Node *f = fa, *gf = f -> fa;
        (fa = gf, f -> fa != null) ? fa -> ch[f -> type()] = this : 0;
        (f -> ch[d] = ch[!d]) != null ? ch[!d] -> fa = f : 0;
```

```
        (ch[!d] = f) -> fa = this;
        f -> pushup();
    }
    inline void splay()
    {
        for (; fa != null; rotate())
            if (fa -> fa != null)
                (type() == fa -> type() ? fa : this) -> rotate();
        pushup();
    }
} mem[maxn];
```

数组版

```
// BZOJ - 1500 维修数列
int fa[maxn], ch[maxn][2], a[maxn], size[maxn], cnt;
int sum[maxn], lmx[maxn], rmx[maxn], mx[maxn], v[maxn], id[maxn], root;
bool rev[maxn], tag[maxn];
inline void update(R int x)
{
    R int ls = ch[x][0], rs = ch[x][1];
    size[x] = size[ls] + size[rs] + 1;
    sum[x] = sum[ls] + sum[rs] + v[x];
    mx[x] = gmax(mx[ls], mx[rs]);
    cmax(mx[x], lmx[rs] + rmx[ls] + v[x]);
    lmx[x] = gmax(lmx[ls], sum[ls] + v[x] + lmx[rs]);
    rmx[x] = gmax(rmx[rs], sum[rs] + v[x] + rmx[ls]);
}
inline void pushdown(R int x)
{
    R int ls = ch[x][0], rs = ch[x][1];
    if (tag[x])
    {
        rev[x] = tag[x] = 0;
        if (ls) tag[ls] = 1, v[ls] = v[x], sum[ls] = size[ls] * v[x];
        if (rs) tag[rs] = 1, v[rs] = v[x], sum[rs] = size[rs] * v[x];
        if (v[x]>= 0)
        {
            if (ls) lmx[ls] = rmx[ls] = mx[ls] = sum[ls];
            if (rs) lmx[rs] = rmx[rs] = mx[rs] = sum[rs];
        }
        else
        {
            if (ls) lmx[ls] = rmx[ls] = 0, mx[ls] = v[x];
            if (rs) lmx[rs] = rmx[rs] = 0, mx[rs] = v[x];
        }
    }
    if (rev[x])
    {
        rev[x] ^= 1; rev[ls] ^= 1; rev[rs] ^= 1;
        swap(lmx[ls], rmx[ls]);swap(lmx[rs], rmx[rs]);
        swap(ch[ls][0], ch[ls][1]); swap(ch[rs][0], ch[rs][1]);
    }
}
inline void rotate(R int x)
{
    R int f = fa[x], gf = fa[f], d = ch[f][1] == x;
    if (f == root) root = x;
    (ch[f][d] = ch[x][d ^ 1]) > 0 ? fa[ch[f][d]] = f : 0;
    (fa[x] = gf) > 0 ? ch[gf][ch[gf][1] == f] = x : 0;
    fa[ch[x][d ^ 1] = f] = x;
```

```
48      update(f);
49  }
50  inline void splay(R int x, R int rt)
51  {
52      while (fa[x] != rt)
53      {
54          R int f = fa[x], gf = fa[f];
55          if (gf != rt) rotate((ch[gf][1] == f) ^ (ch[f][1] == x) ? x : f);
56          rotate(x);
57      }
58      update(x);
59  }
60  void build(R int l, R int r, R int rt)
61  {
62      if (l > r) return ;
63      R int mid = l + r >> 1, now = id[mid], last = id[rt];
64      if (l == r)
65      {
66          sum[now] = a[l];
67          size[now] = 1;
68          tag[now] = rev[now] = 0;
69          if (a[l] >= 0) lmx[now] = rmx[now] = mx[now] = a[l];
70          else lmx[now] = rmx[now] = 0, mx[now] = a[l];
71      }
72      else
73      {
74          build(l, mid - 1, mid);
75          build(mid + 1, r, mid);
76      }
77      v[now] = a[mid];
78      fa[now] = last;
79      update(now);
80      ch[last][mid >= rt] = now;
81  }
82  int find(R int x, R int rank)
83  {
84      if (tag[x] || rev[x]) pushdown(x);
85      R int ls = ch[x][0], rs = ch[x][1], lsize = size[ls];
86      if (lsize + 1 == rank) return x;
87      if (lsize >= rank)
88          return find(ls, rank);
89      else
90          return find(rs, rank - lsize - 1);
91  }
92  inline int prepare(R int l, R int tot)
93  {
94      R int x = find(root, l - 1), y = find(root, l + tot);
95      splay(x, 0);
96      splay(y, x);
97      return ch[y][0];
98  }
99  std::queue <int> q;
100 inline void Insert(R int left, R int tot)
101 {
102     for (R int i = 1; i <= tot; ++i ) a[i] = FastIn();
103     for (R int i = 1; i <= tot; ++i )
104         if (!q.empty()) id[i] = q.front(), q.pop();
105         else id[i] = ++cnt;
106     build(1, tot, 0);
107     R int z = id[(1 + tot) >> 1];
108     R int x = find(root, left), y = find(root, left + 1);
```

```
109     splay(x, 0);
110     splay(y, x);
111     fa[z] = y;
112     ch[y][0] = z;
113     update(y);
114     update(x);
115 }
116 void rec(R int x)
117 {
118     if (!x) return ;
119     R int ls = ch[x][0], rs = ch[x][1];
120     rec(ls); rec(rs); q.push(x);
121     fa[x] = ch[x][0] = ch[x][1] = 0;
122     tag[x] = rev[x] = 0;
123 }
124 inline void Delete(R int l, R int tot)
125 {
126     R int x = prepare(l, tot), f = fa[x];
127     rec(x); ch[f][0] = 0;
128     update(f); update(fa[f]);
129 }
130 inline void Makesame(R int l, R int tot, R int val)
131 {
132     R int x = prepare(l, tot), y = fa[x];
133     v[x] = val; tag[x] = 1; sum[x] = size[x] * val;
134     if (val >= 0) lmx[x] = rmx[x] = mx[x] = sum[x];
135     else lmx[x] = rmx[x] = 0, mx[x] = val;
136     update(y); update(fa[y]);
137 }
138 inline void Reverse(R int l, R int tot)
139 {
140     R int x = prepare(l, tot), y = fa[x];
141     if (!tag[x])
142     {
143         rev[x] ^= 1;
144         swap(ch[x][0], ch[x][1]);
145         swap(lmx[x], rmx[x]);
146         update(y); update(fa[y]);
147     }
148 }
149 inline void Query(R int l, R int tot)
150 {
151     R int x = prepare(l, tot);
152     printf("%d\n",sum[x] );
153 }
154 #define inf ((1 << 30))
155 int main()
156 {
157     R int n = FastIn(),  m = FastIn(), l, tot, val;
158     R char op, op2;
159     mx[0] = a[1] = a[n + 2] = -inf;
160     for (R int i = 2; i <= n + 1; i++ )
161     {
162         a[i] = FastIn();
163     }
164     for (R int i = 1; i <= n + 2; ++i) id[i] = i;
165     n += 2; cnt = n; root = (n + 1) >> 1;
166     build(1, n, 0);
167     for (R int i = 1; i <= m; i++ )
168     {
169         op = getc();
```

```
        while (op < 'A' || op > 'Z') op = getc();
        getc(); op2 = getc();getc();getc();getc();getc();
        if (op == 'M' && op2 == 'X')
        {
            printf("%d\n",mx[root] );
        }
        else
        {
            l = FastIn() + 1; tot = FastIn();
            if (op == 'I') Insert(l, tot);
            if (op == 'D') Delete(l, tot);
            if (op == 'M') val = FastIn(), Makesame(l, tot, val);
            if (op == 'R')
                Reverse(l, tot);
            if (op == 'G')
                Query(l, tot);
        }
    }
    return 0;
}
```

## 5.7 Treap (ct)

```
struct Treap {
    Treap *ls, *rs;
    int size;
    bool rev;
    inline void update()
    {
        size = ls -> size + rs -> size + 1;
    }
    inline void set_rev()
    {
        rev ^= 1;
        std::swap(ls, rs);
    }
    inline void pushdown()
    {
        if (rev)
        {
            ls -> set_rev();
            rs -> set_rev();
            rev = 0;
        }
    }
} mem[maxn], *root, *null = mem;
struct Pair {
    Treap *fir, *sec;
};
Treap *build(R int l, R int r)
{
    if (l > r) return null;
    R int mid = l + r >> 1;
    R Treap *now = mem + mid;
    now -> rev = 0;
    now -> ls = build(l, mid - 1);
    now -> rs = build(mid + 1, r);
    now -> update();
```

```
36      return now;
37  }
38  inline Treap *Find_kth(R Treap *now, R int k)
39  {
40      if (!k) return mem;
41      if (now -> ls -> size >= k) return Find_kth(now -> ls, k);
42      else if (now -> ls -> size + 1 == k) return now;
43      else return Find_kth(now -> rs, k - now -> ls -> size - 1);
44  }
45  Treap *merge(R Treap *a, R Treap *b)
46  {
47      if (a == null) return b;
48      if (b == null) return a;
49      if (rand() % (a -> size + b -> size) < a -> size)
50      {
51          a -> pushdown();
52          a -> rs = merge(a -> rs, b);
53          a -> update();
54          return a;
55      }
56      else
57      {
58          b -> pushdown();
59          b -> ls = merge(a, b -> ls);
60          b -> update();
61          return b;
62      }
63  }
64  Pair split(R Treap *now, R int k)
65  {
66      if (now == null) return (Pair) {null, null};
67      R Pair t = (Pair) {null, null};
68      now -> pushdown();
69      if (k <= now -> ls -> size)
70      {
71          t = split(now -> ls, k);
72          now -> ls = t.sec;
73          now -> update();
74          t.sec = now;
75      }
76      else
77      {
78          t = split(now -> rs, k - now -> ls -> size - 1);
79          now -> rs = t.fir;
80          now -> update();
81          t.fir = now;
82      }
83      return t;
84  }
85  inline void set_rev(int l, int r)
86  {
87      R Pair x = split(root, l - 1);
88      R Pair y = split(x.sec, r - l + 1);
89      y.fir -> set_rev();
90      root = merge(x.fir, merge(y.fir, y.sec));
91  }
```

## 5.8 可持久化平衡树 (ct)

```
char str[maxn];
struct Treap
{
    Treap *ls, *rs;
    char data; int size;
    inline void update()
    {
        size = ls -> size + rs -> size + 1;
    }
} *root[maxn], mem[maxcnt], *tot = mem, *last = mem, *null = mem;
inline Treap* new_node(char ch)
{
    *++tot = (Treap) {null, null, ch, 1};
    return tot;
}
struct Pair
{
    Treap *fir, *sec;
};
inline Treap *copy(Treap *x)
{
    if (x == null) return null;
    if(x > last) return x;
    *++tot = *x;
    return tot;
}
Pair Split(Treap *x, int k)
{
    if (x == null) return (Pair) {null, null};
    Pair y;
    Treap *nw = copy(x);
    if (nw -> ls -> size >= k)
    {
        y = Split(nw -> ls, k);
        nw -> ls = y.sec;
        nw -> update();
        y.sec = nw;
    }
    else
    {
        y = Split(nw -> rs, k - nw -> ls -> size - 1);
        nw -> rs = y.fir;
        nw -> update();
        y.fir = nw;
    }
    return y;
}
Treap *Merge(Treap *a, Treap *b)
{
    if (a == null) return b;
    if (b == null) return a;
    Treap *nw;
    if (rand() % (a -> size + b -> size) < a -> size)
    {
        nw = copy(a);
        nw -> rs = Merge(nw -> rs, b);
    }
    else
```

```
    {
        nw = copy(b);
        nw -> ls = Merge(a, nw -> ls);
    }
    nw -> update();
    return nw;
}
Treap *Build(int l, int r)
{
    if (l > r) return null;
    R int mid = l + r >> 1;
    Treap *nw = new_node(str[mid]);
    nw -> ls = Build(l, mid - 1);
    nw -> rs = Build(mid + 1, r);
    nw -> update();
    return nw;
}
int now;
inline void Insert(int k, char ch)
{
    Pair x = Split(root[now], k);
    Treap *nw = new_node(ch);
    root[++now] = Merge(Merge(x.fir, nw), x.sec);
}
inline void Del(int l, int r)
{
    Pair x = Split(root[now], l - 1);
    Pair y = Split(x.sec, r - l + 1);
    root[++now] = Merge(x.fir, y.sec);
}
inline void Copy(int l, int r, int ll)
{
    Pair x = Split(root[now], l - 1);
    Pair y = Split(x.sec, r - l + 1);
    Pair z = Split(root[now], ll);
    Treap *ans = y.fir;
    root[++now] = Merge(Merge(z.fir, ans), z.sec);
}
void Print(Treap *x, int l, int r)
{
    if (!x) return ;
    if (l > r) return;
    R int mid = x -> ls -> size + 1;
    if (r < mid)
    {
        Print(x -> ls, l, r);
        return ;
    }
    if (l > mid)
    {
        Print(x -> rs, l - mid, r - mid);
        return ;
    }
    Print(x -> ls, l, mid - 1);
    printf("%c", x -> data );
    Print(x -> rs, 1, r - mid);
}
void Printtree(Treap *x)
{
    if (!x) return;
    Printtree(x -> ls);
```

```
120     printf("%c", x -> data );
121     Printtree(x -> rs);
122 }
123 int main()
124 {
125     srand(time(0) + clock());
126     null -> ls = null -> rs = null; null -> size = 0; null -> data = 0;
127     int n = F();
128     gets(str + 1);
129     int len = strlen(str + 1);
130     root[0] = Build(1, len);
131     while (1)
132     {
133         last = tot;
134         R char opt = getc();
135         while (opt < 'A' || opt > 'Z')
136         {
137             if (opt == EOF) return 0;
138             opt = getc();
139         }
140         if (opt == 'I')
141         {
142             R int x = F();
143             R char ch = getc();
144             Insert(x, ch);
145         }
146         else if (opt == 'D')
147         {
148             R int l = F(), r = F();
149             Del(l, r);
150         }
151         else if (opt == 'C')
152         {
153             R int x = F(), y = F(), z = F();
154             Copy(x, y, z);
155         }
156         else if (opt == 'P')
157         {
158             R int x = F(), y = F(), z = F();
159             Print(root[now - x], y, z);
160             puts("");
161         }
162     }
163     return 0;
164 }
```

## 5.9 CDQ 分治 (ct)

```
1  struct event
2  {
3      int x, y, id, opt, ans;
4  } t[maxn], q[maxn];
5  void cdq(int left, int right)
6  {
7      if (left == right) return ;
8      R int mid = left + right >> 1;
9      cdq(left, mid);
10     cdq(mid + 1, right);
11     //分成若干个子问题
```

```
    ++now;
    for (int i = left, j = mid + 1; j <= right; ++j)
    {
        for (; i <= mid && q[i].x <= q[j].x; ++i)
            if (!q[i].opt)
                add(q[i].y, q[i].ans);
        //考虑前面的修改操作对后面的询问的影响
        if (q[j].opt)
            q[j].ans += query(q[j].y);
    }
    R int i, j, k = 0;
    //以下相当于归并排序
    for (i = left, j = mid + 1; i <= mid && j <= right; )
    {
        if (q[i].x <= q[j].x)
            t[k++] = q[i++];
        else
            t[k++] = q[j++];
    }
    for (; i <= mid; )
        t[k++] = q[i++];
    for (; j <= right; )
        t[k++] = q[j++];
    for (int i = 0; i < k; ++i)
        q[left + i] = t[i];
}
```

## 5.10 Bitset (ct)

```
namespace Game {
#define maxn 300010
#define maxs 30010
uint b1[32][maxs], b2[32][maxs];
int popcnt[256];
inline void set(R uint *s, R int pos)
{
    s[pos >> 5] |= 1u << (pos & 31);
}
inline int popcount(R uint x)
{
    return popcnt[x >> 24 & 255]
         + popcnt[x >> 16 & 255]
         + popcnt[x >> 8  & 255]
         + popcnt[x       & 255];
}
void main() {
    int n, q;
    scanf("%d%d", &n, &q);

    char *s1 = new char[n + 1];
    char *s2 = new char[n + 1];
    scanf("%s%s", s1, s2);

    uint *anss = new uint[q];

    for (R int i = 1; i < 256; ++i) popcnt[i] = popcnt[i >> 1] + (i & 1);

    #define modify(x, _p)\
    {\
```

```
27          for (R int j = 0; j < 32 && j <= _p; ++j)\
28              set(b##x[j], _p - j);\
29      }
30      for (R int i = 0; i < n; ++i)
31          if (s1[i] == '0') modify(1, 3 * i)
32          else if (s1[i] == '1') modify(1, 3 * i + 1)
33          else modify(1, 3 * i + 2)

34      for (R int i = 0; i < n; ++i)
35          if (s2[i] == '1') modify(2, 3 * i)
36          else if (s2[i] == '2') modify(2, 3 * i + 1)
37          else modify(2, 3 * i + 2)

38      for (int Q = 0; Q < q; ++Q) {
39          R int x, y, l;
40          scanf("%d%d%d", &x, &y, &l); x *= 3; y *= 3; l *= 3;
41          uint *f1 = b1[x & 31], *f2 = b2[y & 31], ans = 0;
42          R int i = x >> 5, j = y >> 5, p, lim;
43          for (p = 0, lim = l >> 5; p + 8 < lim; p += 8, i += 8, j += 8)
44          {
45              ans += popcount(f1[i + 0] & f2[j + 0]);
46              ans += popcount(f1[i + 1] & f2[j + 1]);
47              ans += popcount(f1[i + 2] & f2[j + 2]);
48              ans += popcount(f1[i + 3] & f2[j + 3]);
49              ans += popcount(f1[i + 4] & f2[j + 4]);
50              ans += popcount(f1[i + 5] & f2[j + 5]);
51              ans += popcount(f1[i + 6] & f2[j + 6]);
52              ans += popcount(f1[i + 7] & f2[j + 7]);
53          }
54          for (; p < lim; ++p, ++i, ++j) ans += popcount(f1[i] & f2[j]);
55          R uint S = (1u << (l & 31)) - 1;
56          ans += popcount(f1[i] & f2[j] & S);
57          anss[Q] = ans;
58      }

59      output_arr(anss, q * sizeof(uint));
60  }
61  }
```

# Chapter 6

# Others

## 6.1 vimrc (gy)

```
se et ts=4 sw=4 sts=4 nu sc sm lbr is hls mouse=a
sy on
ino <tab> <c-n>
ino <s-tab> <tab>
au winnew * winc L

nm <f6> ggVG"+y
nm <f7> :w<cr>:make<cr>
nm <f8> :!@@<cr>
nm <f9> :!@@ < in<cr>
nm <s-f9> :!(time @@ < in &>> out) &>> out<cr>:sp out<cr>

au filetype cpp cm @@ ./a.out | se cin fdm=syntax mp=g++\ %\ -std=c++11\ -Wall\ -Wextra\ -O2

map <c-p> :ha<cr>
se pheader=%n\ %f

au filetype java cm @@ java %< | se cin fdm=syntax mp=javac\ %
au filetype python cm @@ python % | se si fdm=indent
au bufenter *.kt setf kotlin
au filetype kotlin cm @@ kotlin _%<Kt | se si mp=kotlinc\ %
```

## 6.2 Java Template (gy)

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.math.BigDecimal;
import java.math.BigInteger;
import java.math.RoundingMode;
import java.util.ArrayDeque;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Comparator;
import java.util.Deque;
import java.util.LinkedList;
import java.util.List;
import java.util.Scanner;
import java.util.StringTokenizer;
```

```java
public class Template {
    // Input
    private static BufferedReader reader;
    private static StringTokenizer tokenizer;

    private static String next() {
        try {
            while (tokenizer == null || !tokenizer.hasMoreTokens())
                tokenizer = new StringTokenizer(reader.readLine());
        } catch (IOException e) {
            // do nothing
        }
        return tokenizer.nextToken();
    }

    private static int nextInt() {
        return Integer.parseInt(next());
    }

    private static double nextDouble() {
        return Double.parseDouble(next());
    }

    private static BigInteger nextBigInteger() {
        return new BigInteger(next());
    }

    public static void main(String[] args) {
        reader = new BufferedReader(new InputStreamReader(System.in));
        Scanner scanner = new Scanner(System.in);
        while (scanner.hasNext())
            scanner.next();
    }

    // BigInteger & BigDecimal
    private static void bigDecimal() {
        BigDecimal a = BigDecimal.valueOf(1.0);
        BigDecimal b = a.setScale(50, RoundingMode.HALF_EVEN);
        BigDecimal c = b.abs();
        // if scale omitted, b.scale is used
        BigDecimal d = c.divide(b, 50, RoundingMode.HALF_EVEN);
        // since Java 9
        BigDecimal e = d.sqrt(new MathContext(50, RoundingMode.HALF_EVEN));
        BigDecimal x = new BigDecimal(BigInteger.ZERO);
        BigInteger y = BigDecimal.ZERO.toBigInteger(); // RoundingMode.DOWN
        y = BigDecimal.ZERO.setScale(0, RoundingMode.HALF_EVEN).unscaledValue();
    }

    // sqrt for Java 8
    private static BigDecimal sqrt(BigDecimal a, int scale, RoundingMode mode) {
        if (a.equals(BigDecimal.ZERO))
            return BigDecimal.ZERO;
        a = a.setScale(scale, mode);
        BigDecimal ans = a;
        BigDecimal TWO = BigDecimal.valueOf(2L);
        for (int i = 1; i <= scale; i++)
            ans = ans.add(a.divide(ans, scale, mode)).divide(TWO, scale, mode);
        return ans;
    }

    private static BigInteger sqrt(BigInteger a) {
```

```java
        BigInteger about = BigInteger.ZERO.setBit(a.bitLength() / 2);
        return sqrt(new BigDecimal(a.toString()), new BigDecimal(about.toString())).setScale(0,
          ↪ RoundingMode.FLOOR).unscaledValue();
    }

    private static BigDecimal sqrt(BigDecimal a, BigDecimal initial) {
        if (a.equals(BigDecimal.ZERO))
            return BigDecimal.ZERO;
        a = a.setScale(50, RoundingMode.HALF_EVEN);
        BigDecimal ans = initial;
        for (int i = 1; i <= 10; i++)
            ans = ans.add(a.divide(ans, RoundingMode.HALF_EVEN)).divide(BigDecimal.valueOf(2),
              ↪ RoundingMode.HALF_EVEN);
        return ans;
    }

    // ArrayList
    private static void arrayList() {
        List<Integer> list = new ArrayList<>();
        // Generic array is banned
        List[] lists = new List[100];
        lists[0] = new ArrayList<Integer>();
        // for List<Integer>, remove(Integer) stands for element, while remove(int) stands for
          ↪ index
        list.remove(list.get(1));
        list.remove(list.size() - 1);
        list.clear();
    }

    // Queue
    private static void queue() {
        LinkedList<Integer> queue = new LinkedList<>();
        // return the value without popping
        queue.peek();
        // pop and return the value
        queue.poll();
        Deque<Integer> deque = new ArrayDeque<>();
        deque.peekFirst();
        deque.peekLast();
        deque.pollFirst();
    }

    // Others
    private static void others() {
        Arrays.sort(new int[10]);
        Arrays.sort(new Integer[10], (a, b) -> {
            if (a.equals(b)) return 0;
            if (a > b) return -1;
            return 1;
        });
        Arrays.sort(new Integer[10], Comparator.comparingInt((a) -> (int) a).reversed());
        long a = 1_000_000_000_000_000_000L;
        int b = Integer.MAX_VALUE;
        int c = 'a';
    }
}
```

## 6.3 Big Fraction (gy)

```
fun gcd(a: Long, b: Long): Long = if (b == 0L) a else gcd(b, a % b)

class Fraction(val a: BigInteger, val b: BigInteger) {
    constructor(a: Long, b: Long) : this(BigInteger.valueOf(a / gcd(a, b)), BigInteger.valueOf(b / gcd(a, b)))

    operator fun plus(o: Fraction): Fraction {
        var gcd = b.gcd(o.b)
        val tempProduct = (b / gcd) * (o.b / gcd)
        var ansA = a * (o.b / gcd) + o.a * (b / gcd)
        val gcd2 = ansA.gcd(gcd)
        ansA /= gcd2
        gcd /= gcd2
        return Fraction(ansA, gcd * tempProduct)
    }

    operator fun minus(o: Fraction): Fraction {
        var gcd = b.gcd(o.b)
        val tempProduct = (b / gcd) * (o.b / gcd)
        var ansA = a * (o.b / gcd) - o.a * (b / gcd)
        val gcd2 = ansA.gcd(gcd)
        ansA /= gcd2
        gcd /= gcd2
        return Fraction(ansA, gcd * tempProduct)
    }

    operator fun times(o: Fraction): Fraction {
        val gcd1 = a.gcd(o.b)
        val gcd2 = b.gcd(o.a)
        return Fraction((a / gcd1) * (o.a / gcd2), (b / gcd2) * (o.b / gcd1))
    }
}
```

## 6.4 模拟退火 (ct)

```
db ans_x, fans;
inline double rand01() {return rand() / 2147483647.0;}
inline double randp() {return (rand() & 1 ? 1 : -1) * rand01();}
inline double f(double x)
{
    /*
        write your function here.
    */
    if (maxx < fans) {fans = maxx; ans_x = x;}
    return maxx;
}
int main()
{
    srand(time(NULL) + clock());
    db x = 0, fnow = f(x);
    fans = 1e30;
    for (db T = 1e4; T > 1e-4; T *= 0.997)
    {
        db nx = x + randp() * T, fnext = f(nx);
        db delta = fnext - fnow;
        if (delta < 1e-9 || exp(-delta / T) > rand01())
```

```
        {
            x = nx;
            fnow = fnext;
        }
    }
    return 0;
}
```

## 6.5 三分 (ct)

```
inline db cubic_search()
{
    double l = -1e4, r = 1e4;
    for (int i = 1; i <= 100; ++i)
    {
        double ll = (l + r) * 0.5;
        double rr = (ll + r) * 0.5;
        if (check(ll) < check(rr)) r = rr;
        else l = ll;
    }
    return (l + r) * 0.5;
}
```

## 6.6 博弈论模型 (gy)

- Wythoff's game
  给定两堆石子，每次可以从任意一堆中取至少一个石子，或从两堆中取相同的至少一个石子，取走最后石子的胜
  先手胜当且仅当石子数满足：
  $\lfloor (b-a) \times \phi \rfloor = a, (a \leq b, \phi = \frac{\sqrt{5}+1}{2})$
  先手胜对应的石子数构成两个序列：
  Lower Wythoff sequence: $a_n = \lfloor n \times \phi \rfloor$
  Upper Wythoff sequence: $b_n = \lfloor n \times \phi^2 \rfloor$

- Fibonacci nim
  给定一堆石子，第一次可以取至少一个、少于石子总数数量的石子，之后每次可以取至少一个、不超过上次取石子数量两倍的石子，取走最后石子的胜
  先手胜当且仅当石子数为斐波那契数

## 6.7 积分表