

Contents

1 Foundations	1
1.1 PyMath	1
1.2 Java Integer	1
1.3 Java String	1
1.4 Java String builder	2
1.5 Java Math	2
2 Mathematics & Number Theory	2
2.1 Number Theory	2
2.2 Combinatorics	3
2.3 Geometry	3
3 Data Structure	3
3.1 MST	3
3.2 SegmentTree	3
3.3 HLD	4
3.4 PST	4
3.5 Trie	5
3.6 BIT 單修區查	5
3.7 BIT 區修單查	5
3.8 BIT 區修區查	5
4 Graph	5
4.1 cut vertex AND bridges	5
4.2 SCC - Tarjan	6
4.3 BCC - Tarjan	6
4.4 Convex	6
4.5 Max Flow	6
4.6 min cut max flow	7
5 String	7
5.1 KMP	7
5.2 ACAM	7
6 Techniques	8
6.1 二分搜	8
6.2 倍增 LCA	8
6.3 SG	8
7 DP	8
7.1 輪廓線 DP	8
7.2 數位 DP	8
7.3 樹 DP	8

1 Foundations

1.1 PyMath

```

1 import math
2
3     math.ceil(x) #上高斯
4     math.floor(x) #下高斯
5     math.factorial(x) #階乘
6     math.fabs(x) #絕對值
7     math.fsum(arr) #求和
8     math.gcd(x, y)
9     math.exp(x) # e^x
10    math.log(x, base)
11    math.log2(x)
12    math.log10(x)
13    math.sqrt(x)
14    math.pow(x, y, mod)
15    math.sin(x) # cos, tan, asin, acos, atan,
16      atan2, sinh ...
17    math.hypot(x, y) #歐幾里得範數
18    math.degrees(x) #x從弧度轉角度
19    math.radians(x) #x從角度轉弧度
20    math.gamma(x) #x的gamma函數
21    math.pi #const
22    math.e #const
23    math.inf

```

1.2 Java Integer

```

1 // 常量
2 MAX_VALUE, MIN_VALUE, BYTES, SIZE, TYPE
3
4 // 轉換/解析
5 static int parseInt(String s)
6 static int parseInt(String s, int radix)
7 static int parseUnsignedInt(String s)
8 static int parseUnsignedInt(String s, int
9   radix)
10 static Integer valueOf(int i)
11 static Integer valueOf(String s)
12 static Integer valueOf(String s, int radix)
13 static String toString(int i)
14 static String toString(int i, int radix)
15 static String toUnsignedString(int i)
16 static String toUnsignedString(int i, int
17   radix)
18 static long toUnsignedLong(int x)
19 static Integer decode(String nm)
20   // 支援 0x/0/# 前綴
21 static Integer getInteger(String nm[, int
22   val]) // 從系統屬性讀取整數
23
24 // 比較/雜湊/聚合
25 static int compare(int x, int y)
26 static int compareUnsigned(int x, int y)
27 static int hashCode(int value)
28 static int min(int a, int b)
29 static int max(int a, int b)
30 static int sum(int a, int b)
31
32 // 位元操作
33 static int bitCount(int i) // 設定位數
34 static int highestOneBit(int i)
35 static int lowestOneBit(int i)
36 static int numberOfLeadingZeros(int i)
37 static int numberOfTrailingZeros(int i)
38 static int rotateLeft(int i, int distance)
39 static int rotateRight(int i, int distance)
40 static int reverse(int i)
41 static int reverseBytes(int i)
42
43 // 無號運算
44 static int divideUnsigned(int dividend, int
45   divisor)

```

```

41 static int remainderUnsigned(int dividend,
42   int divisor)

```

1.3 Java String

```

1 // 查詢
2 int length()
3 boolean isEmpty()
4 boolean isBlank() // (since 11)
5 char charAt(int index)
6 int codePointAt(int index)
7 int codePointBefore(int index)
8 int codePointCount(int beginIndex, int
9   endIndex)
10 boolean contains(CharSequence s)
11 boolean startsWith(String prefix[, int
12   toffset])
13 boolean endsWith(String suffix)
14 int indexOf(String str[, int fromIndex])
15 int lastIndexOf(String str[, int
16   fromIndex])
17 // 取子字串/子序列
18 String substring(int beginIndex)
19 String substring(int beginIndex, int
20   endIndex)
21 CharSequence subSequence(int beginIndex, int
22   endIndex)
23
24 // 比較/等價
25 boolean equals(Object obj)
26 boolean equalsIgnoreCase(String
27   anotherString)
28 int compareTo(String anotherString)
29 int compareToIgnoreCase(String str)
30 boolean matches(String regex)
31 boolean regionMatches(int toffset, String
32   other, int offset, int len)
33 boolean regionMatches(boolean ignoreCase,
34   int toffset, String other, int offset,
35   int len)
36
37 // 建構/轉換/連接
38 String concat(String str)
39 String replace(char oldChar, char newChar)
40 String replace(CharSequence target,
41   CharSequence replacement)
42 String replaceAll(String regex, String
43   replacement)
44 String replaceFirst(String regex, String
45   replacement)
46 String[] split(String regex[, int limit])
47 String toLowerCase()
48 String toUpperCase()
49 String trim()
50 String strip() // (since 11)
51 String stripLeading() // (since 11)
52 String stripTrailing() // (since 11)
53 String repeat(int count) // (since 11)
54 IntStream chars()
55 Stream<String> lines() // (since 11)
56 String intern()
57
58 // 靜態工具
59 static String format(String format,
60   Object... args)
61 static String join(CharSequence delimiter,
62   CharSequence... elements)
63 static String join(CharSequence delimiter,
64   Iterable<? extends CharSequence>
65   elements)
66 static String
67   valueOf(primitive/char[]/Object)
68 static String copyValueOf(char[] data[, int
69   offset, int count])

```

1.4 Java String builder

```

1 // 長度/容量
2 int length()
3 int capacity()
4 void ensureCapacity(int minimumCapacity)
5 void trimToSize()
6 void setLength(int newLength)
7
8 // 存取/修改
9 char      charAt(int index)
10 void     setCharAt(int index, char ch)
11 StringBuilder append(... 各種型別 ...)
12 StringBuilder insert(int offset, ... 各種型別
    ...)
13 StringBuilder delete(int start, int end)
14 StringBuilder deleteCharAt(int index)
15 StringBuilder replace(int start, int end,
    String str)
16 StringBuilder reverse()
17
18 // 子字串/查找
19 String      substring(int start)
20 String      substring(int start, int end)
21 CharSequence subSequence(int start, int end)
22 int        indexOf(String str[], int
    fromIndex])
23 int        lastIndexOf(String str[], int
    fromIndex])
24
25 // 轉換
26 String toString()

```

1.5 Java Math

```

1 // 常量
2 static final double E, PI
3
4 // 絶對值/比較
5 static int/long/float,double abs(x)
6 static T max(a, b)
7 static T min(a, b)
8
9 // 取整/四捨五入
10 static double floor(double a)
11 static double ceil(double a)
12 static double rint(double a)          // 最接近整數(偶數優先)
13 static long round(double a) / int
    round(float a)
14 static int   floorDiv(int x, int y)
15 static int   floorMod(int x, int y)
16
17 // 溢位保護(exact 系列, Java 8+)
18 static int/long addExact(a, b)
19 static int/long subtractExact(a, b)
20 static int/long multiplyExact(a, b)
21 static int/long incrementExact(a)
22 static int/long decrementExact(a)
23 static int   toIntExact(long value)
24 static int/long negateExact(a)
25
26 // 指對數/幂根
27 static double pow(double a, double b)
28 static double sqrt(double a)
29 static double cbrt(double a)
30 static double exp(double a)
31 static double expm1(double x)
32 static double log(double a)
33 static double log10(double a)
34 static double log1p(double x)
35
36 // 三角/雙曲
37 static double sin/cos/tan(double a)
38 static double asin/acos/atan(double a)
39 static double atan2(double y, double x)

```

```

40 static double sinh/cosh/tanh(double a)
41
42 // 其他實用
43 static double hypot(double x, double y)
44 static double toDegrees(double angrad)
45 static double toRadians(double angdeg)
46 static double copySign(double magnitude,
    double sign)
47 static double nextUp/nextDown(double a)
48 static double nextAfter(double start, double
    direction)
49 static double ulp(double d)
50 static double random()
51 static double scalb(double d, int
    scaleFactor)
52 static double fma(double a, double b, double
    c) // (since 8)
53 static long multiplyHigh(long x, long y)
    // (since 9)
54 static long multiplyFull(int x, int y)
    // (since 9, 回傳 long)

```

7 ll d = extgcd(b, a%b, c, x, y), tmp =
 x;
 x = y;
 y = tmp - (a/b)*y;
 return d;

8 }
 // x, y 為 ax + by = gcd(a, b) 的解
 x = x * c / d; // ax + by = c 的解
 y = y * c / d; // ax + by = c 的解

9 Modular Inverse Basics 對於 $\gcd(a, m) = 1$, 存在唯一
 $a^{-1} \in [0, m)$ 使得 $a \cdot a^{-1} \equiv 1 \pmod{m}$ 。
 • 用途：實作模除法、解線性同餘、組合數除法、CRT 等。
 • 求法：擴展歐幾里得適用任意模數；若 m 為質數可用
 $a^{m-2} \pmod{m}$ 。
 • 存在條件：僅當 $\gcd(a, m) = 1$ ；不互質時可除以
 $d = \gcd(a, m)$ 後再檢查。

10 1 g = gcd(a, m, c, x, y);
 2 if(g == 1){
 3 inv = (x % m + m) % m;
 4 }

11 Congruence ($a \equiv b \pmod{m}$) Essentials
 • $a \equiv b \pmod{m} \iff m | (a - b)$, 同餘類以差整
 除判斷。
 • $a \equiv b \pmod{m} \Rightarrow f(a) \equiv f(b) \pmod{m}$ 對所有
 以整數係數的多項式 f 成立。
 • 若 $a \equiv b \pmod{m}$ 且 $c \equiv d \pmod{m}$, 則
 $a \pm c \equiv b \pm d \pmod{m}$, $ac \equiv bd \pmod{m}$,
 $a^n \equiv b^n \pmod{m}$ 。
 • 若 $\gcd(k, m) = 1$ 且 $ka \equiv kb \pmod{m}$, 可約去
 $k: a \equiv b \pmod{m}$ 。
 • 若 $a \equiv b \pmod{m}$, 則 $a \equiv b \pmod{d}$ 對任何 d 整
 除 m 亦成立。
 • 若 $d | a, b, m$, 則 $\frac{a}{d} \equiv \frac{b}{d} \pmod{\frac{m}{d}}$

12 Bézout's Identity
 若 $a, b \in \mathbb{Z}, \exists x, y \in \mathbb{Z}$ 使得 $ax + by = \gcd(a, b)$ 。
 任何方程 $ax + by = c$ 有整數解當且僅當 $\gcd(a, b) | c$ 。擴展
 歐幾里得演算法可同時求得 \gcd 與一組 (x, y) , 常用於計算模逆
 與結合 China Remainder。

根據貝祖定理，只有當 $d | c$ 時， $ax + by = c$ 才有整數解。
 對於 $ax + by = c$ 假設有整數解，且 $\gcd(a, b) = 1$ 使用
 extgcd 求出 (a_0, b_0) 的特解後，根據逆模元可得出通解：
 $a = a_0 + \frac{b}{d}t, b = b_0 - \frac{a}{d}t$ 其中 t 可以為任意整數

13 Chinese Remainder Theorem
 設同餘系統
 $x \equiv a_i \pmod{m_i} \quad (i = 1, \dots, k)$,
 其中 m_i 兩兩互質，令 $M = \prod_{i=1}^k m_i$, $M_i = M/m_i$, 再
 取 $t_i \equiv M_i^{-1} \pmod{m_i}$ 。則唯一解 (模 M) 為
 $x \equiv \sum_{i=1}^k a_i M_i t_i \pmod{M}$.

14 兩式合併 (允許非互質)
 // solve x a1 (mod m1), x a2 (mod m2)
 // return {x0, lcm}; if no solution, lcm = -1
 pair<ll, ll> crt(ll a1, ll m1,
 ll a2, ll m2) {
 1 ll g = std::gcd(m1, m2);
 2 if ((a2 - a1) % g != 0) return {0, -1};
 // no solution
 3 ll lcm = m1 / g * m2;
 4 ll m1_reduced = m1 / g;
 5 ll m2_reduced = m2 / g;

2 Mathematics & Number Theory

2.1 Number Theory

Fermat's Little Theorem:

$$a^{p-1} \equiv 1 \pmod{p} \quad (\gcd(a, p) = 1, p \text{ prime})$$

Euler's Theorem:

$$a^{\varphi(n)} \equiv 1 \pmod{n} \quad (\gcd(a, n) = 1)$$

Modular Inverse:

$$a^{-1} \equiv a^{p-2} \pmod{p} \quad (\gcd(a, p) = 1, p \text{ prime})$$

Euler Totient:

$$\varphi(n) = n \prod_{p|n} \left(1 - \frac{1}{p}\right)$$

Fast Modular Exponentiation

```

long long mod_pow(long long a, long long b,
    long long mod) {
    long long res = 1 % mod;
    while (b > 0) {
        if (b & 1) res = res * a % mod;
        a = a * a % mod;
        b >>= 1;
    }
    return res;
}

```

Counting Coprimes Below n

$$\forall n > 0, \sum_{d|n} \varphi(d) = n, \quad \sum_{d|n} \mu(d) \left\lfloor \frac{n}{d} \right\rfloor = 1,$$

其中 μ 為莫比烏斯函數；常用於反演及計算互質對數量。

Modulo Arithmetic Quick Facts

$$\begin{aligned}
 (a \pm b) \bmod m &= ((a \bmod m) \pm (b \bmod m)) \bmod m, \\
 (ab) \bmod m &= ((a \bmod m)(b \bmod m)) \bmod m, \\
 (a^k) \bmod m &= ((a \bmod m)^k) \bmod m, \\
 (-a) \bmod m &= (m - (a \bmod m)) \bmod m.
 \end{aligned}$$

若 $\gcd(a, m) = 1$, 可計算乘法逆元 a^{-1} 並套用 $(a/b) \bmod m \equiv a \cdot b^{-1} \bmod m$ 。

Extended Euclidean algorithm 給定 a, b, c , 求 $ax + by = c$ 的解

```

1 ll extgcd(ll a, ll b, ll &x, ll &y){
2     if(b == 0){
3         x = 1;
4         y = 0;
5         return a;
6     }

```

```

11    ll diff = (a2 - a1) / g % m2_reduced;
12    if (diff < 0) diff += m2_reduced;
13
14    ll inv = mod_pow(m1_reduced, m2_reduced
15        - 1, m2_reduced);
16    ll step = diff * inv % m2_reduced;
17    ll x0 = (a1 + step * m1) % lcm;
18    if (x0 < 0) x0 += lcm;
19    return {x0, lcm};
20 }

遞增地將每個同餘式與當前解做合併即可取得最終答案，也能偵測無解情況。給定  $a, b, c$ , 求  $ax + by = c$  的解

1  ll extgcd(ll a, ll b, ll c, ll &x, ll
2      &y){
3      if(b == 0){
4          x = c/a ;
5          y = 0 ;
6          return a ;
7      }
8      ll d = extgcd(b, a%b, c, x, y), tmp =
9          x ;
10     x = y ;
11     y = tmp - (a/b)*y ;
12     return d ;
13 }
```

2.2 Combinatorics

Binomial Coefficient Identities

$$\begin{aligned} \binom{n}{k} &= \frac{n!}{k!(n-k)!}, \\ \binom{n}{k} &= \binom{n-1}{k} + \binom{n-1}{k-1}, \\ \sum_{k=0}^n \binom{n}{k} &= 2^n, \quad \sum_{k=0}^n k \binom{n}{k} = n2^{n-1}. \end{aligned}$$

Stars and Bars 非負整數解數量：

$$x_1 + x_2 + \dots + x_k = n \Rightarrow \binom{n+k-1}{k-1}.$$

若各變數至少為 1，將 $x_i = y_i + 1$ 轉為非負情況即可。

Inclusion-Exclusion Principle 對集合

A_1, \dots, A_k :

$$\left| \bigcup_{i=1}^k A_i \right| = \sum_{i=1}^k |A_i| - \sum_{1 \leq i < j \leq k} |A_i \cap A_j| - \dots + (-1)^{k-1} |A_1 \cap \dots \cap A_k|.$$

計算滿足限制的排列或整數解時廣泛使用。

Catalan Numbers 基本定義：

$$C_n = \frac{1}{n+1} \binom{2n}{n}, \quad C_{n+1} = \frac{4n+2}{n+2} C_n.$$

$$C = 1, 1, 2, 5, 14, 42, 132, 429, 1430, 4862, \dots$$

常見應用包含合法括號、凸多邊形三角剖分、二元樹結構計數等。

2.3 Geometry

對於 V 個點, E 條邊, F 個面, C 個連通分量

$$V + F = E + 2$$

$$V + F = E + C + 1$$

點座標均是整數或是正方形格子點的簡單多邊形，其面積 A 和內部點數量 i, 邊上格點數量 b 的關係為

$$A = i + \frac{b}{2} - 1$$

3 Data Structure

3.1 MST

```

1 struct Edge{
2     int u, v, w ;
3     // 這是最大生成樹，最小生成樹要改成 w < o.w
4     bool operator>(const Edge &o) const
5         {return w > o.w ;}
6 }
7 int par[N] ;
8 int sz[N] ;
9 int sum ;
10 vector<Edge> edge ;
11 void init(){
12     edge.clear() ;
13     for ( int i=0 ; i<N ; i++ ){
14         par[i] = i ;
15         sz[i] = 1 ;
16     }
17     sum = 0 ;
18 }
19 int find(int x){
20     if(x == par[x]) return x ;
21     return par[x] = find(par[x]) ;
22 }
23 int merge(int x, int y){
24     x = find(x) ;
25     y = find(y) ;
26
27     if(x == y) return 0 ;
28     if(sz[x] > sz[y]) swap(x, y) ;
29     par[x] = y ;
30     sz[y] += sz[x] ;
31
32     return 1 ;
33 }
34 void MST(){
35     int cnt = 0 ;
36     for ( int i=0 ; i<edge.size() && cnt < n-1
37         ; i++ ){
38         auto [u, v, w] = edge[i] ;
39         if(merge(u, v)){
40             cnt++ ;
41             sum -= w ;
42         }
43     }
44 }
45 int main(){
46     for ( int i=0 ; i<m ; i++ ){
47         scanf("%d%d%d", &u, &v, &w) ;
48         edge.push_back({u, v, w}) ;
49         sum += w ;
50     }
51
52     sort(edge.begin(), edge.end(),
53          greater<Edge>()) ;
54     MST() ;
55 }
```

3.2 SegmentTree

```

1 #define lc (id << 1)
2 #define rc ((id << 1) | 1)
3
4 struct LazyTag{
5     // type 0 : increase val
6     // type 1 : set to val
7     // type 1 can overwrite type 0
8 }
```

```

8     int type ;
9     ll val ;
10 } ;
11
12 struct Node{
13     LazyTag tag ;
14     ll sum ;
15     int sz ;
16 }seg[Maxn << 2] ;
17
18 class SegmentTree{
19 private:
20     void pull(int id){
21         seg[id].sum = seg[lc].sum +
22             seg[rc].sum ;
23     }
24
25     void AddTag(int id, LazyTag &tag){
26         if(tag.type == 0){
27             seg[id].sum += tag.val *
28                 seg[id].sz ;
29             seg[id].tag.val += tag.val ;
30         }
31         else{
32             seg[id].sum = tag.val *
33                 seg[id].sz ;
34             seg[id].tag = {1, tag.val} ;
35         }
36
37     void push(int id){
38         AddTag(lc, seg[id].tag) ;
39         AddTag(rc, seg[id].tag) ;
40         seg[id].tag = {0, 0} ;
41     }
42
43     public:
44     void build(int L=1, int R=n, int id=1){
45         seg[id].sum = 0 ;
46         seg[id].tag = {0, 0} ;
47         seg[id].sz = 1 ;
48
49         if(L == R){
50             seg[id].sum = arr[L] ;
51             return ;
52         }
53
54         int M = (L + R) >> 1 ;
55         build(L, M, lc) ;
56         build(M+1, R, rc) ;
57
58         pull(id) ;
59         seg[id].sz = seg[lc].sz + seg[rc].sz ;
60     }
61
62     void modify(int l, int r, LazyTag &tag,
63                int L=1, int R=n, int id=1){
64         if(l <= L && R <= r){
65             AddTag(id, tag) ;
66             return ;
67         }
68
69         push(id) ;
70         int M = (L + R) >> 1 ;
71         if(r <= M) modify(l, r, tag, L, M,
72                           lc) ;
73         else if(l > M) modify(l, r, tag, M+1,
74                               R, rc) ;
75         else{
76             modify(l, r, tag, L, M, lc) ;
77             modify(l, r, tag, M+1, R, rc) ;
78         }
79     }
80
81     pull(id) ;
82 }
```

```

77    ll query(int l, int r, int L=1, int R=n,
78        int id=1){
79        if(l <= L && R <= r) return
80            seg[id].sum ;
81
82        push(id) ;
83        int M = (L + R) >> 1 ;
84        if(r <= M) return query(l, r, L, M,
85            lc) ;
86        else if(l > M) return query(l, r,
87            M+1, R, rc) ;
88        else return query(l, r, L, M, lc) +
89            query(l, r, M+1, R, rc) ;
90    }
91 }tree ;

```

3.3 HLD

```

1 /* HLD */
2 int fa[Maxn], top[Maxn], son[Maxn],
3     sz[Maxn], dep[Maxn] = {0}, dfn[Maxn],
4     rnk[Maxn], dfscnt = 0 ;
5
6 void dfs1(int u, int from){
7     fa[u] = from ;
8     dep[u] = dep[from] + 1 ;
9     sz[u] = 1 ;
10
11    for ( auto v : g[u] ) if(v != from){
12        dfs1(v, u) ;
13        sz[u] += sz[v] ;
14        if(son[u] == -1 || sz[v] > sz[son[u]])
15            son[u] = v ;
16    }
17
18    void dfs2(int u, int t){
19        top[u] = t ;
20        dfn[u] = ++dfscnt ;
21        rnk[dfscnt] = u ;
22
23        if(son[u] == -1) return ;
24
25        for ( auto v : g[u] ) if(v != fa[u] && v
26            != son[u]){
27            dfs2(v, v) ;
28        }
29
30 /* Segment Tree */
31 #define lc (id << 1)
32 #define rc ((id << 1) | 1)
33
34 struct ColorSeg{
35     int left, right, tot ;
36
37     ColorSeg operator+(const ColorSeg &o)
38         const {
39             if(tot == 0) return o ;
40             if(o.tot == 0) return *this ;
41
42             ColorSeg tmp ;
43             tmp.left = left ;
44             tmp.right = o.right ;
45             tmp.tot = tot + o.tot - (right ==
46                 o.left) ;
47
48             return tmp ;
49         }
50
51     struct Node{
52         ColorSeg color ;
53         int tag ;
54     }seg[Maxn << 2] ;

```

```

54
55     class SegmentTree{
56     private:
57         void pull(int id){
58             // normal pull
59         }
60
61         void AddTag(int id, int tag){
62             // normal AddTag
63         }
64
65         void push(int id){
66             // normal push
67         }
68
69         void modify(int l, int r, int tag, int
70             L=1, int R=n, int id=1){
71             // normal modify
72         }
73
74         ColorSeg query(int l, int r, int L=1, int
75             R=n, int id=1){
76             // normal query
77         }
78
79         public:
80             void build(int L=1, int R=n, int id=1){
81                 // normal build
82             }
83
84             // update val from u to v (simple path)
85             void update(int u, int v, int val){
86                 while(top[u] != top[v]){
87                     if(dep[top[u]] < dep[top[v]]) swap(u,
88                         v) ;
89                     modify(dfn[top[u]], dfn[u], val) ;
90                     u = fa[top[u]] ;
91                 }
92
93                 if(dep[u] < dep[v]) swap(u, v) ;
94                 modify(dfn[v], dfn[u], val) ;
95
96                 // get sum from u to v (simple path)
97                 int get(int u, int v){
98                     pair<int, ColorSeg> U, V ;
99                     ColorSeg M ;
100                    U = {u, {0, 0, 0}} ;
101                    V = {v, {0, 0, 0}} ;
102
103                    while(top[U.first] != top[V.first]){
104                        if(dep[top[U.first]] <
105                            dep[top[V.first]]) swap(U, V) ;
106                        U.second = query(dfn[top[U.first]],
107                            dfn[U.first]) + U.second ;
108                        U.first = fa[top[U.first]] ;
109
110                        if(dep[U.first] < dep[V.first]) swap(U,
111                            V) ;
112
113                        M = query(dfn[V.first], dfn[U.first]) ;
114
115                        return (U.second.tot + V.second.tot +
116                                M.tot) - (U.second.left == M.right)
117                                - (V.second.left == M.left) ;
118
119                    }
120
121                }
122 }tree ;
123
124 void init(){
125     memset(son, -1, sizeof(son)) ;
126 }

```

3.4 PST

```

1 // Find range k-th largest number
2 struct Node{
3     int sum, left, right ;

```

```

4 }seg[Maxn + 20 * Maxn] ;
5
6 class PersistentSegmentTree{
7     private:
8         int n ;
9         int cnt ;
10        vector<int> version ;
11
12        int build(int L, int R){
13            int cur_cnt = cnt++ ;
14            if(L == R){
15                seg[cur_cnt] = {0, 0, 0} ;
16                return cur_cnt ;
17            }
18
19            int M = (L + R) >> 1 ;
20            int lc = build(L, M) ;
21            int rc = build(M+1, R) ;
22
23            seg[cur_cnt] = {0, lc, rc} ;
24            return cur_cnt ;
25        }
26
27        public:
28        PersistentSegmentTree(int _n){
29            n = _n ;
30            cnt = 0 ;
31
32            int root = build(1, n) ;
33            version.push_back(root) ;
34        }
35
36        void update(int ver, int idx){
37            auto upd = [&](auto &&self, const int
38                cur, int L, int R){
39                int cur_cnt = cnt++ ;
40
41                if(L == R){
42                    seg[cur_cnt] = {seg[cur].sum + 1, 0,
43                        0} ;
44                    return cur_cnt ;
45                }
46
47                int M = (L + R) >> 1 ;
48                int lc = seg[cur].left ;
49                int rc = seg[cur].right ;
50
51                if(idx <= M) lc = self(self,
52                    seg[cur].left, L, M) ;
53                else rc = self(self, seg[cur].right,
54                    M+1, R) ;
55
56                seg[cur_cnt] = {seg[lc].sum +
57                    seg[rc].sum, lc, rc} ;
58
59                return cur_cnt ;
60            };
61
62            int root = upd(upd, version[ver], 1, n) ;
63            version.push_back(root) ;
64        }
65
66        int query(int verL, int verR, int k){
67            auto qry = [&](auto &&self, const int
68                cur_old, const int cur_new, int L,
69                int R){
70                if(L == R) return L ;
71
72                int old_l = seg[cur_old].left, old_r =
73                    seg[cur_old].right ;
74                int new_l = seg[cur_new].left, new_r =
75                    seg[cur_new].right ;
76
77                int dl = seg[new_l].sum -
78                    seg[old_l].sum ;
79                int dr = seg[new_r].sum -
80                    seg[old_r].sum ;
81
82                int M = (L + R) >> 1 ;
83
84            };
85
86        }
87
88    }
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116

```

```

71     if(dl >= k) return self(self, old_l,
72         new_l, L, M) ;
73     k -= dl ;
74     return self(self, old_r, new_r, M+1,
75         R) ;
76 }
77
78     int idx = qry(qry, version[verL-1],
79         version[verR], 1, n) ;
80     return idx ;
81 }
```

3.5 Trie

```

1 class TrieNode{
2 public:
3     set<int> end ;
4     TrieNode *next[26] ;
5
6     TrieNode(){
7         for ( int i=0 ; i<26 ; i++ ) next[i]
8             = nullptr ;
9     }
10
11    class Trie{
12 private:
13     int cnt ;
14     TrieNode *root ;
15 public:
16     Trie() : cnt(0) {
17         root = new TrieNode() ;
18     }
19
20     void insert(string &str, int n){
21         TrieNode* node = root ;
22         for ( auto s : str ){
23             int path = s - 'a' ;
24
25             if(node->next[path] == nullptr)
26                 node->next[path] = new
27                     TrieNode() ;
28             node = node->next[path] ;
29         }
30         node->end.insert(n) ;
31     }
32
33     void search(string &str){
34         TrieNode* node = root ;
35         for ( auto s : str ){
36             int path = s - 'a' ;
37             if(node->next[path] == nullptr)
38                 return ;
39             node = node->next[path] ;
40         }
41
42         int flg = 0 ;
43         for ( auto n : node->end ){
44             if(flg) cout << " " ;
45             else flg = 1 ;
46
47             cout << n ;
48         }
49
50     void clear(TrieNode* node) {
51         if (!node) return ;
52         for (int i = 0; i < 26; i++) {
53             if (node->next[i]) {
54                 clear(node->next[i]) ;
55             }
56             delete node ;
57         }
58     }
59     ~Trie(){
60         clear(root) ;
61     }
62 }
```

3.6 BIT 單修區查

```

1 // 單點修改 區間查詢
2 #define lowbit(x) (x & -x)
3
4 int BIT[MAX_SIZE] ;
5 int n ;
6
7 void modify(int idx, int val){
8     for ( ; idx <= n ; idx += lowbit(idx) ){
9         BIT[idx] += val ;
10    }
11 }
12
13 // ans: query(R) - query(L-1)
14 int query(int idx){
15     int sum = 0 ;
16     for ( ; idx ; idx -= lowbit(idx) ){
17         sum += BIT[idx] ;
18     }
19 }
20
21 void init(){
22     memset(BIT, 0, sizeof(BIT)) ;
23 }
```

3.7 BIT 區修單查

```

1 // 區間修改， 單點查詢
2 #define lowbit(x) (x & -x)
3
4 int BIT[MAX_SIZE] ;
5 int n ;
6
7 void modify(int idx, int val){
8     for ( ; idx <= n ; idx += lowbit(idx) ){
9         BIT[idx] += val ;
10    }
11 }
12
13 // ans: query(i)
14 int query(int idx){
15     int sum = 0 ;
16     for ( ; idx ; idx -= lowbit(idx) ){
17         sum += BIT[idx] ;
18     }
19     return sum ;
20 }
21
22 void init(){
23     memset(BIT, 0, sizeof(BIT)) ;
24 }
25
26 void build(){
27     arr[0] = 0 ;
28     for ( int i=1 ; i<=n ; i++ ) modify(i,
29             arr[i] - arr[i-1]) ;
30 }
31
32 // usage
33 // add val
34 modify(L, x) ;
35 modify(R+1, -x) ;
```

3.8 BIT 區修區查

```

1 // 區間修改 區間查詢
2 #define lowbit(x) x & -x
3
4 ll BIT1[MAXN], BIT2[MAXN] ;
5
6 void update(int idx, ll val){
7     for ( int i=idx ; i<=n ; i+=lowbit(i) ){
8         BIT1[i] += val ;
9         BIT2[i] += idx * val ;
10    }
11 }
12
13 // range query: query(R) - query(X - 1)
14 ll query(int idx){
15     ll sum1 = 0, sum2 = 0 ;
16     for ( int i=idx ; i ; i-=lowbit(i) ){
17         sum1 += BIT1[i] ;
18         sum2 += BIT2[i] ;
19     }
20     return (idx + 1) * sum1 - sum2 ;
21 }
22
23 void build(){
24     for ( int i=1 ; i<=n ; i++ ){
25         update(i, arr[i] - arr[i-1]) ;
26     }
27 }
28
29 void usage(){
30     update(L, x) ;
31     update(R+1, -x) ;
32 }
33
34 void init(){
35     memset(BIT1, 0, sizeof(BIT1)) ;
36     memset(BIT2, 0, sizeof(BIT2)) ;
37 }
```

4 Graph

4.1 cut vertex AND bridges

```

1 int dfn[Maxn] = {-1}, low[Maxn] = {-1},
2     dfscnt ;
3
4 void dfs(int u, int fa){
5     dfn[u] = low[u] = ++dfscnt ;
6     int child = 0 ;
7
8     for ( auto v : g[u] ) if(v != fa){
9         if(dfn[v] == -1){
10             child++ ;
11             dfs(v, u) ;
12             low[u] = min(low[u], low[v]) ;
13
14             if(low[v] >= dfn[u]){
15                 // this edge is a bridge
16             }
17
18             if(u != fa && low[v] >= dfn[u]){
19                 // this node v is a articulation point
20             }
21         }
22     }
23
24     if(u == fa && child > 1){
25         // this node u is a articulation point
26     }
27 }
```

4.2 SCC - Tarjan

```

1 vector<int> scc[Maxn] ;
2 int dfn[Maxn], low[Maxn], sccId[Maxn],
3     dfscnt = 0, cnt_scc = 0 ;
4 stack<int> st ;
5 bitset<Maxn> inSt, vis ;
6
7 void dfs(int u, int from){
8     dfn[u] = low[u] = ++dfscnt ;
9     st.push(u) ;
10    inSt[u] = 1 ;
11
12    for ( auto v : g[u] ){
13        if(!inSt[v] && dfn[v] != -1) continue ;
14        if(dfn[v] == -1) dfs(v, u) ;
15        low[u] = min(low[u], low[v]) ;
16    }
17
18    if(dfn[u] == low[u]){
19        cnt_scc++ ;
20        int x ;
21
22        do{
23            x = st.top() ;
24            st.pop() ;
25
26            inSt[x] = 0 ;
27            sccId[x] = cnt_scc ;
28            scc[cnt_scc].push_back(x) ;
29        }
30        while(x != u) ;
31    }
32
33 // SCC to DAG (after dfs)
34 vector<int> dag[Maxn] ;
35
36 void scc_to_dag(){
37     vector<int> dag[Maxn] ;
38     for ( int u=1 ; u<=n ; u++ ){
39         for ( auto v : g[u] ){
40             if(sccId[u] != sccId[v]){
41                 dag[sccId[u]].push_back(sccId[v]) ;
42             }
43         }
44     }
45
46     void init(){
47         memset(dfn, -1, sizeof(dfn)) ;
48         memset(low, -1, sizeof(low)) ;
49     }
50
51     int main(){
52         init() ;
53         input() ;
54         for ( int i=1 ; i<=n ; i++ ) if(dfn[i]
55             == -1){
56             dfs(i, i) ;
57         }
58     }

```

4.3 BCC - Tarjan

```

1 struct Edge{
2     int v, next ;
3 }e[Maxm << 1] ;
4 int head[Maxm], tot = 1 ;
5
6 void add(int u, int v){
7     e[++tot] = {v, head[u]} ;
8     head[u] = tot ;
9     e[++tot] = {u, head[v]} ;
10    head[v] = tot ;
11}

```

```

12
13 bitset<Maxm << 1> bz ;
14 vector<vector<int>> bcc ;
15 int dfn_cnt = 0, dfn[Maxn], low[Maxn],
16     vis_bcc[Maxn], bcc_cnt = 0 ;
17
18 void dfs1(int u, int from){
19     dfn[u] = low[u] = ++dfn_cnt ;
20
21     for ( int i=head[u] ; i!= -1 ; i=e[i].next
22         ){
23         int v = e[i].v ;
24
25         if(dfn[v] == -1){
26             dfs1(v, i) ;
27             if(dfn[u] < low[v]) bz[i] = bz[i^1] =
28                 1 ;
29             low[u] = min(low[u], low[v]) ;
30         }
31         else if(i != (from ^ 1)) low[u] =
32             min(low[u], dfn[v]) ;
33     }
34
35 void dfs2(int u, int id){
36     vis_bcc[u] = id ;
37     bcc[id].push_back(u) ;
38
39     for ( int i=head[u] ; i!= -1 ; i=e[i].next
40         ){
41         int v = e[i].v ;
42
43         if(vis_bcc[v] != -1 || bz[i]) continue ;
44         dfs2(v, id) ;
45     }
46
47 void init(){
48     memset(dfn, -1, sizeof(dfn)) ;
49     memset(head, -1, sizeof(head)) ;
50     memset(vis_bcc, -1, sizeof(vis_bcc)) ;
51
52     int main(){
53         init() ;
54         input() ;
55         for ( int i=1 ; i<=n ; i++ ) if(dfn[i]
56             == -1){
57             dfs1(i, 0) ;
58         }
59
60         for ( int i=1 ; i<=n ; i++ ) if(vis_bcc[i]
61             == -1){
62             bcc.push_back(vector<int>()) ;
63             dfs2(i, bcc_cnt++) ;
64         }
65     }

```

4.4 Convex

```

1 struct Coordinate{
2     long long x, y ;
3
4     friend bool operator<(const Coordinate&a,
5                             const Coordinate&b){
6         if(a.x == b.x) return a.y < b.y ;
7         return a.x < b.x ;
8     }
9
10    friend bool operator==(const Coordinate&
11                           a, const Coordinate&b){
12        return a.x == b.x && a.y == b.y ;
13    }
14
15    vector<Coordinate> nodes ;
16
17}

```

```

16 long long cross(const Coordinate& o, const
17                  Coordinate& a, const Coordinate& b){
18     return (a.x - o.x) * (b.y - o.y) - (a.y -
19                                         o.y) * (b.x - o.x) ;
20 }
21
22 void input(){
23     nodes.clear() ;
24
25     int n, x, y ;
26     char c ;
27
28     for ( int i=0 ; i<n ; i++ ){
29         cin >> x >> y >> c ;
30         if(c == 'Y') nodes.push_back({x, y}) ;
31     }
32
33 void monotone(){
34     sort(nodes.begin(), nodes.end()) ;
35
36     int n = unique(nodes.begin(), nodes.end())
37         - nodes.begin() ;
38
39     vector<Coordinate> ch(n+1) ;
40
41     int m = 0 ;
42
43     for ( int i=0 ; i<n ; i++ ){
44         while(m > 1 && cross(ch[m-2], ch[m-1],
45                               nodes[i]) < 0) m-- ;
46         ch[m++] = nodes[i] ;
47
48     for ( int i=n-2, t=m ; i>=0 ; i-- ){
49         while(m > t && cross(ch[m-2], ch[m-1],
50                               nodes[i]) < 0) m-- ;
51         ch[m++] = nodes[i] ;
52
53     if(n > 1) m-- ;
54     cout << m << endl ;
55 }
56
57 for ( int i=0 ; i<m ; i++ ) cout <<
58     ch[i].x << " " << ch[i].y << endl ;
59 }

```

4.5 Max Flow

```

1 struct Edge{
2     int v, cap, next ;
3 };
4
5 class MaxFlow{
6 private:
7     int N, S, T ;
8     vector<Edge> e ;
9     vector<int> head, cur, dep ;
10
11    bool bfs(){
12        queue<int> q ;
13        for ( int i=0 ; i<=N ; i++ ){
14            cur[i] = head[i] ;
15            dep[i] = -1 ;
16        }
17
18        q.push(S) ;
19        dep[S] = 0 ;
20
21        while(!q.empty()){
22            int u = q.front() ; q.pop() ;
23
24            for ( int i=head[u] ; i!= -1 ;
25                  i=e[i].next ){
26                int v = e[i].v ;
27                if(dep[v] == -1 && e[i].cap > 0){
28                    dep[v] = dep[u] + 1 ;
29                }
30            }
31        }
32    }
33
34    int dfs(int u, int t, int f){
35        if(u == t) return f ;
36
37        for ( int i=cur[u] ; i!= -1 ;
38              i=e[i].next ){
39            int v = e[i].v ;
40
41            if(dep[v] == dep[u] + 1 && e[i].cap > 0){
42                int d = dfs(v, t, min(f, e[i].cap)) ;
43
44                if(d > 0){
45                    e[i].cap -= d ;
46                    e[i^1].cap += d ;
47
48                    cur[u] = i ;
49                    return d ;
50                }
51            }
52        }
53
54        return 0 ;
55    }
56
57    int maxflow(){
58        int f = 0 ;
59
60        while(bfs()){
61            f += dfs(S, T, INT_MAX) ;
62        }
63
64        return f ;
65    }
66
67    void print(){
68        for ( int i=0 ; i<=N ; i++ )
69            cout << i << " : " << cur[i] << endl ;
70
71        for ( int i=0 ; i<=N ; i++ )
72            cout << i << " : " << dep[i] << endl ;
73
74        for ( int i=0 ; i<=N ; i++ )
75            cout << i << " : " << head[i] << endl ;
76
77        for ( int i=0 ; i<=N ; i++ )
78            cout << i << " : " << cur[i] << endl ;
79
80        for ( int i=0 ; i<=N ; i++ )
81            cout << i << " : " << dep[i] << endl ;
82
83        for ( int i=0 ; i<=N ; i++ )
84            cout << i << " : " << head[i] << endl ;
85
86        for ( int i=0 ; i<=N ; i++ )
87            cout << i << " : " << cur[i] << endl ;
88
89        for ( int i=0 ; i<=N ; i++ )
90            cout << i << " : " << dep[i] << endl ;
91
92        for ( int i=0 ; i<=N ; i++ )
93            cout << i << " : " << head[i] << endl ;
94
95        for ( int i=0 ; i<=N ; i++ )
96            cout << i << " : " << cur[i] << endl ;
97
98        for ( int i=0 ; i<=N ; i++ )
99            cout << i << " : " << dep[i] << endl ;
100
101        for ( int i=0 ; i<=N ; i++ )
102            cout << i << " : " << head[i] << endl ;
103
104        for ( int i=0 ; i<=N ; i++ )
105            cout << i << " : " << cur[i] << endl ;
106
107        for ( int i=0 ; i<=N ; i++ )
108            cout << i << " : " << dep[i] << endl ;
109
110        for ( int i=0 ; i<=N ; i++ )
111            cout << i << " : " << head[i] << endl ;
112
113        for ( int i=0 ; i<=N ; i++ )
114            cout << i << " : " << cur[i] << endl ;
115
116        for ( int i=0 ; i<=N ; i++ )
117            cout << i << " : " << dep[i] << endl ;
118
119        for ( int i=0 ; i<=N ; i++ )
120            cout << i << " : " << head[i] << endl ;
121
122        for ( int i=0 ; i<=N ; i++ )
123            cout << i << " : " << cur[i] << endl ;
124
125        for ( int i=0 ; i<=N ; i++ )
126            cout << i << " : " << dep[i] << endl ;
127
128        for ( int i=0 ; i<=N ; i++ )
129            cout << i << " : " << head[i] << endl ;
130
131        for ( int i=0 ; i<=N ; i++ )
132            cout << i << " : " << cur[i] << endl ;
133
134        for ( int i=0 ; i<=N ; i++ )
135            cout << i << " : " << dep[i] << endl ;
136
137        for ( int i=0 ; i<=N ; i++ )
138            cout << i << " : " << head[i] << endl ;
139
140        for ( int i=0 ; i<=N ; i++ )
141            cout << i << " : " << cur[i] << endl ;
142
143        for ( int i=0 ; i<=N ; i++ )
144            cout << i << " : " << dep[i] << endl ;
145
146        for ( int i=0 ; i<=N ; i++ )
147            cout << i << " : " << head[i] << endl ;
148
149        for ( int i=0 ; i<=N ; i++ )
150            cout << i << " : " << cur[i] << endl ;
151
152        for ( int i=0 ; i<=N ; i++ )
153            cout << i << " : " << dep[i] << endl ;
154
155        for ( int i=0 ; i<=N ; i++ )
156            cout << i << " : " << head[i] << endl ;
157
158        for ( int i=0 ; i<=N ; i++ )
159            cout << i << " : " << cur[i] << endl ;
160
161        for ( int i=0 ; i<=N ; i++ )
162            cout << i << " : " << dep[i] << endl ;
163
164        for ( int i=0 ; i<=N ; i++ )
165            cout << i << " : " << head[i] << endl ;
166
167        for ( int i=0 ; i<=N ; i++ )
168            cout << i << " : " << cur[i] << endl ;
169
170        for ( int i=0 ; i<=N ; i++ )
171            cout << i << " : " << dep[i] << endl ;
172
173        for ( int i=0 ; i<=N ; i++ )
174            cout << i << " : " << head[i] << endl ;
175
176        for ( int i=0 ; i<=N ; i++ )
177            cout << i << " : " << cur[i] << endl ;
178
179        for ( int i=0 ; i<=N ; i++ )
180            cout << i << " : " << dep[i] << endl ;
181
182        for ( int i=0 ; i<=N ; i++ )
183            cout << i << " : " << head[i] << endl ;
184
185        for ( int i=0 ; i<=N ; i++ )
186            cout << i << " : " << cur[i] << endl ;
187
188        for ( int i=0 ; i<=N ; i++ )
189            cout << i << " : " << dep[i] << endl ;
190
191        for ( int i=0 ; i<=N ; i++ )
192            cout << i << " : " << head[i] << endl ;
193
194        for ( int i=0 ; i<=N ; i++ )
195            cout << i << " : " << cur[i] << endl ;
196
197        for ( int i=0 ; i<=N ; i++ )
198            cout << i << " : " << dep[i] << endl ;
199
200        for ( int i=0 ; i<=N ; i++ )
201            cout << i << " : " << head[i] << endl ;
202
203        for ( int i=0 ; i<=N ; i++ )
204            cout << i << " : " << cur[i] << endl ;
205
206        for ( int i=0 ; i<=N ; i++ )
207            cout << i << " : " << dep[i] << endl ;
208
209        for ( int i=0 ; i<=N ; i++ )
210            cout << i << " : " << head[i] << endl ;
211
212        for ( int i=0 ; i<=N ; i++ )
213            cout << i << " : " << cur[i] << endl ;
214
215        for ( int i=0 ; i<=N ; i++ )
216            cout << i << " : " << dep[i] << endl ;
217
218        for ( int i=0 ; i<=N ; i++ )
219            cout << i << " : " << head[i] << endl ;
220
221        for ( int i=0 ; i<=N ; i++ )
222            cout << i << " : " << cur[i] << endl ;
223
224        for ( int i=0 ; i<=N ; i++ )
225            cout << i << " : " << dep[i] << endl ;
226
227        for ( int i=0 ; i<=N ; i++ )
228            cout << i << " : " << head[i] << endl ;
229
230        for ( int i=0 ; i<=N ; i++ )
231            cout << i << " : " << cur[i] << endl ;
232
233        for ( int i=0 ; i<=N ; i++ )
234            cout << i << " : " << dep[i] << endl ;
235
236        for ( int i=0 ; i<=N ; i++ )
237            cout << i << " : " << head[i] << endl ;
238
239        for ( int i=0 ; i<=N ; i++ )
240            cout << i << " : " << cur[i] << endl ;
241
242        for ( int i=0 ; i<=N ; i++ )
243            cout << i << " : " << dep[i] << endl ;
244
245        for ( int i=0 ; i<=N ; i++ )
246            cout << i << " : " << head[i] << endl ;
247
248        for ( int i=0 ; i<=N ; i++ )
249            cout << i << " : " << cur[i] << endl ;
250
251        for ( int i=0 ; i<=N ; i++ )
252            cout << i << " : " << dep[i] << endl ;
253
254        for ( int i=0 ; i<=N ; i++ )
255            cout << i << " : " << head[i] << endl ;
256
257        for ( int i=0 ; i<=N ; i++ )
258            cout << i << " : " << cur[i] << endl ;
259
260        for ( int i=0 ; i<=N ; i++ )
261            cout << i << " : " << dep[i] << endl ;
262
263        for ( int i=0 ; i<=N ; i++ )
264            cout << i << " : " << head[i] << endl ;
265
266        for ( int i=0 ; i<=N ; i++ )
267            cout << i << " : " << cur[i] << endl ;
268
269        for ( int i=0 ; i<=N ; i++ )
270            cout << i << " : " << dep[i] << endl ;
271
272        for ( int i=0 ; i<=N ; i++ )
273            cout << i << " : " << head[i] << endl ;
274
275        for ( int i=0 ; i<=N ; i++ )
276            cout << i << " : " << cur[i] << endl ;
277
278        for ( int i=0 ; i<=N ; i++ )
279            cout << i << " : " << dep[i] << endl ;
280
281        for ( int i=0 ; i<=N ; i++ )
282            cout << i << " : " << head[i] << endl ;
283
284        for ( int i=0 ; i<=N ; i++ )
285            cout << i << " : " << cur[i] << endl ;
286
287        for ( int i=0 ; i<=N ; i++ )
288            cout << i << " : " << dep[i] << endl ;
289
290        for ( int i=0 ; i<=N ; i++ )
291            cout << i << " : " << head[i] << endl ;
292
293        for ( int i=0 ; i<=N ; i++ )
294            cout << i << " : " << cur[i] << endl ;
295
296        for ( int i=0 ; i<=N ; i++ )
297            cout << i << " : " << dep[i] << endl ;
298
299        for ( int i=0 ; i<=N ; i++ )
300            cout << i << " : " << head[i] << endl ;
301
302        for ( int i=0 ; i<=N ; i++ )
303            cout << i << " : " << cur[i] << endl ;
304
305        for ( int i=0 ; i<=N ; i++ )
306            cout << i << " : " << dep[i] << endl ;
307
308        for ( int i=0 ; i<=N ; i++ )
309            cout << i << " : " << head[i] << endl ;
310
311        for ( int i=0 ; i<=N ; i++ )
312            cout << i << " : " << cur[i] << endl ;
313
314        for ( int i=0 ; i<=N ; i++ )
315            cout << i << " : " << dep[i] << endl ;
316
317        for ( int i=0 ; i<=N ; i++ )
318            cout << i << " : " << head[i] << endl ;
319
320        for ( int i=0 ; i<=N ; i++ )
321            cout << i << " : " << cur[i] << endl ;
322
323        for ( int i=0 ; i<=N ; i++ )
324            cout << i << " : " << dep[i] << endl ;
325
326        for ( int i=0 ; i<=N ; i++ )
327            cout << i << " : " << head[i] << endl ;
328
329        for ( int i=0 ; i<=N ; i++ )
330            cout << i << " : " << cur[i] << endl ;
331
332        for ( int i=0 ; i<=N ; i++ )
333            cout << i << " : " << dep[i] << endl ;
334
335        for ( int i=0 ; i<=N ; i++ )
336            cout << i << " : " << head[i] << endl ;
337
338        for ( int i=0 ; i<=N ; i++ )
339            cout << i << " : " << cur[i] << endl ;
340
341        for ( int i=0 ; i<=N ; i++ )
342            cout << i << " : " << dep[i] << endl ;
343
344        for ( int i=0 ; i<=N ; i++ )
345            cout << i << " : " << head[i] << endl ;
346
347        for ( int i=0 ; i<=N ; i++ )
348            cout << i << " : " << cur[i] << endl ;
349
350        for ( int i=0 ; i<=N ; i++ )
351            cout << i << " : " << dep[i] << endl ;
352
353        for ( int i=0 ; i<=N ; i++ )
354            cout << i << " : " << head[i] << endl ;
355
356        for ( int i=0 ; i<=N ; i++ )
357            cout << i << " : " << cur[i] << endl ;
358
359        for ( int i=0 ; i<=N ; i++ )
360            cout << i << " : " << dep[i] << endl ;
361
362        for ( int i=0 ; i<=N ; i++ )
363            cout << i << " : " << head[i] << endl ;
364
365        for ( int i=0 ; i<=N ; i++ )
366            cout << i << " : " << cur[i] << endl ;
367
368        for ( int i=0 ; i<=N ; i++ )
369            cout << i << " : " << dep[i] << endl ;
370
371        for ( int i=0 ; i<=N ; i++ )
372            cout << i << " : " << head[i] << endl ;
373
374        for ( int i=0 ; i<=N ; i++ )
375            cout << i << " : " << cur[i] << endl ;
376
377        for ( int i=0 ; i<=N ; i++ )
378            cout << i << " : " << dep[i] << endl ;
379
380        for ( int i=0 ; i<=N ; i++ )
381            cout << i << " : " << head[i] << endl ;
382
383        for ( int i=0 ; i<=N ; i++ )
384            cout << i << " : " << cur[i] << endl ;
385
386        for ( int i=0 ; i<=N ; i++ )
387            cout << i << " : " << dep[i] << endl ;
388
389        for ( int i=0 ; i<=N ; i++ )
390            cout << i << " : " << head[i] << endl ;
391
392        for ( int i=0 ; i<=N ; i++ )
393            cout << i << " : " << cur[i] << endl ;
394
395        for ( int i=0 ; i<=N ; i++ )
396            cout << i << " : " << dep[i] << endl ;
397
398        for ( int i=0 ; i<=N ; i++ )
399            cout << i << " : " << head[i] << endl ;
400
401        for ( int i=0 ; i<=N ; i++ )
402            cout << i << " : " << cur[i] << endl ;
403
404        for ( int i=0 ; i<=N ; i++ )
405            cout << i << " : " << dep[i] << endl ;
406
407        for ( int i=0 ; i<=N ; i++ )
408            cout << i << " : " << head[i] << endl ;
409
410        for ( int i=0 ; i<=N ; i++ )
411            cout << i << " : " << cur[i] << endl ;
412
413        for ( int i=0 ; i<=N ; i++ )
414            cout << i << " : " << dep[i] << endl ;
415
416        for ( int i=0 ; i<=N ; i++ )
417            cout << i << " : " << head[i] << endl ;
418
419        for ( int i=0 ; i<=N ; i++ )
420            cout << i << " : " << cur[i] << endl ;
421
422        for ( int i=0 ; i<=N ; i++ )
423            cout << i << " : " << dep[i] << endl ;
424
425        for ( int i=0 ; i<=N ; i++ )
426            cout << i << " : " << head[i] << endl ;
427
428        for ( int i=0 ; i<=N ; i++ )
429            cout << i << " : " << cur[i] << endl ;
430
431        for ( int i=0 ; i<=N ; i++ )
432            cout << i << " : " << dep[i] << endl ;
433
434        for ( int i=0 ; i<=N ; i++ )
435            cout << i << " : " << head[i] << endl ;
436
437        for ( int i=0 ; i<=N ; i++ )
438            cout << i << " : " << cur[i] << endl ;
439
440        for ( int i=0 ; i<=N ; i++ )
441            cout << i << " : " << dep[i] << endl ;
442
443        for ( int i=0 ; i<=N ; i++ )
444            cout << i << " : " << head[i] << endl ;
445
446        for ( int i=0 ; i<=N ; i++ )
447            cout << i << " : " << cur[i] << endl ;
448
449        for ( int i=0 ; i<=N ; i++ )
450            cout << i << " : " << dep[i] << endl ;
451
452        for ( int i=0 ; i<=N ; i++ )
453            cout << i << " : " << head[i] << endl ;
454
455        for ( int i=0 ; i<=N ; i++ )
456            cout << i << " : " << cur[i] << endl ;
457
458        for ( int i=0 ; i<=N ; i++ )
459            cout << i << " : " << dep[i] << endl ;
460
461        for ( int i=0 ; i<=N ; i++ )
462            cout << i << " : " << head[i] << endl ;
463
464        for ( int i=0 ; i<=N ; i++ )
465            cout << i << " : " << cur[i] << endl ;
466
467        for ( int i=0 ; i<=N ; i++ )
468            cout << i << " : " << dep[i] << endl ;
469
470        for ( int i=0 ; i<=N ; i++ )
471            cout << i << " : " << head[i] << endl ;
472
473        for ( int i=0 ; i<=N ; i++ )
474            cout << i << " : " << cur[i] << endl ;
475
476        for ( int i=0 ; i<=N ; i++ )
477            cout << i << " : " << dep[i] << endl ;
478
479        for ( int i=0 ; i<=N ; i++ )
480            cout << i << " : " << head[i] << endl ;
481
482        for ( int i=0 ; i<=N ; i++ )
483            cout << i << " : " << cur[i] << endl ;
484
485        for ( int i=0 ; i<=N ; i++ )
486            cout << i << " : " << dep[i] << endl ;
487
488        for ( int i=0 ; i<=N ; i++ )
489            cout << i << " : " << head[i] << endl ;
490
491        for ( int i=0 ; i<=N ; i++ )
492            cout << i << " : " << cur[i] << endl ;
493
494        for ( int i=0 ; i<=N ; i++ )
495            cout << i << " : " << dep[i] << endl ;
496
497        for ( int i=0 ; i<=N ; i++ )
498            cout << i << " : " << head[i] << endl ;
499
500        for ( int i=0 ; i<=N ; i++ )
501            cout << i << " : " << cur[i] << endl ;
502
503        for ( int i=0 ; i<=N ; i++ )
504            cout << i << " : " << dep[i] << endl ;
505
506        for ( int i=0 ; i<=N ; i++ )
507            cout << i << " : " << head[i] << endl ;
508
509        for ( int i=0 ; i<=N ; i++ )
510            cout << i << " : " << cur[i] << endl ;
511
512        for ( int i=0 ; i<=N ; i++ )
513            cout << i <&lt
```

```

28     if(v == T) return 1 ;
29     q.push(v) ;
30   }
31 }
32 }
33 return 0 ;
34 }
35 }

36 int dfs(int u, int flow){
37   if(u == T) return flow ;
38   int d, rest = 0 ;
39
40   for (int &i=cur[u] ; i!=-1 ;
41     i=e[i].next ){
42     int v = e[i].v ;
43     if(dep[v] == dep[u] + 1 && e[i].cap >
44       0){
45       d = dfs(v, min(flow - rest,
46                 e[i].cap)) ;
47
48     if(d > 0){
49       e[i].cap -= d ;
50       e[i^1].cap += d ;
51       rest += d ;
52
53     if(rest == flow) break ;
54   }
55
56   if(rest != flow) dep[u] = -1 ;
57   return rest ;
58 }
59 public:
60 MaxFlow(int n, int s, int t){
61   N = n ; S = s ; T = t ;
62   e.reserve(n*n) ;
63   head.assign(n+1, -1) ;
64   cur.resize(n+1) ;
65   dep.resize(n+1) ;
66 }
67 void AddEdge(int u, int v, int cap){
68   e.push_back({v, cap, head[u]}) ;
69   head[u] = e.size() - 1 ;
70   e.push_back({u, 0, head[v]}) ;
71   head[v] = e.size() - 1 ;
72 }
73
74 int run(){
75   int ans = 0 ;
76   while(bfs()){
77     ans += dfs(S, 0x3f3f3f3f) ;
78   }
79   return ans ;
80 }
81 }
82 }
```

4.6 min cut max flow

```

1 struct Edge{
2   int v, cap, cost , next ;
3 };
4
5 using pii = pair<int, int> ;
6 class MCMF{
7 private:
8   int N, s, t, tot ;
9   vector<Edge> e ;
10  vector<int> head ;
11 public:
12  MCMF(int n, int _s, int _t){
13    N = n ;
14    s = _s ;
15    t = _t ;
16    e.resize(n*n + 5) ;
```

```

17   head.assign(n+5, -1) ;
18   tot = -1 ;
19 }
20
21 void AddEdge(int u, int v, int cap, int
22   cost){
23   e[++tot] = {v, cap, cost, head[u]} ;
24   head[u] = tot ;
25   e[++tot] = {u, 0, -cost, head[v]} ;
26   head[v] = tot ;
27 }
28
29 int run(){
30   vector<int> dis(N+1), pot(N+1, 0),
31     preE(N+1) ;
32   int flow = 0, cost = 0 ;
33
34   auto dijkstra = [&](){
35     fill(dis.begin(), dis.end(), INF) ;
36     priority_queue<pii, vector<pii>,
37       greater<pii>> pq ;
38     dis[s] = 0 ;
39     pq.push({0, s}) ;
40
41     while(!pq.empty()){
42       auto [d, u] = pq.top() ; pq.pop() ;
43       if(d > dis[u]) continue ;
44       for (int i=head[u] ; i!=-1 ;
45         i=e[i].next ){
46         int v = e[i].v, cap = e[i].cap, w =
47           e[i].cost ;
48         if(cap && dis[v] > d + w + pot[u] -
49           pot[v]){
50           dis[v] = d + w + pot[u] - pot[v] ;
51           preE[v] = i ;
52           pq.push({dis[v], v}) ;
53         }
54       }
55
56       return dis[t] != INF ;
57     };
58
59     while(dijkstra()){
60       for (int v=1; v<=N ; v++ ) if(dis[v]
61         < INF){
62         pot[v] += dis[v] ;
63       }
64
65       int aug = INT_MAX ;
66       for (int v=t ; v!=s ;
67         v=e[preE[v]^1].v ){
68         aug = min(aug, e[preE[v]].cap) ;
69
70         for (int v=t ; v!=s ;
71           v=e[preE[v]^1].v ){
72           e[preE[v]].cap -= aug ;
73           e[preE[v]^1].cap += aug ;
74           cost += aug * e[preE[v]].cost ;
75         }
76
77         return cost ;
78       }
79     }
80   }
81 }
```

```

17   head.assign(n+5, -1) ;
18   tot = -1 ;
19 }
20
21 void AddEdge(int u, int v, int cap, int
22   cost){
23   e[++tot] = {v, cap, cost, head[u]} ;
24   head[u] = tot ;
25   e[++tot] = {u, 0, -cost, head[v]} ;
26   head[v] = tot ;
27 }
28
29 int run(){
30   vector<int> dis(N+1), pot(N+1, 0),
31     preE(N+1) ;
32   int flow = 0, cost = 0 ;
33
34   auto dijkstra = [&](){
35     fill(dis.begin(), dis.end(), INF) ;
36     priority_queue<pii, vector<pii>,
37       greater<pii>> pq ;
38     dis[s] = 0 ;
39     pq.push({0, s}) ;
40
41     while(!pq.empty()){
42       auto [d, u] = pq.top() ; pq.pop() ;
43       if(d > dis[u]) continue ;
44       for (int i=head[u] ; i!=-1 ;
45         i=e[i].next ){
46         int v = e[i].v, cap = e[i].cap, w =
47           e[i].cost ;
48         if(cap && dis[v] > d + w + pot[u] -
49           pot[v]){
50           dis[v] = d + w + pot[u] - pot[v] ;
51           preE[v] = i ;
52           pq.push({dis[v], v}) ;
53         }
54       }
55
56       return dis[t] != INF ;
57     };
58
59     while(dijkstra()){
60       for (int v=1; v<=N ; v++ ) if(dis[v]
61         < INF){
62         pot[v] += dis[v] ;
63       }
64
65       int aug = INT_MAX ;
66       for (int v=t ; v!=s ;
67         v=e[preE[v]^1].v ){
68         aug = min(aug, e[preE[v]].cap) ;
69
70         for (int v=t ; v!=s ;
71           v=e[preE[v]^1].v ){
72           e[preE[v]].cap -= aug ;
73           e[preE[v]^1].cap += aug ;
74           cost += aug * e[preE[v]].cost ;
75         }
76
77         return cost ;
78       }
79     }
80   }
81 }
```

5 String

5.1 KMP

```

1 int Next[N] ;
2 void kmp(string &str){
3   Next[0] = -1 ;
4   if(str.size() <= 1) return ;
5   Next[1] = 0 ;
```

```

6
7   int cur = 2, check = 0 ;
8
9   while(cur < str.size()){
10     if(str[cur - 1] == str[check])
11       Next[cur++] = ++check ;
12     else if(check > 0) check =
13       Next[check] ;
14   }
15
16 int main(){
17   ios::sync_with_stdio(false) ;
18   cin.tie(nullptr) ;
19   cout.tie(nullptr) ;
20
21   string s1, s2 ;
22   while(cin >> s1){
23     s2 = s1 ;
24     reverse(s2.begin(), s2.end()) ;
25     kmp(s2) ;
26
27     int x=0, y=0 ;
28     while(x < s1.size() && y < s2.size()){
29       if(s1[x] == s2[y]){
30         x++ ;
31         y++ ;
32       }
33       else if(y > 0) y = Next[y] ;
34       else x++ ;
35     }
36
37     cout << s1 << s2.substr(y) << endl ;
38   }
39
40   return 0 ;
41 }
```

5.2 ACAM

```

1 class ACAutomaton{
2 private:
3   vector<int> fail, end, order ;
4   vector<vector<int>> tree ;
5
6   int base, alpha ;
7
8   int new_node(){
9     tree.emplace_back(alpha, 0) ;
10    fail.push_back(0) ;
11
12    return tree.size() - 1 ;
13  }
14 public:
15  ACAutomaton(int _base='a', int _alpha=26)
16    : base(_base), alpha(_alpha) {
17    clear() ;
18  }
19
20  void clear(){
21    fail.assign(1, 0) ;
22    order.clear() ;
23    end.clear() ;
24    tree.assign(1, vector<int>(alpha, 0)) ;
25  }
26
27  void add_pattern(const string &pattern){
28    int u = 0 ;
29    for (auto ch : pattern ){
30      int v = ch - base ;
31
32      if(tree[u][v] == 0) tree[u][v] =
33        new_node() ;
34      u = tree[u][v] ;
35    }
```

```

36     end.push_back(u) ;
37 }
38
39 void build(){
40     queue<int> q;
41     order.clear();
42     order.push_back(0);
43
44     for ( int i=0 ; i<alpha ; i++ )
45         if(tree[0][i] > 0){
46             q.push(tree[0][i]);
47         }
48
49     while(!q.empty()){
50         int u = q.front(); q.pop();
51         order.push_back(u);
52
53         for ( int i=0 ; i<alpha ; i++ ){
54             if(tree[u][i] == 0) tree[u][i] =
55                 tree[fail[u]][i];
56             else{
57                 fail[tree[u][i]] = tree[fail[u]][i];
58                 ;
59                 q.push(tree[u][i]);
60             }
61         }
62
63     vector<int> count_per_pattern(const string
64         &text) const {
65         int u = 0;
66         vector<int> vis(tree.size(), 0);
67
68         for ( char ch : text ){
69             u = tree[u][ch - base];
70             vis[u]++;
71         }
72
73         for ( int i=order.size()-1 ; i>=1 ; i-- )
74             int x = order[i];
75             vis[fail[x]] += vis[x];
76
77         vector<int> ans(end.size(), 0);
78         for ( int id=0 ; id<end.size() ; id++ ){
79             ans[id] = vis[end[id]];
80         }
81
82         return ans;
83     };

```

6 Techniques

6.1 二分搜

```

1 // xxxxxooo 找最小解
2 bool binary_search(){
3     while(l < r){
4         int m = (l + r) >> 1;
5         if(check(m)) r = m;
6         else l = m + 1;
7     }
7
8     return 1;
8
9
10}
11
12 // oooooxxx 找最大解
13 bool binary_search(){
14     while(l < r){
15         int m = (l + r) >> 1;
16         if(check(m)) l = m;
17         else r = m - 1;
18     }
19
20     return 1;
21 }
22 // 如果l & r 太大, m = (l + (r - 1)) >> 1;

```

6.2 倍增 LCA

```

1 #include <bits/stdc++.h>
2
3 using namespace std;
4
5 const int Maxn = 500005;
6
7 vector<int> e[Maxn];
8 int depth[Maxn];
9 int up[Maxn][40];
10 int MaxLog;
11
12 void dfs(int u, int from, int d){
13     up[u][0] = from;
14     depth[u] = d;
15
16     for ( int i=1 ; i<=MaxLog ; i++ ){
17         up[u][i] = up[up[u][i - 1]][i - 1];
18     }
19
20     for ( auto v : e[u] ){
21         if(v == from) continue;
22         dfs(v, u, d + 1);
23     }
24 }
25
26 int lca(int u, int v){
27     if(depth[u] < depth[v]) swap(u, v);
28
29     for ( int i=MaxLog ; i>=0 ; i-- )
30         if(depth[u] - (1 << i) >= depth[v]){
31             u = up[u][i];
32         }
33
34     if(u == v) return u;
35
36     for ( int i=MaxLog ; i>=0 ; i-- )
37         if(up[u][i] != up[v][i]){
38             u = up[u][i];
39             v = up[v][i];
40         }
41
42     return up[u][0];
43 }
44
45 int main(){
46     int n, q, root;
47     scanf("%d%d%d", &n, &q, &root);
48     MaxLog = __lg(n);
49
50     for ( int i=0 ; i<n-1 ; i++ ){
51         int u, v;
52         scanf("%d%d", &u, &v);
53         e[u].push_back(v);
54         e[v].push_back(u);
55     }
56
57     dfs(root, root, 0);
58
59     while(q--){
60         int u, v;
61         scanf("%d%d", &u, &v);
62         printf("%d\n", lca(u, v));
63     }
64 }

```

6.3 SG

```

1 long long SG(long long k){
2
3     if(k % 2 == 0){ return k / 2; }
4     else{ return SG(k / 2); }
5 }
6
7 int main(){
8     int cas, n;
9
10    scanf("%d", &cas);
11    while(cas--){
12        scanf("%d", &n);
13
14        long long s, v = 0;
15
16        for(int i = 0; i < n; i++){
17            scanf("%lld", &s);
18            v ^= SG(s); //XOR
19        }
20
21        if(v) printf("YES\n");
22        else printf("NO\n");
23    }
24
25    int SG[30];
26    int vis[Maxn], stone[Maxn];
27
28    void build(){
29        SG[0] = 0;
30        memset(vis, 0, sizeof(vis));
31
32        for ( int i=1 ; i<30 ; i++ ){
33            int cur = 0;
34            for ( int j=0 ; j<i ; j++ ) for ( int
35                k=0 ; k<=j ; k++ ){
36                vis[SG[j] ^ SG[k]] = i;
37            }
38            while(vis[cur] == i) cur++;
39            SG[i] = cur;
40        }
41
42        int main(){
43            build();
44
45            int T = 0;
46            while(~scanf("%d", &n) && n){
47                int ans = 0;
48
49                for ( int i=1 ; i<=n ; i++ ) scanf("%d",
50                    &stone[i]);
51
52                for ( int i=1 ; i<=n ; i++ ) if(stone[i]
53                    & 1){
54                    ans ^= SG[n-i];
55                }
56            }
57        }
58
59        int T = 0;
60        while(~scanf("%d", &n) && n){
61            int ans = 0;
62
63            for ( int i=1 ; i<=n ; i++ ) if(stone[i]
64                & 1){
65                    ans ^= SG[n-i];
66                }
67        }
68    }
69
70    int main(){
71        build();
72
73        int T = 0;
74        while(~scanf("%d", &n) && n){
75            int ans = 0;
76
77            for ( int i=1 ; i<=n ; i++ ) scanf("%d",
78                &stone[i]);
79
80            for ( int i=1 ; i<=n ; i++ ) if(stone[i]
81                & 1){
82                ans ^= SG[n-i];
83            }
84        }
85    }
86
87    int T = 0;
88    while(~scanf("%d", &n) && n){
89        int ans = 0;
90
91        for ( int i=1 ; i<=n ; i++ ) scanf("%d",
92            &stone[i]);
93
94        for ( int i=1 ; i<=n ; i++ ) if(stone[i]
95            & 1){
96                ans ^= SG[n-i];
97            }
98        }
99    }
100
101    int T = 0;
102    while(~scanf("%d", &n) && n){
103        int ans = 0;
104
105        for ( int i=1 ; i<=n ; i++ ) scanf("%d",
106            &stone[i]);
107
108        for ( int i=1 ; i<=n ; i++ ) if(stone[i]
109            & 1){
110                ans ^= SG[n-i];
111            }
112        }
113    }
114
115    int T = 0;
116    while(~scanf("%d", &n) && n){
117        int ans = 0;
118
119        for ( int i=1 ; i<=n ; i++ ) scanf("%d",
120            &stone[i]);
121
122        for ( int i=1 ; i<=n ; i++ ) if(stone[i]
123            & 1){
124                ans ^= SG[n-i];
125            }
126        }
127    }
128
129    int T = 0;
130    while(~scanf("%d", &n) && n){
131        int ans = 0;
132
133        for ( int i=1 ; i<=n ; i++ ) scanf("%d",
134            &stone[i]);
135
136        for ( int i=1 ; i<=n ; i++ ) if(stone[i]
137            & 1){
138                ans ^= SG[n-i];
139            }
140        }
141    }
142
143    int T = 0;
144    while(~scanf("%d", &n) && n){
145        int ans = 0;
146
147        for ( int i=1 ; i<=n ; i++ ) scanf("%d",
148            &stone[i]);
149
150        for ( int i=1 ; i<=n ; i++ ) if(stone[i]
151            & 1){
152                ans ^= SG[n-i];
153            }
154        }
155    }
156
157    int T = 0;
158    while(~scanf("%d", &n) && n){
159        int ans = 0;
160
161        for ( int i=1 ; i<=n ; i++ ) scanf("%d",
162            &stone[i]);
163
164        for ( int i=1 ; i<=n ; i++ ) if(stone[i]
165            & 1){
166                ans ^= SG[n-i];
167            }
168        }
169    }
170
171    int T = 0;
172    while(~scanf("%d", &n) && n){
173        int ans = 0;
174
175        for ( int i=1 ; i<=n ; i++ ) scanf("%d",
176            &stone[i]);
177
178        for ( int i=1 ; i<=n ; i++ ) if(stone[i]
179            & 1){
180                ans ^= SG[n-i];
181            }
182        }
183    }
184
185    int T = 0;
186    while(~scanf("%d", &n) && n){
187        int ans = 0;
188
189        for ( int i=1 ; i<=n ; i++ ) scanf("%d",
190            &stone[i]);
191
192        for ( int i=1 ; i<=n ; i++ ) if(stone[i]
193            & 1){
194                ans ^= SG[n-i];
195            }
196        }
197    }
198
199    int T = 0;
200    while(~scanf("%d", &n) && n){
201        int ans = 0;
202
203        for ( int i=1 ; i<=n ; i++ ) scanf("%d",
204            &stone[i]);
205
206        for ( int i=1 ; i<=n ; i++ ) if(stone[i]
207            & 1){
208                ans ^= SG[n-i];
209            }
210        }
211    }
212
213    int T = 0;
214    while(~scanf("%d", &n) && n){
215        int ans = 0;
216
217        for ( int i=1 ; i<=n ; i++ ) scanf("%d",
218            &stone[i]);
219
220        for ( int i=1 ; i<=n ; i++ ) if(stone[i]
221            & 1){
222                ans ^= SG[n-i];
223            }
224        }
225    }
226
227    int T = 0;
228    while(~scanf("%d", &n) && n){
229        int ans = 0;
230
231        for ( int i=1 ; i<=n ; i++ ) scanf("%d",
232            &stone[i]);
233
234        for ( int i=1 ; i<=n ; i++ ) if(stone[i]
235            & 1){
236                ans ^= SG[n-i];
237            }
238        }
239    }
240
241    int T = 0;
242    while(~scanf("%d", &n) && n){
243        int ans = 0;
244
245        for ( int i=1 ; i<=n ; i++ ) scanf("%d",
246            &stone[i]);
247
248        for ( int i=1 ; i<=n ; i++ ) if(stone[i]
249            & 1){
250                ans ^= SG[n-i];
251            }
252        }
253    }
254
255    int T = 0;
256    while(~scanf("%d", &n) && n){
257        int ans = 0;
258
259        for ( int i=1 ; i<=n ; i++ ) scanf("%d",
260            &stone[i]);
261
262        for ( int i=1 ; i<=n ; i++ ) if(stone[i]
263            & 1){
264                ans ^= SG[n-i];
265            }
266        }
267    }
268
269    int T = 0;
270    while(~scanf("%d", &n) && n){
271        int ans = 0;
272
273        for ( int i=1 ; i<=n ; i++ ) scanf("%d",
274            &stone[i]);
275
276        for ( int i=1 ; i<=n ; i++ ) if(stone[i]
277            & 1){
278                ans ^= SG[n-i];
279            }
280        }
281    }
282
283    int T = 0;
284    while(~scanf("%d", &n) && n){
285        int ans = 0;
286
287        for ( int i=1 ; i<=n ; i++ ) scanf("%d",
288            &stone[i]);
289
290        for ( int i=1 ; i<=n ; i++ ) if(stone[i]
291            & 1){
292                ans ^= SG[n-i];
293            }
294        }
295    }
296
297    int T = 0;
298    while(~scanf("%d", &n) && n){
299        int ans = 0;
300
301        for ( int i=1 ; i<=n ; i++ ) scanf("%d",
302            &stone[i]);
303
304        for ( int i=1 ; i<=n ; i++ ) if(stone[i]
305            & 1){
306                ans ^= SG[n-i];
307            }
308        }
309    }
310
311    int T = 0;
312    while(~scanf("%d", &n) && n){
313        int ans = 0;
314
315        for ( int i=1 ; i<=n ; i++ ) scanf("%d",
316            &stone[i]);
317
318        for ( int i=1 ; i<=n ; i++ ) if(stone[i]
319            & 1){
320                ans ^= SG[n-i];
321            }
322        }
323    }
324
325    int T = 0;
326    while(~scanf("%d", &n) && n){
327        int ans = 0;
328
329        for ( int i=1 ; i<=n ; i++ ) scanf("%d",
330            &stone[i]);
331
332        for ( int i=1 ; i<=n ; i++ ) if(stone[i]
333            & 1){
334                ans ^= SG[n-i];
335            }
336        }
337    }
338
339    int T = 0;
340    while(~scanf("%d", &n) && n){
341        int ans = 0;
342
343        for ( int i=1 ; i<=n ; i++ ) scanf("%d",
344            &stone[i]);
345
346        for ( int i=1 ; i<=n ; i++ ) if(stone[i]
347            & 1){
348                ans ^= SG[n-i];
349            }
350        }
351    }
352
353    int T = 0;
354    while(~scanf("%d", &n) && n){
355        int ans = 0;
356
357        for ( int i=1 ; i<=n ; i++ ) scanf("%d",
358            &stone[i]);
359
360        for ( int i=1 ; i<=n ; i++ ) if(stone[i]
361            & 1){
362                ans ^= SG[n-i];
363            }
364        }
365    }
366
367    int T = 0;
368    while(~scanf("%d", &n) && n){
369        int ans = 0;
370
371        for ( int i=1 ; i<=n ; i++ ) scanf("%d",
372            &stone[i]);
373
374        for ( int i=1 ; i<=n ; i++ ) if(stone[i]
375            & 1){
376                ans ^= SG[n-i];
377            }
378        }
379    }
380
381    int T = 0;
382    while(~scanf("%d", &n) && n){
383        int ans = 0;
384
385        for ( int i=1 ; i<=n ; i++ ) scanf("%d",
386            &stone[i]);
387
388        for ( int i=1 ; i<=n ; i++ ) if(stone[i]
389            & 1){
390                ans ^= SG[n-i];
391            }
392        }
393    }
394
395    int T = 0;
396    while(~scanf("%d", &n) && n){
397        int ans = 0;
398
399        for ( int i=1 ; i<=n ; i++ ) scanf("%d",
400            &stone[i]);
401
402        for ( int i=1 ; i<=n ; i++ ) if(stone[i]
403            & 1){
404                ans ^= SG[n-i];
405            }
406        }
407    }
408
409    int T = 0;
410    while(~scanf("%d", &n) && n){
411        int ans = 0;
412
413        for ( int i=1 ; i<=n ; i++ ) scanf("%d",
414            &stone[i]);
415
416        for ( int i=1 ; i<=n ; i++ ) if(stone[i]
417            & 1){
418                ans ^= SG[n-i];
419            }
420        }
421    }
422
423    int T = 0;
424    while(~scanf("%d", &n) && n){
425        int ans = 0;
426
427        for ( int i=1 ; i<=n ; i++ ) scanf("%d",
428            &stone[i]);
429
430        for ( int i=1 ; i<=n ; i++ ) if(stone[i]
431            & 1){
432                ans ^= SG[n-i];
433            }
434        }
435    }
436
437    int T = 0;
438    while(~scanf("%d", &n) && n){
439        int ans = 0;
440
441        for ( int i=1 ; i<=n ; i++ ) scanf("%d",
442            &stone[i]);
443
444        for ( int i=1 ; i<=n ; i++ ) if(stone[i]
445            & 1){
446                ans ^= SG[n-i];
447            }
448        }
449    }
450
451    int T = 0;
452    while(~scanf("%d", &n) && n){
453        int ans = 0;
454
455        for ( int i=1 ; i<=n ; i++ ) scanf("%d",
456            &stone[i]);
457
458        for ( int i=1 ; i<=n ; i++ ) if(stone[i]
459            & 1){
460                ans ^= SG[n-i];
461            }
462        }
463    }
464
465    int T = 0;
466    while(~scanf("%d", &n) && n){
467        int ans = 0;
468
469        for ( int i=1 ; i<=n ; i++ ) scanf("%d",
470            &stone[i]);
471
472        for ( int i=1 ; i<=n ; i++ ) if(stone[i]
473            & 1){
474                ans ^= SG[n-i];
475            }
476        }
477    }
478
479    int T = 0;
480    while(~scanf("%d", &n) && n){
481        int ans = 0;
482
483        for ( int i=1 ; i<=n ; i++ ) scanf("%d",
484            &stone[i]);
485
486        for ( int i=1 ; i<=n ; i++ ) if(stone[i]
487            & 1){
488                ans ^= SG[n-i];
489            }
490        }
491    }
492
493    int T = 0;
494    while(~scanf("%d", &n) && n){
495        int ans = 0;
496
497        for ( int i=1 ; i<=n ; i++ ) scanf("%d",
498            &stone[i]);
499
500        for ( int i=1 ; i<=n ; i++ ) if(stone[i]
501            & 1){
502                ans ^= SG[n-i];
503            }
504        }
505    }
506
507    int T = 0;
508    while(~scanf("%d", &n) && n){
509        int ans = 0;
510
511        for ( int i=1 ; i<=n ; i++ ) scanf("%d",
512            &stone[i]);
513
514        for ( int i=1 ; i<=n ; i++ ) if(stone[i]
515            & 1){
516                ans ^= SG[n-i];
517            }
518        }
519    }
520
521    int T = 0;
522    while(~scanf("%d", &n) && n){
523        int ans = 0;
524
525        for ( int i=1 ; i<=n ; i++ ) scanf("%d",
526            &stone[i]);
527
528        for ( int i=1 ; i<=n ; i++ ) if(stone[i]
529            & 1){
530                ans ^= SG[n-i];
531            }
532        }
533    }
534
535    int T = 0;
536    while(~scanf("%d", &n) && n){
537        int ans = 0;
538
539        for ( int i=1 ; i<=n ; i++ ) scanf("%d",
540            &stone[i]);
541
542        for ( int i=1 ; i<=n ; i++ ) if(stone[i]
543            & 1){
544                ans ^= SG[n-i];
545            }
546        }
547    }
548
549    int T = 0;
550    while(~scanf("%d", &n) && n){
551        int ans = 0;
552
553        for ( int i=1 ; i<=n ; i++ ) scanf("%d",
554            &stone[i]);
555
556        for ( int i=1 ; i<=n ; i++ ) if(stone[i]
557            & 1){
558                ans ^= SG[n-i];
559            }
560        }
561    }
562
563    int T = 0;
564    while(~scanf("%d", &n) && n){
565        int ans = 0;
566
567        for ( int i=1 ; i<=n ; i++ ) scanf("%d",
568            &stone[i]);
569
570        for ( int i=1 ; i<=n ; i++ ) if(stone[i]
571            & 1){
572                ans ^= SG[n-i];
573            }
574        }
575    }
576
577    int T = 0;
578    while(~scanf("%d", &n) && n){
579        int ans = 0;
580
581        for ( int i=1 ; i<=n ; i++ ) scanf("%d",
582            &stone[i]);
583
584        for ( int i=1 ; i<=n ; i++ ) if(stone[i]
585            & 1){
586                ans ^= SG[n-i];
587            }
588        }
589    }
590
591    int T = 0;
592    while(~scanf("%d", &n) && n){
593        int ans = 0;
594
595        for ( int i=1 ; i<=n ; i++ ) scanf("%d",
596            &stone[i]);
597
598        for ( int i=1 ; i<=n ; i++ ) if(stone[i]
599            & 1){
600                ans ^= SG[n-i];
601            }
602        }
603    }
604
605    int T = 0;
606    while(~scanf("%d", &n) && n){
607        int ans = 0;
608
609        for ( int i=1 ; i<=n ; i++ ) scanf("%d",
610            &stone[i]);
611
612        for ( int i=1 ; i<=n ; i++ ) if(stone[i]
613            & 1){
614                ans ^= SG[n-i];
615            }
616        }
617    }
618
619    int T = 0;
620    while(~scanf("%d", &n) && n){
621        int ans = 0;
622
623        for ( int i=1 ; i<=n ; i++ ) scanf("%d",
624            &stone[i]);
625
626        for ( int i=1 ; i<=n ; i++ ) if(stone[i]
627            & 1){
628                ans ^= SG[n-i];
629            }
630        }
631    }
632
633    int T = 0;
634    while(~scanf("%d", &n) && n){
635        int ans = 0;
636
637        for ( int i=1 ; i<=n ; i++ ) scanf("%d",
638            &stone[i]);
639
640        for ( int i=1 ; i<=n ; i++ ) if(stone[i]
641            & 1){
642                ans ^= SG[n-i];
643            }
644        }
645    }
646
647    int T = 0;
648    while(~scanf("%d", &n) && n){
649        int ans = 0;
650
651        for ( int i=1 ; i<=n ; i++ ) scanf("%d",
652            &stone[i]);
653
654        for ( int i=1 ; i<=n ; i++ ) if(stone[i]
655            & 1){
656                ans ^= SG[n-i];
657            }
658        }
659    }
660
661    int T = 0;
662    while(~scanf("%d", &n) && n){
663        int ans = 0;
664
665        for ( int i=1 ; i<=n ; i++ ) scanf("%d",
666            &stone[i]);
667
668        for ( int i=1 ; i<=n ; i++ ) if(stone[i]
669            & 1){
670                ans ^= SG[n-i];
671            }
672        }
673    }
674
675    int T = 0;
676    while(~scanf("%d", &n) && n){
677        int ans = 0;
678
679        for ( int i=1 ; i<=n ; i++ ) scanf("%d",
680            &stone[i]);
681
682        for ( int i=1 ; i<=n ; i++ ) if(stone[i]
683            & 1){
684                ans ^= SG[n-i];
685            }
686        }
687    }
688
689    int T = 0;
690    while(~scanf("%d", &n) && n){
691        int ans = 0;
692
693        for ( int i=1 ; i<=n ; i++ ) scanf("%d",
694            &stone[i]);
695
696        for ( int i=1 ; i<=n ; i++ ) if(stone[i]
697            & 1){
698                ans ^= SG[n-i];
699            }
700        }
701    }
702
703    int T = 0;
704    while(~scanf("%d", &n) && n){
705        int ans = 0;
706
707        for ( int i=1 ; i<=n ; i++ ) scanf("%d",
708            &stone[i]);
709
710        for ( int i=1 ; i<=n ; i++ ) if(stone[i]
711            & 1){
712                ans ^= SG[n-i];
713            }
714        }
715    }
716
717    int T = 0;
718    while(~scanf("%d", &n) && n){
719        int ans = 0;
720
721        for ( int i=1 ; i<=n ; i++ ) scanf("%d",
722            &stone[i]);
723
724        for ( int i=1 ; i<=n ; i++ ) if(stone[i]
725            & 1){
726                ans ^= SG[n-i];
727            }
728        }
729    }
730
731    int T = 0;
732    while(~scanf("%d", &n) && n){
733        int ans = 0;
734
735        for ( int i=1 ; i<=n ; i++ ) scanf("%d",
736            &stone[i]);
737
738        for ( int i=1 ; i<=n ; i++ ) if(stone[i]
739            & 1){
740                ans ^= SG[n-i];
741            }
742        }
743    }
744
745    int T = 0;
746    while(~scanf("%d", &n) && n){
747        int ans = 0;
748
749        for ( int i=1 ; i<=n ; i++ ) scanf("%d",
750            &stone[i]);
751
752        for ( int i=1 ; i<=n ; i++ ) if(stone[i]
753            & 1){
754                ans ^= SG[n-i];
755            }
756        }
757    }
758
759    int T = 0;
760    while(~scanf("%d", &n) && n){
761        int ans = 0;
762
763        for ( int i=1 ; i<=n ; i++ ) scanf("%d",
764            &stone[i]);
765
766        for ( int i=1 ; i<=n ; i++ ) if(stone[i]
767            & 1){
768                ans ^= SG[n-i];
769            }
770        }
771    }
772
773    int T = 0;
774    while(~scanf("%d", &n) && n){
775        int ans = 0;
776
777        for ( int i=1 ; i<=n ; i++ ) scanf("%d",
778            &stone[i]);
779
780        for ( int i=1 ; i<=n ; i++ ) if(stone[i]
781            & 1){
782                ans ^= SG[n-i];
783            }
784        }
785    }
786
787    int T = 0;
788    while(~scanf("%d", &n) && n){
789        int ans = 0;
790
791        for ( int i=1 ; i<=n ; i++ ) scanf("%d",
792            &stone[i]);
793
794        for ( int i=1 ; i<=n ; i++ ) if(stone[i]
795            & 1){
796                ans ^= SG[n-i];
797            }
798        }
799    }
800
801    int T = 0;
802    while(~scanf("%d", &n) && n){
803        int ans = 0;
804
805        for ( int i=1 ; i<=n ; i++ ) scanf("%d",
806            &stone[i]);
807
808        for ( int i=1 ; i<=n ; i++ ) if(stone[i]
809            & 1){
810                ans ^= SG[n-i];
811            }
812        }
813    }
814
815    int T = 0;
816    while(~scanf("%d", &n) && n){
817        int ans = 0;
818
819        for ( int i=1 ; i<=n ; i++ ) scanf("%d",
820            &stone[i]);
821
822        for ( int i=1 ; i<=n ; i++ ) if(stone[i]
823            & 1){
824                ans ^= SG[n-i];
825            }
826        }
827    }
828
829    int T = 0;
830    while(~scanf("%d", &n) && n){
831        int ans = 0;
832
833        for ( int i=1 ; i<=n ; i++ ) scanf("%d",
834            &stone[i]);
835
836        for ( int i=1 ; i<=n ; i++ ) if(stone[i]
837            & 1){
838                ans ^= SG[n-i];
839            }
840        }
841    }
842
843    int T = 0;
844    while(~scanf("%d", &n) && n){
845        int ans = 0;
846
847        for ( int i=1 ; i<=n ; i++ ) scanf("%d",
848            &stone[i]);
849
850        for ( int i=1 ; i<=n ; i++ ) if(stone[i]
851            & 1){
852                ans ^= SG[n-i];
853            }
854        }
855    }
856
857    int T = 0;
858    while(~scanf("%d", &n) && n){
859        int ans = 0;
860
861        for ( int i=1 ; i<=n ; i++ ) scanf("%d",
862            &stone[i]);
863
864        for ( int i=1 ; i<=n ; i++ ) if(stone[i]
865            & 1){
866                ans ^= SG[n-i];
867            }
868        }
869    }
870
871    int T = 0;
872    while(~scanf("%d", &n) && n){
873        int ans = 0;
874
875        for ( int i=1 ; i<=n ; i++ ) scanf("%d",
876            &stone[i]);
877
878        for ( int i=1 ; i<=n ; i++ ) if(stone[i]
879            & 1){
880                ans ^= SG[n-i];
881            }
882        }
883    }
884
885    int T = 0;
886    while(~scanf("%d", &n) && n){
887        int ans = 0;
888
889        for ( int i=1 ; i<=n ; i++ ) scanf("%d",
890            &stone[i]);
891
892        for ( int i=1 ; i<=n ; i++ ) if(stone[i]
893            & 1){
894                ans ^= SG[n-i];
895            }
896        }
897    }
898
899    int T = 0;
900    while(~scanf("%d", &n) && n){
901        int ans = 0;
902
903        for ( int i=1 ; i<=n ; i++ ) scanf("%d",
904            &stone[i]);
905
906        for ( int i=1 ; i<=n ; i++ ) if(stone[i]
907            & 1){
908                ans ^= SG[n-i];
909            }
910        }
911    }
912
913    int T = 0;
914    while(~scanf("%d", &n) && n){
915        int ans = 0;
916
917        for ( int i=1 ; i<=n ; i++ ) scanf("%d",
918            &stone[i]);
919
920        for ( int i=1 ; i<=n ; i++ ) if(stone[i]
921            & 1){
922                ans ^= SG[n-i];
923            }
924        }
925    }
926
927    int T = 0;
928    while(~scanf("%d", &n) && n){
929        int ans = 0;
930
931        for ( int i=1 ; i<=n ; i++ ) scanf("%d",
932            &stone[i]);
933
934        for ( int i=1 ; i<=n ; i++ ) if(stone[i]
935            & 1){
936                ans ^= SG[n-i];
937            }
938        }
939    }
940
941    int T = 0;
942    while(~scanf("%d", &n) && n){
943        int ans = 0;
944
945        for ( int i=1 ; i<=n ; i++ ) scanf("%d",
946            &stone[i]);
947
948        for ( int i=1 ; i<=n ; i++ )
```

```

6 11 dp[2][(1 << 10) + 5] ;
7 int n, m ;
8 int cur ;
9
10 void update(int s1, int s2){
11     if(s2 & (1 << m)){
12         dp[cur][s2 ^ (1 << m)] += dp[cur] ^
13             1][s1] ;
14 }
15
16 int main(){
17     while(~scanf("%d%d", &n, &m)){
18         if(m > n) swap(n, m) ;
19         memset(dp, 0, sizeof(dp)) ;
20         cur = 0 ;
21         dp[cur][(1 << m) - 1] = 1 ;
22         for ( int i=0 ; i<n ; i++ ) for ( int
23             j=0 ; j<m ; j++ ){
24             cur ^= 1 ;
25             memset(dp[cur], 0, sizeof(dp[cur])) ;
26
27             for ( int k=0 ; k<(1 << m) ; k++ ){
28                 update(k, k << 1) ; // not put
29                 if(i && !(k & (1 << (m - 1)))) {
30                     update(k, (k << 1) | (1 << m) |
31                         1) ; // put up
32                     if(j && !(k & 1)) update(k, (k << 1)
33                         | 3) ; // put left
34                 }
35             }
36             printf("%lld\n", dp[cur][(1 << m) - 1]) ;
37         }
38         return 0 ;
39     }
40 }
```

7.2 數位 DP

```

1 #include <bits/stdc++.h>
2
3 using namespace std ;
4
5 int K ;
6 int dp[20][105][105][2] ;
7 vector<int> dig ;
8
9 int solve(int pos, int sum, int dsum, bool
10    lim){
11    if(pos == -1){
12        if(sum == 0 && dsum == 0) return 1 ;
13        return 0 ;
14    }
15
16    int &d = dp[pos][sum][dsum][lim] ;
17    if(d != -1) return d ;
18
19    int up = lim ? dig[pos] : 9 ;
20    int res = 0 ;
21    for ( int i=0 ; i<=up ; i++ ){
22        res += solve(pos-1, (sum * 10 + i) %
23            K, (dsum + i) % K, lim && i==up)
24            ;
25    }
26
27    return d = res ;
28 }
29
30 int count(int n){
31     memset(dp, -1, sizeof(dp)) ;
32     dig.clear() ;
33
34     while(n > 0){
35         dig.push_back(n % 10) ;
36         n /= 10 ;
37     }
38
39     return solve(dig.size() - 1, 0, 0, 1) ;
40 }
```

7.3 樹 DP

```

1 #include <bits/stdc++.h>
2
3 #define N 505
4 #define INF 0x3f3f3f3f
5
6 using namespace std ;
7
8 struct Edge{
9     int v, w ;
10 } ;
11
12 vector<Edge> edge[N] ;
13 int n ;
14 int cnt[N] ;
15 int dp[N][N][2] ;
16
17 void init(){
18     for ( int i=0 ; i<N ; i++ )
19         edge[i].clear() ;
20     memset(cnt, 0, sizeof(cnt)) ;
21     memset(dp, INF, sizeof(dp)) ;
22 }
23
24 void DFS(int u){
25     cnt[u] = 1 ;
26     for ( auto [v, w] : edge[u] ){
27         DFS(v) ;
28         cnt[u] += cnt[v] ;
29     }
30
31     dp[u][1][0] = dp[u][1][1] = 0 ;
32
33     for ( auto [v, w] : edge[u] ){
34         for ( int i=cnt[u] ; i>1 ; i-- ) for (
35             int j=1 ; j<i && j<=cnt[v] ; j++ ){
36             dp[u][i][1] = min(dp[u][i][1],
37                 dp[u][i-j][1] + dp[v][j][1] + 2 *
38                     w) ;
39             dp[u][i][0] = min(dp[u][i][0],
40                 dp[u][i-j][1] + dp[v][j][0] + w) ;
41             dp[u][i][0] = min(dp[u][i][0],
42                 dp[u][i-j][0] + dp[v][j][1] + 2 *
43                     w) ;
44         }
45     }
46 }
47
48 int main(){
49     int t = 0 ;
50
51     while(~scanf("%d", &n) && n){
52         init() ;
53         for ( int i=0 ; i<n-1 ; i++ ){
54             int u, v, w ;
55             scanf("%d%d%d", &v, &u, &w) ;
56             edge[u].push_back({v, w}) ;
57         }
58
59         DFS(0) ;
60         printf("Case %d:\n", ++t) ;
61
62         int q, e ;
63         scanf("%d", &q) ;
64
65         while(q--){
66             scanf("%d", &e) ;
67
68             for ( int i=n ; i>=1 ; i-- )
69                 if(dp[0][i][0] <= e){
70                     printf("%d\n", i) ;
71                     break ;
72                 }
73         }
74     }
75 }
```