

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	2
2.3 Geometry	3
3 Data Structure	3
3.1 MST	3
3.2 SegmentTree	3
3.3 HLD	3
3.4 PST	4
3.5 Trie	4
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	5
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/冪根
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)

```

- 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。

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

$$x \equiv \sum_{i=1}^k a_i M_i t_i \pmod{M}.$$

兩式合併 (允許非互質)

```

1 // solve x  a1 (mod m1), x  a2 (mod m2)
2 // return {x0, lcm}; if no solution, lcm = -1
3 pair<ll, ll> crt(ll a1, ll m1,
4                   ll a2, ll m2) {
5     ll g = std::gcd(m1, m2);
6     if ((a2 - a1) % g != 0) return {0, -1};
        // no solution
7
8     ll lcm = m1 / g * m2;
9     ll m1_reduced = m1 / g;
10    ll m2_reduced = m2 / g;
11
12    ll diff = (a2 - a1) / g % m2_reduced;
13    if (diff < 0) diff += m2_reduced;
14
15    ll inv = mod_pow(m1_reduced, m2_reduced
        - 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
    &y){
2     if(b == 0){
3         x = c/a ;
4         y = 0 ;
5         return a ;
6     }
7     ll d = extgcd(b, a%b, c, x, y), tmp =
        x ;
8     x = y ;
9     y = tmp - (a/b)*y ;
10    return d ;
11 }

```

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

$$(a \pm b) \pmod{m} = ((a \pmod{m}) \pm (b \pmod{m})) \pmod{m},$$

$$(ab) \pmod{m} = ((a \pmod{m})(b \pmod{m})) \pmod{m},$$

$$(a^k) \pmod{m} = ((a \pmod{m})^k) \pmod{m},$$

$$(-a) \pmod{m} = (m - (a \pmod{m})) \pmod{m}.$$

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

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 $a \pm c \equiv b \pm d \pmod{m}$, $ac \equiv bd \pmod{m}$ 。

2.2 Combinatorics

Binomial Coefficient Identities

$${n \choose k} = \frac{n!}{k!(n-k)!},$$

$${n \choose k} = {n-1 \choose k} + {n-1 \choose k-1},$$

$$\sum_{k=0}^n {n \choose k} = 2^n, \quad \sum_{k=0}^n k {n \choose k} = n2^{n-1}.$$

Stars and Bars 非負整數解數量：

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

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 :

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

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
11 vector<Edge> edge ;
12
13 void init(){
14     edge.clear() ;
15     for ( int i=0 ; i<N ; i++ ){
16         par[i] = i ;
17         sz[i] = 1 ;
18     }
19     sum = 0 ;
20 }
21
22 int find(int x){
23     if(x == par[x]) return x ;
24     return par[x] = find(par[x]) ;
25 }
26
27 int merge(int x, int y){
28     x = find(x) ;
29     y = find(y) ;
30
31     if(x == y) return 0 ;
32     if(sz[x] > sz[y]) swap(x, y) ;
33     par[x] = y ;
34     sz[y] += sz[x] ;
35
36     return 1 ;
37 }
```

```

48     seg[id].sum = arr[L] ;
49     return ;
50 }
51
52     int M = (L + R) >> 1 ;
53     build(L, M, lc) ;
54     build(M+1, R, rc) ;
55
56     pull(id) ;
57     seg[id].sz = seg[lc].sz + seg[rc].sz ;
58 }
59
60 void modify(int l, int r, LazyTag &tag,
61             int L=1, int R=n, int id=1){
62     if(l <= L && R <= r){
63         AddTag(id, tag) ;
64         return ;
65     }
66
67     push(id) ;
68     int M = (L + R) >> 1 ;
69     if(r <= M) modify(l, r, tag, L, M,
70                         lc) ;
71     else if(l > M) modify(l, r, tag, M+1,
72                           R, rc) ;
73     else{
74         modify(l, r, tag, L, M, lc) ;
75         modify(l, r, tag, M+1, R, rc) ;
76     }
77     pull(id) ;
78 }
79
80 int query(int l, int r, int L=1, int R=n,
81            int id=1){
82     if(l <= L && R <= r) return
83         seg[id].sum ;
84
85     push(id) ;
86     int M = (L + R) >> 1 ;
87     if(r <= M) return query(l, r, L, M,
88                             lc) ;
89     else if(l > M) return query(l, r,
90                                 M+1, R, rc) ;
91     else return query(l, r, L, M, lc) +
92             query(l, r, M+1, R, rc) ;
93 }
```

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     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
38     void push(int id){
39         AddTag(lc, seg[id].tag) ;
40         AddTag(rc, seg[id].tag) ;
41         seg[id].tag = {0, 0} ;
42     }
43
44 public:
45     void build(int L=1, int R=n, int id=1){
46         seg[id].sum = 0 ;
47         seg[id].tag = {0, 0} ;
48         seg[id].sz = 1 ;
49
50         if(L == R){
```

3.3 HLD

```

1 /* HLD */
2 int fa[Maxn], top[Maxn], son[Maxn],
3      sz[Maxn], dep[Maxn] = {0}, dfn[Maxn],
4      rk[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]]) son[u] = v ;
15     }
16
17     void dfs2(int u, int t){
18         top[u] = t ;
19         dfn[u] = ++dfscnt ;
20         rk[dfscnt] = u ;
21
22         if(son[u] == -1) return ;
23         dfs2(son[u], t) ;
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] ;
55
56 class SegmentTree{
57 private:
58   void pull(int id){
59     // normal pull
60   }
61   void AddTag(int id, int tag){
62     // normal AddTag
63   }
64   void push(int id){
65     // normal push
66   }
67
68   void modify(int l, int r, int tag, int
69   L=1, int R=n, int id=1){
70     // normal modify
71   }
72
73   ColorSeg query(int l, int r, int L=1, int
74   R=n, int id=1){
75     // normal query
76   }
77 public:
78   void build(int L=1, int R=n, int id=1){
79     // normal build
80   }
81
82   // update val from u to v (simple path)
83   void update(int u, int v, int val){
84     while(top[u] != top[v]){
85       if(dep[top[u]] < dep[top[v]]) swap(u,
86                                               v) ;
87       modify(dfn[top[u]], dfn[u], val) ;
88       u = fa[top[u]] ;
89     }
90
91     if(dep[u] < dep[v]) swap(u, v) ;
92     modify(dfn[v], dfn[u], val) ;
93
94   // get sum from u to v (simple path)
95   int get(int u, int v){
96     pair<int, ColorSeg> U, V ;
97     ColorSeg M ;

```

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 public:
27  PersistentSegmentTree(int _n){
28    n = _n ;
29    cnt = 0 ;
30
31    int root = build(1, n) ;
32    version.push_back(root) ;
33  }
34
35  void update(int ver, int idx){
36    auto upd = [&](auto &self, const int
37                  cur, int L, int R){
38      int cur_cnt = cnt++ ;
39
40      if(L == R){
41        seg[cur_cnt] = {seg[cur].sum + 1, 0,
42                       0} ;
43        return cur_cnt ;
44      }
45
46      int M = (L + R) >> 1 ;
47      int lc = seg[cur].left ;
48      int rc = seg[cur].right ;

```

```

48      if(idx <= M) lc = self(self,
49          seg[cur].left, L, M) ;
50      else rc = self(self, seg[cur].right,
51                      M+1, R) ;
52
53      seg[cur_cnt] = {seg[lc].sum +
54                      seg[rc].sum, lc, rc} ;
55
56      return cur_cnt ;
57  };
58
59  int root = upd(upd, version[ver], 1, n) ;
60  version.push_back(root) ;
61
62 int query(int verL, int verR, int k){
63  auto qry = [&](auto &self, const int
64                cur_old, const int cur_new, int L,
65                int R){
66    if(L == R) return L ;
67
68    int old_l = seg[cur_old].left, old_r =
69                seg[cur_old].right ;
70    int new_l = seg[cur_new].left, new_r =
71                seg[cur_new].right ;
72
73    int dl = seg[new_l].sum -
74            seg[old_l].sum ;
75    int dr = seg[new_r].sum -
76            seg[old_r].sum ;
77
78    int M = (L + R) >> 1 ;
79
80    if(dl >= k) return self(self, old_l,
81                             new_l, L, M) ;
82    k -= dl ;
83    return self(self, old_r, new_r, M+1,
84                           R) ;
85  };
86
87  int idx = qry(qry, version[verL-1],
88                version[verR], 1, n) ;
89  return idx ;
90}

```

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   }
31
32   string search(string &str, int n){
33     TrieNode* node = root ;
34     for ( auto s : str ){
35       int path = s - 'a' ;
36
37       if(node->next[path] == nullptr)
38         return "NO" ;
39       node = node->next[path] ;
40     }
41
42     if(node->end.size() > 0)
43       return "YES" ;
44     else
45       return "NO" ;
46   }
47
48   int count(string &str, int n){
49     TrieNode* node = root ;
50     for ( auto s : str ){
51       int path = s - 'a' ;
52
53       if(node->next[path] == nullptr)
54         return 0 ;
55       node = node->next[path] ;
56     }
57
58     if(node->end.size() > 0)
59       return node->end.size() ;
60     else
61       return 0 ;
62   }
63
64   void erase(string &str, int n){
65     TrieNode* node = root ;
66     for ( auto s : str ){
67       int path = s - 'a' ;
68
69       if(node->next[path] == nullptr)
70         return ;
71       node = node->next[path] ;
72     }
73
74     if(node->end.size() == 1)
75       delete node ;
76     else
77       node->end.erase(str.back());
78   }
79
80   void print(TrieNode* node, string str=""){
81     if(node->end.size() > 0)
82       cout << str << endl ;
83
84     for ( int i=0 ; i<26 ; i++ )
85       if(node->next[i] != nullptr)
86         print(node->next[i], str + char(i+'a'));
87   }
88
89   void print(TrieNode* node, string str=""){
90     if(node->end.size() > 0)
91       cout << str << endl ;
92
93     for ( int i=0 ; i<26 ; i++ )
94       if(node->next[i] != nullptr)
95         print(node->next[i], str + char(i+'a'));
96   }
97
98   void print(TrieNode* node, string str=""){
99     if(node->end.size() > 0)
100       cout << str << endl ;
101
102    for ( int i=0 ; i<26 ; i++ )
103      if(node->next[i] != nullptr)
104        print(node->next[i], str + char(i+'a'));
105
106    if(node->end.size() > 0)
107      cout << str << endl ;
108  }
109
110  void print(TrieNode* node, string str=""){
111    if(node->end.size() > 0)
112      cout << str << endl ;
113
114    for ( int i=0 ; i<26 ; i++ )
115      if(node->next[i] != nullptr)
116        print(node->next[i], str + char(i+'a'));
117
118    if(node->end.size() > 0)
119      cout << str << endl ;
120  }
121
122  void print(TrieNode* node, string str=""){
123    if(node->end.size() > 0)
124      cout << str << endl ;
125
126    for ( int i=0 ; i<26 ; i++ )
127      if(node->next[i] != nullptr)
128        print(node->next[i], str + char(i+'a'));
129
130    if(node->end.size() > 0)
131      cout << str << endl ;
132  }
133
134  void print(TrieNode* node, string str=""){
135    if(node->end.size() > 0)
136      cout << str << endl ;
137
138    for ( int i=0 ; i<26 ; i++ )
139      if(node->next[i] != nullptr)
140        print(node->next[i], str + char(i+'a'));
141
142    if(node->end.size() > 0)
143      cout << str << endl ;
144  }
145
146  void print(TrieNode* node, string str=""){
147    if(node->end.size() > 0)
148      cout << str << endl ;
149
150    for ( int i=0 ; i<26 ; i++ )
151      if(node->next[i] != nullptr)
152        print(node->next[i], str + char(i+'a'));
153
154    if(node->end.size() > 0)
155      cout << str << endl ;
156  }
157
158  void print(TrieNode* node, string str=""){
159    if(node->end.size() > 0)
160      cout << str << endl ;
161
162    for ( int i=0 ; i<26 ; i++ )
163      if(node->next[i] != nullptr)
164        print(node->next[i], str + char(i+'a'));
165
166    if(node->end.size() > 0)
167      cout << str << endl ;
168  }
169
170  void print(TrieNode* node, string str=""){
171    if(node->end.size() > 0)
172      cout << str << endl ;
173
174    for ( int i=0 ; i<26 ; i++ )
175      if(node->next[i] != nullptr)
176        print(node->next[i], str + char(i+'a'));
177
178    if(node->end.size() > 0)
179      cout << str << endl ;
180  }
181
182  void print(TrieNode* node, string str=""){
183    if(node->end.size() > 0)
184      cout << str << endl ;
185
186    for ( int i=0 ; i<26 ; i++ )
187      if(node->next[i] != nullptr)
188        print(node->next[i], str + char(i+'a'));
189
190    if(node->end.size() > 0)
191      cout << str << endl ;
192  }
193
194  void print(TrieNode* node, string str=""){
195    if(node->end.size() > 0)
196      cout << str << endl ;
197
198    for ( int i=0 ; i<26 ; i++ )
199      if(node->next[i] != nullptr)
200        print(node->next[i], str + char(i+'a'));
201
202    if(node->end.size() > 0)
203      cout << str << endl ;
204  }
205
206  void print(TrieNode* node, string str=""){
207    if(node->end.size() > 0)
208      cout << str << endl ;
209
210    for ( int i=0 ; i<26 ; i++ )
211      if(node->next[i] != nullptr)
212        print(node->next[i], str + char(i+'a'));
213
214    if(node->end.size() > 0)
215      cout << str << endl ;
216  }
217
218  void print(TrieNode* node, string str=""){
219    if(node->end.size() > 0)
220      cout << str << endl ;
221
222    for ( int i=0 ; i<26 ; i++ )
223      if(node->next[i] != nullptr)
224        print(node->next[i], str + char(i+'a'));
225
226    if(node->end.size() > 0)
227      cout << str << endl ;
228  }
229
230  void print(TrieNode* node, string str=""){
231    if(node->end.size() > 0)
232      cout << str << endl ;
233
234    for ( int i=0 ; i<26 ; i++ )
235      if(node->next[i] != nullptr)
236        print(node->next[i], str + char(i+'a'));
237
238    if(node->end.size() > 0)
239      cout << str << endl ;
240  }
241
242  void print(TrieNode* node, string str=""){
243    if(node->end.size() > 0)
244      cout << str << endl ;
245
246    for ( int i=0 ; i<26 ; i++ )
247      if(node->next[i] != nullptr)
248        print(node->next[i], str + char(i+'a'));
249
250    if(node->end.size() > 0)
251      cout << str << endl ;
252  }
253
254  void print(TrieNode* node, string str=""){
255    if(node->end.size() > 0)
256      cout << str << endl ;
257
258    for ( int i=0 ; i<26 ; i++ )
259      if(node->next[i] != nullptr)
260        print(node->next[i], str + char(i+'a'));
261
262    if(node->end.size() > 0)
263      cout << str << endl ;
264  }
265
266  void print(TrieNode* node, string str=""){
267    if(node->end.size() > 0)
268      cout << str << endl ;
269
270    for ( int i=0 ; i<26 ; i++ )
271      if(node->next[i] != nullptr)
272        print(node->next[i], str + char(i+'a'));
273
274    if(node->end.size() > 0)
275      cout << str << endl ;
276  }
277
278  void print(TrieNode* node, string str=""){
279    if(node->end.size() > 0)
280      cout << str << endl ;
281
282    for ( int i=0 ; i<26 ; i++ )
283      if(node->next[i] != nullptr)
284        print(node->next[i], str + char(i+'a'));
285
286    if(node->end.size() > 0)
287      cout << str << endl ;
288  }
289
290  void print(TrieNode* node, string str=""){
291    if(node->end.size() > 0)
292      cout << str << endl ;
293
294    for ( int i=0 ; i<26 ; i++ )
295      if(node->next[i] != nullptr)
296        print(node->next[i], str + char(i+'a'));
297
298    if(node->end.size() > 0)
299      cout << str << endl ;
300  }
301
302  void print(TrieNode* node, string str=""){
303    if(node->end.size() > 0)
304      cout << str << endl ;
305
306    for ( int i=0 ; i<26 ; i++ )
307      if(node->next[i] != nullptr)
308        print(node->next[i], str + char(i+'a'));
309
310    if(node->end.size() > 0)
311      cout << str << endl ;
312  }
313
314  void print(TrieNode* node, string str=""){
315    if(node->end.size() > 0)
316      cout << str << endl ;
317
318    for ( int i=0 ; i<26 ; i++ )
319      if(node->next[i] != nullptr)
320        print(node->next[i], str + char(i+'a'));
321
322    if(node->end.size() > 0)
323      cout << str << endl ;
324  }
325
326  void print(TrieNode* node, string str=""){
327    if(node->end.size() > 0)
328      cout << str << endl ;
329
330    for ( int i=0 ; i<26 ; i++ )
331      if(node->next[i] != nullptr)
332        print(node->next[i], str + char(i+'a'));
333
334    if(node->end.size() > 0)
335      cout << str << endl ;
336  }
337
338  void print(TrieNode* node, string str=""){
339    if(node->end.size() > 0)
340      cout << str << endl ;
341
342    for ( int i=0 ; i<26 ; i++ )
343      if(node->next[i] != nullptr)
344        print(node->next[i], str + char(i+'a'));
345
346    if(node->end.size() > 0)
347      cout << str << endl ;
348  }
349
350  void print(TrieNode* node, string str=""){
351    if(node->end.size() > 0)
352      cout << str << endl ;
353
354    for ( int i=0 ; i<26 ; i++ )
355      if(node->next[i] != nullptr)
356        print(node->next[i], str + char(i+'a'));
357
358    if(node->end.size() > 0)
359      cout << str << endl ;
360  }
361
362  void print(TrieNode* node, string str=""){
363    if(node->end.size() > 0)
364      cout << str << endl ;
365
366    for ( int i=0 ; i<26 ; i++ )
367      if(node->next[i] != nullptr)
368        print(node->next[i], str + char(i+'a'));
369
370    if(node->end.size() > 0)
371      cout << str << endl ;
372  }
373
374  void print(TrieNode* node, string str=""){
375    if(node->end.size() > 0)
376      cout << str << endl ;
377
378    for ( int i=0 ; i<26 ; i++ )
379      if(node->next[i] != nullptr)
380        print(node->next[i], str + char(i+'a'));
381
382    if(node->end.size() > 0)
383      cout << str << endl ;
384  }
385
386  void print(TrieNode* node, string str=""){
387    if(node->end.size() > 0)
388      cout << str << endl ;
389
390    for ( int i=0 ; i<26 ; i++ )
391      if(node->next[i] != nullptr)
392        print(node->next[i], str + char(i+'a'));
393
394    if(node->end.size() > 0)
395      cout << str << endl ;
396  }
397
398  void print(TrieNode* node, string str=""){
399    if(node->end.size() > 0)
400      cout << str << endl ;
401
402    for ( int i=0 ; i<26 ; i++ )
403      if(node->next[i] != nullptr)
404        print(node->next[i], str + char(i+'a'));
405
406    if(node->end.size() > 0)
407      cout << str << endl ;
408  }
409
410  void print(TrieNode* node, string str=""){
411    if(node->end.size() > 0)
412      cout << str << endl ;
413
414    for ( int i=0 ; i<26 ; i++ )
415      if(node->next[i] != nullptr)
416        print(node->next[i], str + char(i+'a'));
417
418    if(node->end.size() > 0)
419      cout << str << endl ;
420  }
421
422  void print(TrieNode* node, string str=""){
423    if(node->end.size() > 0)
424      cout << str << endl ;
425
426    for ( int i=0 ; i<26 ; i++ )
427      if(node->next[i] != nullptr)
428        print(node->next[i], str + char(i+'a'));
429
430    if(node->end.size() > 0)
431      cout << str << endl ;
432  }
433
434  void print(TrieNode* node, string str=""){
435    if(node->end.size() > 0)
436      cout << str << endl ;
437
438    for ( int i=0 ; i<26 ; i++ )
439      if(node->next[i] != nullptr)
440        print(node->next[i], str + char(i+'a'));
441
442    if(node->end.size() > 0)
443      cout << str << endl ;
444  }
445
446  void print(TrieNode* node, string str=""){
447    if(node->end.size() > 0)
448      cout << str << endl ;
449
450    for ( int i=0 ; i<26 ; i++ )
451      if(node->next[i] != nullptr)
452        print(node->next[i], str + char(i+'a'));
453
454    if(node->end.size() > 0)
455      cout << str << endl ;
456  }
457
458  void print(TrieNode* node, string str=""){
459    if(node->end.size() > 0)
460      cout << str << endl ;
461
462    for ( int i=0 ; i<26 ; i++ )
463      if(node->next[i] != nullptr)
464        print(node->next[i], str + char(i+'a'));
465
466    if(node->end.size() > 0)
467      cout << str << endl ;
468  }
469
470  void print(TrieNode* node, string str=""){
471    if(node->end.size() > 0)
472      cout << str << endl ;
473
474    for ( int i=0 ; i<26 ; i++ )
475      if(node->next[i] != nullptr)
476        print(node->next[i], str + char(i+'a'));
477
478    if(node->end.size() > 0)
479      cout << str << endl ;
480  }
481
482  void print(TrieNode* node, string str=""){
483    if(node->end.size() > 0)
484      cout << str << endl ;
485
486    for ( int i=0 ; i<26 ; i++ )
487      if(node->next[i] != nullptr)
488        print(node->next[i], str + char(i+'a'));
489
490    if(node->end.size() > 0)
491      cout << str << endl ;
492  }
493
494  void print(TrieNode* node, string str=""){
495    if(node->end.size() > 0)
496      cout << str << endl ;
497
498    for ( int i=0 ; i<26 ; i++ )
499      if(node->next[i] != nullptr)
500        print(node->next[i], str + char(i+'a'));
501
502    if(node->end.size() > 0)
503      cout << str << endl ;
504  }
505
506  void print(TrieNode* node, string str=""){
507    if(node->end.size() > 0)
508      cout << str << endl ;
509
510    for ( int i=0 ; i<26 ; i++ )
511      if(node->next[i] != nullptr)
512        print(node->next[i], str + char(i+'a'));
513
514    if(node->end.size() > 0)
515      cout << str << endl ;
516  }
517
518  void print(TrieNode* node, string str=""){
519    if(node->end.size() > 0)
520      cout << str << endl ;
521
522    for ( int i=0 ; i<26 ; i++ )
523      if(node->next[i] != nullptr)
524        print(node->next[i], str + char(i+'a'));
525
526    if(node->end.size() > 0)
527      cout << str << endl ;
528  }
529
530  void print(TrieNode* node, string str=""){
531    if(node->end.size() > 0)
532      cout << str << endl ;
533
534    for ( int i=0 ; i<26 ; i++ )
535      if(node->next[i] != nullptr)
536        print(node->next[i], str + char(i+'a'));
537
538    if(node->end.size() > 0)
539      cout << str << endl ;
540  }
541
542  void print(TrieNode* node, string str=""){
543    if(node->end.size() > 0)
544      cout << str << endl ;
545
546    for ( int i=0 ; i<26 ; i++ )
547      if(node->next[i] != nullptr)
548        print(node->next[i], str + char(i+'a'));
549
550    if(node->end.size() > 0)
551      cout << str << endl ;
552  }
553
554  void print(TrieNode* node, string str=""){
555    if(node->end.size() > 0)
556      cout << str << endl ;
557
558    for ( int i=0 ; i<26 ; i++ )
559      if(node->next[i] != nullptr)
560        print(node->next[i], str + char(i+'a'));
561
562    if(node->end.size() > 0)
563      cout << str << endl ;
564  }
565
566  void print(TrieNode* node, string str=""){
567    if(node->end.size() > 0)
568      cout << str << endl ;
569
570    for ( int i=0 ; i<26 ; i++ )
571      if(node->next[i] != nullptr)
572        print(node->next[i], str + char(i+'a'));
573
574    if(node->end.size() > 0)
575      cout << str << endl ;
576  }
577
578  void print(TrieNode* node, string str=""){
579    if(node->end.size() > 0)
580      cout << str << endl ;
581
582    for ( int i=0 ; i<26 ; i++ )
583      if(node->next[i] != nullptr)
584        print(node->next[i], str + char(i+'a'));
585
586    if(node->end.size() > 0)
587      cout << str << endl ;
588  }
589
590  void print(TrieNode* node, string str=""){
591    if(node->end.size() > 0)
592      cout << str << endl ;
593
594    for ( int i=0 ; i<26 ; i++ )
595      if(node->next[i] != nullptr)
596        print(node->next[i], str + char(i+'a'));
597
598    if(node->end.size() > 0)
599      cout << str << endl ;
600  }
601
602  void print(TrieNode* node, string str=""){
603    if(node->end.size() > 0)
604      cout << str << endl ;
605
606    for ( int i=0 ; i<26 ; i++ )
607      if(node->next[i] != nullptr)
608        print(node->next[i], str + char(i+'a'));
609
610    if(node->end.size() > 0)
611      cout << str << endl ;
612  }
613
614  void print(TrieNode* node, string str=""){
615    if(node->end.size() > 0)
616      cout << str << endl ;
617
618    for ( int i=0 ; i<26 ; i++ )
619      if(node->next[i] != nullptr)
620        print(node->next[i], str + char(i+'a'));
621
622    if(node->end.size() > 0)
623      cout << str << endl ;
624  }
625
626  void print(TrieNode* node, string str=""){
627    if(node->end.size() > 0)
628      cout << str << endl ;
629
630    for ( int i=0 ; i<26 ; i++ )
631      if(node->next[i] != nullptr)
632        print(node->next[i], str + char(i+'a'));
633
634    if(node->end.size() > 0)
635      cout << str << endl ;
636  }
637
638  void print(TrieNode* node, string str=""){
639    if(node->end.size() > 0)
640      cout << str << endl ;
641
642    for ( int i=0 ; i<26 ; i++ )
643      if(node->next[i] != nullptr)
644        print(node->next[i], str + char(i+'a'));
645
646    if(node->end.size() > 0)
647      cout << str << endl ;
648  }
649
650  void print(TrieNode* node, string str=""){
651    if(node->end.size() > 0)
652      cout << str << endl ;
653
654    for ( int i=0 ; i<26 ; i++ )
655      if(node->next[i] != nullptr)
656        print(node->next[i], str + char(i+'a'));
657
658    if(node->end.size() > 0)
659      cout << str << endl ;
660  }
661
662  void print(TrieNode* node, string str=""){
663    if(node->end.size() > 0)
664      cout << str << endl ;
665
666    for ( int i=0 ; i<26 ; i++ )
667      if(node->next[i] != nullptr)
668        print(node->next[i], str + char(i+'a'));
669
670    if(node->end.size() > 0)
671      cout << str << endl ;
672  }
673
674  void print(TrieNode* node, string str=""){
675    if(node->end.size() > 0)
676      cout << str << endl ;
677
678    for ( int i=0 ; i<26 ; i++ )
679      if(node->next[i] != nullptr)
680        print(node->next[i], str + char(i+'a'));
681
682    if(node->end.size() > 0)
683      cout << str << endl ;
684  }
685
686  void print(TrieNode* node, string str=""){
687    if(node->end.size() > 0)
688      cout << str << endl ;
689
690    for ( int i=0 ; i<26 ; i++ )
691      if(node->next[i] != nullptr)
692        print(node->next[i], str + char(i+'a'));
693
694    if(node->end.size() > 0)
695      cout << str << endl ;
696  }
697
698  void print(TrieNode* node, string str=""){
699    if(node->end.size() > 0)
700      cout << str << endl ;
701
702    for ( int i=0 ; i<26 ; i++ )
703      if(node->next[i] != nullptr)
704        print(node->next[i], str + char(i+'a'));
705
706    if(node->end.size() > 0)
707      cout << str << endl ;
708  }
709
710  void print(TrieNode* node, string str=""){
711    if(node->end.size() > 0)
712      cout << str << endl ;
713
714    for ( int i=0 ; i<26 ; i++ )
715      if(node->next[i] != nullptr)
716        print(node->next[i], str + char(i+'a'));
717
718    if(node->end.size() > 0)
719      cout << str << endl ;
720  }
721
722  void print(TrieNode* node, string str=""){
723    if(node->end.size() > 0)
724      cout << str << endl ;
725
726    for ( int i=0 ; i<26 ; i++ )
727      if(node->next[i] != nullptr)
728        print(node->next[i], str + char(i+'a'));
729
730    if(node->end.size() > 0)
731      cout << str << endl ;
732  }
733
734  void print(TrieNode* node, string str=""){
735    if(node->end.size() > 0)
736      cout << str << endl ;
737
738    for ( int i=0 ; i<26 ; i++ )
739      if(node->next[i] != nullptr)
740        print(node->next[i], str + char(i+'a'));
741
742    if(node->end.size() > 0)
743      cout << str << endl ;
744  }
745
746  void print(TrieNode* node, string str=""){
747    if(node->end.size() > 0)
748      cout << str << endl ;
749
750    for ( int i=0 ; i<26 ; i++ )
751      if(node->next[i] != nullptr)
752        print(node->next[i], str + char(i+'a'));
753
754    if(node->end.size() > 0)
755      cout << str << endl ;
756  }
757
758 
```

```

    }
    node->end.insert(n) ;
}

void search(string &str){
    TrieNode* node = root ;
    for ( auto s : str ){
        int path = s - 'a' ;
        if(node->next[path] == nullptr)
            return ;
        node = node->next[path] ;
    }

    int flg = 0 ;
    for ( auto n : node->end ){
        if(flg) cout << " " ;
        else flg = 1 ;

        cout << n ;
    }
}

void clear(TrieNode* node) {
    if (!node) return ;
    for (int i = 0; i < 26; i++) {
        if (node->next[i]) {
            clear(node->next[i]) ;
        }
    }
    delete node ;
}

~Trie(){
    clear(root) ;
}
};

```

3.6 BIT 單修區查

```

// 點單修改 區間查詢
#define lowbit(x) (x & -x)

int BIT[MAX_SIZE] ;
int n ;

void modify(int idx, int val){
    for ( ; idx <= n ; idx += lowbit(idx) ){
        BIT[idx] += val ;
    }
}

// ans: query(R) - query(L-1)
int query(int idx){
    int sum = 0 ;
    for ( ; idx ; idx -= lowbit(idx) ){
        sum += BIT[idx] ;
    }
}

void init(){
    memset(BIT, 0, sizeof(BIT)) ;
}

```

3.7 BIT 區修單查

```
// 區間修改，單點查詢
#define lowbit(x) (x & -x)

int BIT[MAX_SIZE] ;
int n ;

void modify(int idx, int val){
    for ( ; idx <= n ; idx += lowbit(idx) ) {
        BIT[idx] += val ;
    }
}
```

```

10    }
11}
12
13// ans: query(i)
14int query(int idx){
15    int sum = 0 ;
16    for ( ; idx ; idx -= lowbit(idx) ){
17        sum += BIT[idx] ;
18    }
19
20    return sum ;
21}
22
23void init(){
24    memset(BIT, 0, sizeof(BIT)) ;
25}
26
27void build(){
28    arr[0] = 0 ;
29    for ( int i=1 ; i<=n ; i++ ) modify(i,
30        arr[i] - arr[i-1]) ;
31}
32// usage
33// add val
34modify(L, x) ;
35modify(R+1, -x) ;

```

3.8 BIT 區修區查

```

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             else low[u] = min(low[u], dfn[v]) ;
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],
   dfscnt = 0, cnt_scc = 0 ;
3 stack<int> st ;
4 bitset<Maxn> inSt, vis ;
5
6 void dfs(int u, int from){
7   dfn[u] = low[u] = ++dfscnt ;
8   st.push(u) ;
9   inSt[u] = 1 ;
10
11  for ( auto v : g[u] ){
12    if(!inSt[v] && dfn[v] != -1) continue ;
13    if(dfn[v] == -1) dfs(v, u) ;
14    low[u] = min(low[u], low[v]) ;
15  }
16
17  if(dfn[u] == low[u]){
18    cnt_scc++ ;
19    int x ;
20
21    do{
22      x = st.top() ;
23      st.pop() ;
24
25      inSt[x] = 0 ;
26      sccId[x] = cnt_scc ;
27      scc[cnt_scc].push_back(x) ;
28    }
29    while(x != u) ;
30  }
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 }
```

4 Graph

4.1 cut vertex AND bridge

```

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] == -1){
55         dfs(i, i) ;
56     }
57 }
```

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         int v = e[i].v ;
23
24         if(dfn[v] == -1){
25             dfs1(v, i) ;
26             if(dfn[u] < low[v]) bz[i] = bz[i^1] =
27                 1 ;
28             low[u] = min(low[u], low[v]) ;
29         }
30         else if(i != (from ^ 1)) low[u] =
31             min(low[u], dfn[v]) ;
32     }
33
34 void dfs2(int u, int id){
35     vis_bcc[u] = id ;
36     bcc[id].push_back(u) ;
37
38     for ( int i=head[u] ; i!= -1 ; i=e[i].next )
39         int v = e[i].v ;
40
41         if(vis_bcc[v] != -1 || bz[i]) continue ;
42         dfs2(v, id) ;
43     }
44
45 void init(){
46     memset(dfn, -1, sizeof(dfn)) ;
47     memset(head, -1, sizeof(head)) ;
48     memset(vis_bcc, -1, sizeof(vis_bcc)) ;
49 }
50
51 int main(){
52     init() ;
53     input() ;
54     for ( int i=1 ; i<=n ; i++ ) if(dfn[i] == -1){
55         dfs1(i, 0) ;
56     }
57 }
```

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    long long cross(const Coordinate& o, const
18        Coordinate& a, const Coordinate& b){
19        return (a.x - o.x) * (b.y - o.y) - (a.y -
20            o.y) * (b.x - o.x) ;
21    }
22
23    void input(){
24        nodes.clear() ;
25
26        int n, x, y ;
27        char c ;
28        cin >> n ;
29
30        for ( int i=0 ; i<n ; i++ ){
31            cin >> x >> y >> c ;
32            if(c == 'Y') nodes.push_back({x, y}) ;
33        }
34
35        void monotone(){
36            sort(nodes.begin(), nodes.end()) ;
37
38            int n = unique(nodes.begin(), nodes.end()) -
39                nodes.begin() ;
40
41            vector<Coordinate> ch(n+1) ;
42
43            int m = 0 ;
44
45            for ( int i=0 ; i<n ; i++ ){
46                while(m > 1 && cross(ch[m-2], ch[m-1],
47                    nodes[i]) < 0) m-- ;
48                ch[m++] = nodes[i] ;
49            }
50
51            for ( int i=n-2, t=m ; i>=0 ; i-- ){
52                while(m > t && cross(ch[m-2], ch[m-1],
53                    nodes[i]) < 0) m-- ;
54                ch[m++] = nodes[i] ;
55            }
56
57            if(n > 1) m-- ;
58            cout << m << endl ;
59
60            for ( int i=0 ; i<m ; i++ ) cout <<
61                ch[i].x << " " << ch[i].y << endl ;
62        }
63    }
64 }
```

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                     if(v == T) return 1 ;
30                     q.push(v) ;
31                 }
32             }
33
34             return 0 ;
35         }
36
37         int dfs(int u, int flow){
38             if(u == T) return flow ;
39             int d, rest = 0 ;
40
41             for ( int &i=cur[u] ; i!= -1 ;
42                 i=e[i].next ){
43                 int v = e[i].v ;
44                 if(dep[v] == dep[u] + 1 && e[i].cap >
45                     0){
46                     d = dfs(v, min(flow - rest,
47                         e[i].cap)) ;
48
49                     if(d > 0){
50                         e[i].cap -= d ;
51                         e[i^1].cap += d ;
52                         rest += d ;
53
54                         if(rest == flow) break ;
55                     }
56
57                     if(rest != flow) dep[u] = -1 ;
58                     return rest ;
59                 }
60             }
61             MaxFlow(int n, int s, int t){
62                 N = n ; S = s ; T = t ;
63                 e.reserve(n*n) ;
64                 head.assign(n+1, -1) ;
65                 cur.resize(n+1) ;
66                 dep.resize(n+1) ;
67             }
68
69             void AddEdge(int u, int v, int cap){
70                 e.push_back({v, cap, head[u]}) ;
71                 head[u] = e.size() - 1 ;
72                 e.push_back({u, 0, head[v]}) ;
73                 head[v] = e.size() - 1 ;
74             }
75         }
76     }
77 }
```

```

73 }
74
75 int run(){
76     int ans = 0 ;
77     while(bfs()){
78         ans += dfs(S, 0x3f3f3f3f) ;
79     }
80     return ans ;
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 cost){
22        e[++tot] = {v, cap, cost, head[u]} ;
23        head[u] = tot ;
24        e[++tot] = {u, 0, -cost, head[v]} ;
25        head[v] = tot ;
26    }
27
28    int run(){
29        vector<int> dis(N+1), pot(N+1, 0),
30            preE(N+1) ;
31        int flow = 0, cost = 0 ;
32
33        auto dijkstra = [&](){
34            fill(dis.begin(), dis.end(), INF) ;
35            priority_queue<pii, vector<pii>,
36                greater<pii>> pq ;
37            dis[s] = 0 ;
38            pq.push({0, s}) ;
39
40            while(!pq.empty()){
41                auto [d, u] = pq.top() ; pq.pop() ;
42                if(d > dis[u]) continue ;
43                for ( int i=head[u] ; i!= -1 ;
44                    i=e[i].next ){
45                    int v = e[i].v, cap = e[i].cap, w =
46                        e[i].cost ;
47                    if(cap && dis[v] > d + w + pot[u] -
48                        pot[v]){
49                        dis[v] = d + w + pot[u] - pot[v] ;
50                        preE[v] = i ;
51                        pq.push({dis[v], v}) ;
52                    }
53                }
54
55                return dis[t] != INF ;
56            }
57
58            while(dijkstra()){
59                for ( int v=1 ; v<=N ; v++ ) if(dis[v]
60                    < INF){
61                        pot[v] += dis[v] ;
62                    }
63            }
64        }
65    }
66
67    int run(){
68        int ans = 0 ;
69        while(bfs()){
70            ans += dfs(S, 0x3f3f3f3f) ;
71        }
72    }
73
74    int run(){
75        int ans = 0 ;
76        while(bfs()){
77            ans += dfs(S, 0x3f3f3f3f) ;
78        }
79    }
80
81    int run(){
82        int ans = 0 ;
83        while(bfs()){
84            ans += dfs(S, 0x3f3f3f3f) ;
85        }
86    }
87
88    int run(){
89        int ans = 0 ;
90        while(bfs()){
91            ans += dfs(S, 0x3f3f3f3f) ;
92        }
93    }
94
95    int run(){
96        int ans = 0 ;
97        while(bfs()){
98            ans += dfs(S, 0x3f3f3f3f) ;
99        }
100    }
101
102    int run(){
103        int ans = 0 ;
104        while(bfs()){
105            ans += dfs(S, 0x3f3f3f3f) ;
106        }
107    }
108
109    int run(){
110        int ans = 0 ;
111        while(bfs()){
112            ans += dfs(S, 0x3f3f3f3f) ;
113        }
114    }
115
116    int run(){
117        int ans = 0 ;
118        while(bfs()){
119            ans += dfs(S, 0x3f3f3f3f) ;
120        }
121    }
122
123    int run(){
124        int ans = 0 ;
125        while(bfs()){
126            ans += dfs(S, 0x3f3f3f3f) ;
127        }
128    }
129
130    int run(){
131        int ans = 0 ;
132        while(bfs()){
133            ans += dfs(S, 0x3f3f3f3f) ;
134        }
135    }
136
137    int run(){
138        int ans = 0 ;
139        while(bfs()){
140            ans += dfs(S, 0x3f3f3f3f) ;
141        }
142    }
143
144    int run(){
145        int ans = 0 ;
146        while(bfs()){
147            ans += dfs(S, 0x3f3f3f3f) ;
148        }
149    }
150
151    int run(){
152        int ans = 0 ;
153        while(bfs()){
154            ans += dfs(S, 0x3f3f3f3f) ;
155        }
156    }
157
158    int run(){
159        int ans = 0 ;
160        while(bfs()){
161            ans += dfs(S, 0x3f3f3f3f) ;
162        }
163    }
164
165    int run(){
166        int ans = 0 ;
167        while(bfs()){
168            ans += dfs(S, 0x3f3f3f3f) ;
169        }
170    }
171
172    int run(){
173        int ans = 0 ;
174        while(bfs()){
175            ans += dfs(S, 0x3f3f3f3f) ;
176        }
177    }
178
179    int run(){
180        int ans = 0 ;
181        while(bfs()){
182            ans += dfs(S, 0x3f3f3f3f) ;
183        }
184    }
185
186    int run(){
187        int ans = 0 ;
188        while(bfs()){
189            ans += dfs(S, 0x3f3f3f3f) ;
190        }
191    }
192
193    int run(){
194        int ans = 0 ;
195        while(bfs()){
196            ans += dfs(S, 0x3f3f3f3f) ;
197        }
198    }
199
200    int run(){
201        int ans = 0 ;
202        while(bfs()){
203            ans += dfs(S, 0x3f3f3f3f) ;
204        }
205    }
206
207    int run(){
208        int ans = 0 ;
209        while(bfs()){
210            ans += dfs(S, 0x3f3f3f3f) ;
211        }
212    }
213
214    int run(){
215        int ans = 0 ;
216        while(bfs()){
217            ans += dfs(S, 0x3f3f3f3f) ;
218        }
219    }
220
221    int run(){
222        int ans = 0 ;
223        while(bfs()){
224            ans += dfs(S, 0x3f3f3f3f) ;
225        }
226    }
227
228    int run(){
229        int ans = 0 ;
230        while(bfs()){
231            ans += dfs(S, 0x3f3f3f3f) ;
232        }
233    }
234
235    int run(){
236        int ans = 0 ;
237        while(bfs()){
238            ans += dfs(S, 0x3f3f3f3f) ;
239        }
240    }
241
242    int run(){
243        int ans = 0 ;
244        while(bfs()){
245            ans += dfs(S, 0x3f3f3f3f) ;
246        }
247    }
248
249    int run(){
250        int ans = 0 ;
251        while(bfs()){
252            ans += dfs(S, 0x3f3f3f3f) ;
253        }
254    }
255
256    int run(){
257        int ans = 0 ;
258        while(bfs()){
259            ans += dfs(S, 0x3f3f3f3f) ;
260        }
261    }
262
263    int run(){
264        int ans = 0 ;
265        while(bfs()){
266            ans += dfs(S, 0x3f3f3f3f) ;
267        }
268    }
269
270    int run(){
271        int ans = 0 ;
272        while(bfs()){
273            ans += dfs(S, 0x3f3f3f3f) ;
274        }
275    }
276
277    int run(){
278        int ans = 0 ;
279        while(bfs()){
280            ans += dfs(S, 0x3f3f3f3f) ;
281        }
282    }
283
284    int run(){
285        int ans = 0 ;
286        while(bfs()){
287            ans += dfs(S, 0x3f3f3f3f) ;
288        }
289    }
290
291    int run(){
292        int ans = 0 ;
293        while(bfs()){
294            ans += dfs(S, 0x3f3f3f3f) ;
295        }
296    }
297
298    int run(){
299        int ans = 0 ;
300        while(bfs()){
301            ans += dfs(S, 0x3f3f3f3f) ;
302        }
303    }
304
305    int run(){
306        int ans = 0 ;
307        while(bfs()){
308            ans += dfs(S, 0x3f3f3f3f) ;
309        }
310    }
311
312    int run(){
313        int ans = 0 ;
314        while(bfs()){
315            ans += dfs(S, 0x3f3f3f3f) ;
316        }
317    }
318
319    int run(){
320        int ans = 0 ;
321        while(bfs()){
322            ans += dfs(S, 0x3f3f3f3f) ;
323        }
324    }
325
326    int run(){
327        int ans = 0 ;
328        while(bfs()){
329            ans += dfs(S, 0x3f3f3f3f) ;
330        }
331    }
332
333    int run(){
334        int ans = 0 ;
335        while(bfs()){
336            ans += dfs(S, 0x3f3f3f3f) ;
337        }
338    }
339
340    int run(){
341        int ans = 0 ;
342        while(bfs()){
343            ans += dfs(S, 0x3f3f3f3f) ;
344        }
345    }
346
347    int run(){
348        int ans = 0 ;
349        while(bfs()){
350            ans += dfs(S, 0x3f3f3f3f) ;
351        }
352    }
353
354    int run(){
355        int ans = 0 ;
356        while(bfs()){
357            ans += dfs(S, 0x3f3f3f3f) ;
358        }
359    }
360
361    int run(){
362        int ans = 0 ;
363        while(bfs()){
364            ans += dfs(S, 0x3f3f3f3f) ;
365        }
366    }
367
368    int run(){
369        int ans = 0 ;
370        while(bfs()){
371            ans += dfs(S, 0x3f3f3f3f) ;
372        }
373    }
374
375    int run(){
376        int ans = 0 ;
377        while(bfs()){
378            ans += dfs(S, 0x3f3f3f3f) ;
379        }
380    }
381
382    int run(){
383        int ans = 0 ;
384        while(bfs()){
385            ans += dfs(S, 0x3f3f3f3f) ;
386        }
387    }
388
389    int run(){
390        int ans = 0 ;
391        while(bfs()){
392            ans += dfs(S, 0x3f3f3f3f) ;
393        }
394    }
395
396    int run(){
397        int ans = 0 ;
398        while(bfs()){
399            ans += dfs(S, 0x3f3f3f3f) ;
400        }
401    }
402
403    int run(){
404        int ans = 0 ;
405        while(bfs()){
406            ans += dfs(S, 0x3f3f3f3f) ;
407        }
408    }
409
410    int run(){
411        int ans = 0 ;
412        while(bfs()){
413            ans += dfs(S, 0x3f3f3f3f) ;
414        }
415    }
416
417    int run(){
418        int ans = 0 ;
419        while(bfs()){
420            ans += dfs(S, 0x3f3f3f3f) ;
421        }
422    }
423
424    int run(){
425        int ans = 0 ;
426        while(bfs()){
427            ans += dfs(S, 0x3f3f3f3f) ;
428        }
429    }
430
431    int run(){
432        int ans = 0 ;
433        while(bfs()){
434            ans += dfs(S, 0x3f3f3f3f) ;
435        }
436    }
437
438    int run(){
439        int ans = 0 ;
440        while(bfs()){
441            ans += dfs(S, 0x3f3f3f3f) ;
442        }
443    }
444
445    int run(){
446        int ans = 0 ;
447        while(bfs()){
448            ans += dfs(S, 0x3f3f3f3f) ;
449        }
450    }
451
452    int run(){
453        int ans = 0 ;
454        while(bfs()){
455            ans += dfs(S, 0x3f3f3f3f) ;
456        }
457    }
458
459    int run(){
460        int ans = 0 ;
461        while(bfs()){
462            ans += dfs(S, 0x3f3f3f3f) ;
463        }
464    }
465
466    int run(){
467        int ans = 0 ;
468        while(bfs()){
469            ans += dfs(S, 0x3f3f3f3f) ;
470        }
471    }
472
473    int run(){
474        int ans = 0 ;
475        while(bfs()){
476            ans += dfs(S, 0x3f3f3f3f) ;
477        }
478    }
479
480    int run(){
481        int ans = 0 ;
482        while(bfs()){
483            ans += dfs(S, 0x3f3f3f3f) ;
484        }
485    }
486
487    int run(){
488        int ans = 0 ;
489        while(bfs()){
490            ans += dfs(S, 0x3f3f3f3f) ;
491        }
492    }
493
494    int run(){
495        int ans = 0 ;
496        while(bfs()){
497            ans += dfs(S, 0x3f3f3f3f) ;
498        }
499    }
500
501    int run(){
502        int ans = 0 ;
503        while(bfs()){
504            ans += dfs(S, 0x3f3f3f3f) ;
505        }
506    }
507
508    int run(){
509        int ans = 0 ;
510        while(bfs()){
511            ans += dfs(S, 0x3f3f3f3f) ;
512        }
513    }
514
515    int run(){
516        int ans = 0 ;
517        while(bfs()){
518            ans += dfs(S, 0x3f3f3f3f) ;
519        }
520    }
521
522    int run(){
523        int ans = 0 ;
524        while(bfs()){
525            ans += dfs(S, 0x3f3f3f3f) ;
526        }
527    }
528
529    int run(){
530        int ans = 0 ;
531        while(bfs()){
532            ans += dfs(S, 0x3f3f3f3f) ;
533        }
534    }
535
536    int run(){
537        int ans = 0 ;
538        while(bfs()){
539            ans += dfs(S, 0x3f3f3f3f) ;
540        }
541    }
542
543    int run(){
544        int ans = 0 ;
545        while(bfs()){
546            ans += dfs(S, 0x3f3f3f3f) ;
547        }
548    }
549
550    int run(){
551        int ans = 0 ;
552        while(bfs()){
553            ans += dfs(S, 0x3f3f3f3f) ;
554        }
555    }
556
557    int run(){
558        int ans = 0 ;
559        while(bfs()){
560            ans += dfs(S, 0x3f3f3f3f) ;
561        }
562    }
563
564    int run(){
565        int ans = 0 ;
566        while(bfs()){
567            ans += dfs(S, 0x3f3f3f3f) ;
568        }
569    }
570
571    int run(){
572        int ans = 0 ;
573        while(bfs()){
574            ans += dfs(S, 0x3f3f3f3f) ;
575        }
576    }
577
578    int run(){
579        int ans = 0 ;
580        while(bfs()){
581            ans += dfs(S, 0x3f3f3f3f) ;
582        }
583    }
584
585    int run(){
586        int ans = 0 ;
587        while(bfs()){
588            ans += dfs(S, 0x3f3f3f3f) ;
589        }
590    }
591
592    int run(){
593        int ans = 0 ;
594        while(bfs()){
595            ans += dfs(S, 0x3f3f3f3f) ;
596        }
597    }
598
599    int run(){
600        int ans = 0 ;
601        while(bfs()){
602            ans += dfs(S, 0x3f3f3f3f) ;
603        }
604    }
605
606    int run(){
607        int ans = 0 ;
608        while(bfs()){
609            ans += dfs(S, 0x3f3f3f3f) ;
610        }
611    }
612
613    int run(){
614        int ans = 0 ;
615        while(bfs()){
616            ans += dfs(S, 0x3f3f3f3f) ;
617        }
618    }
619
620    int run(){
621        int ans = 0 ;
622        while(bfs()){
623            ans += dfs(S, 0x3f3f3f3f) ;
624        }
625    }
626
627    int run(){
628        int ans = 0 ;
629        while(bfs()){
630            ans += dfs(S, 0x3f3f3f3f) ;
631        }
632    }
633
634    int run(){
635        int ans = 0 ;
636        while(bfs()){
637            ans += dfs(S, 0x3f3f3f3f) ;
638        }
639    }
640
641    int run(){
642        int ans = 0 ;
643        while(bfs()){
644            ans += dfs(S, 0x3f3f3f3f) ;
645        }
646    }
647
648    int run(){
649        int ans = 0 ;
650        while(bfs()){
651            ans += dfs(S, 0x3f3f3f3f) ;
652        }
653    }
654
655    int run(){
656        int ans = 0 ;
657        while(bfs()){
658            ans += dfs(S, 0x3f3f3f3f) ;
659        }
660    }
661
662    int run(){
663        int ans = 0 ;
664        while(bfs()){
665            ans += dfs(S, 0x3f3f3f3f) ;
666        }
667    }
668
669    int run(){
670        int ans = 0 ;
671        while(bfs()){
672            ans += dfs(S, 0x3f3f3f3f) ;
673        }
674    }
675
676    int run(){
677        int ans = 0 ;
678        while(bfs()){
679            ans += dfs(S, 0x3f3f3f3f) ;
680        }
681    }
682
683    int run(){
684        int ans = 0 ;
685        while(bfs()){
686            ans += dfs(S, 0x3f3f3f3f) ;
687        }
688    }
689
690    int run(){
691        int ans = 0 ;
692        while(bfs()){
693            ans += dfs(S, 0x3f3f3f3f) ;
694        }
695    }
696
697    int run(){
698        int ans = 0 ;
699        while(bfs()){
700            ans += dfs(S, 0x3f3f3f3f) ;
701        }
702    }
703
704    int run(){
705        int ans = 0 ;
706        while(bfs()){
707            ans += dfs(S, 0x3f3f3f3f) ;
708        }
709    }
710
711    int run(){
712        int ans = 0 ;
713        while(bfs()){
714            ans += dfs(S, 0x3f3f3f3f) ;
715        }
716    }
717
718    int run(){
719        int ans = 0 ;
720        while(bfs()){
721            ans += dfs(S, 0x3f3f3f3f) ;
722        }
723    }
724
725    int run(){
726        int ans = 0 ;
727        while(bfs()){
728            ans += dfs(S, 0x3f3f3f3f) ;
729        }
730    }
731
732    int run(){
733        int ans = 0 ;
734        while(bfs()){
735            ans += dfs(S, 0x3f3f3f3f) ;
736        }
737    }
738
739    int run(){
740        int ans = 0 ;
741        while(bfs()){
742            ans += dfs(S, 0x3f3f3f3f) ;
743        }
744    }
745
746    int run(){
747        int ans = 0 ;
748        while(bfs()){
749            ans += dfs(S, 0x3f3f3f3f) ;
750        }
751    }
752
753    int run(){
754        int ans = 0 ;
755        while(bfs()){
756            ans += dfs(S, 0x3f3f3f3f) ;
757        }
758    }
759
760    int run(){
761        int ans = 0 ;
762        while(bfs()){
763            ans += dfs(S, 0x3f3f3f3f) ;
764        }
765    }
766
767    int run(){
768        int ans = 0 ;
769        while(bfs()){
770            ans += dfs(S, 0x3f3f3f3f) ;
771        }
772    }
773
774    int run(){
775        int ans = 0 ;
776        while(bfs()){
777            ans += dfs(S, 0x3f3f3f3f) ;
778        }
779    }
780
781    int run(){
782        int ans = 0 ;
783        while(bfs()){
784            ans += dfs(S, 0x3f3f3f3f) ;
785        }
786    }
787
788    int run(){
789        int ans = 0 ;
790        while(bfs()){
791            ans += dfs(S, 0x3f3f3f3f) ;
792        }
793    }
794
795    int run(){
796        int ans = 0 ;
797        while(bfs()){
798            ans += dfs(S, 0x3f3f3f3f) ;
799        }
800    }
801
802    int run(){
803        int ans = 0 ;
804        while(bfs()){
805            ans += dfs(S, 0x3f3f3f3f) ;
806        }
807    }
808
809    int run(){
810        int ans = 0 ;
811        while(bfs()){
812            ans += dfs(S, 0x3f3f3f3f) ;
813        }
814    }
815
816    int run(){
817        int ans = 0 ;
818        while(bfs()){
819            ans += dfs(S, 0x3f3f3f3f) ;
820        }
821    }
822
823    int run(){
824        int ans = 0 ;
825        while(bfs()){
826            ans += dfs(S, 0x3f3f3f3f) ;
827        }
828    }
829
830    int run(){
831        int ans = 0 ;
832        while(bfs()){
833            ans += dfs(S, 0x3f3f3f3f) ;
834        }
835    }
836
837    int run(){
838        int ans = 0 ;
839        while(bfs()){
840            ans += dfs(S, 0x3f3f3f3f) ;
841        }
842    }
843
844    int run(){
845        int ans = 0 ;
846        while(bfs()){
847            ans += dfs(S, 0x3f3f3f3f) ;
848        }
849    }
850
851    int run(){
852        int ans = 0 ;
853        while(bfs()){
854            ans += dfs(S, 0x3f3f3f3f) ;
855        }
856    }
857
858    int run(){
859        int ans = 0 ;
860        while(bfs()){
861            ans += dfs(S, 0x3f3f3f3f) ;
862        }
863    }
864
865    int run(){
866        int ans = 0 ;
867        while(bfs()){
868            ans += dfs(S, 0x3f3f3f3f) ;
869        }
870    }
871
872    int run(){
873        int ans = 0 ;
874        while(bfs()){
875            ans += dfs(S, 0x3f3f3f3f) ;
876        }
877    }
878
879    int run(){
880        int ans = 0 ;
881        while(bfs()){
882            ans += dfs(S, 0x3f3f3f3f) ;
883        }
884    }
885
886    int run(){
887        int ans = 0 ;
888        while(bfs()){
889            ans += dfs(S, 0x3f3f3f3f) ;
890        }
891    }
892
893    int run(){
894        int ans = 0 ;
895        while(bfs()){
896            ans += dfs(S, 0x3f3f3f3f) ;
897        }
898    }
899
900    int run(){
901        int ans = 0 ;
902        while(bfs()){
903            ans += dfs(S, 0x3f3f3f3f) ;
904        }
905    }
906
907    int run(){
908        int ans = 0 ;
909        while(bfs()){
910            ans += dfs(S, 0x3f3f3f3f) ;
911        }
912    }
913
914    int run(){
915        int ans = 0 ;
916        while(bfs()){
917            ans += dfs(S, 0x3f3f3f3f) ;
918        }
919    }
920
921    int run(){
922        int ans = 0 ;
923        while(bfs()){
924            ans += dfs(S, 0x3f3f3f3f) ;
925        }
926    }
927
928    int run(){
929        int ans = 0 ;
930        while(bfs()){
931            ans += dfs(S, 0x3f3f3f3f) ;
932        }
933    }
934
935    int run(){
936        int ans = 0 ;
937        while(bfs()){
938            ans += dfs(S, 0x3f3f3f3f) ;
939        }
940    }
941
942    int run(){
943        int ans = 0 ;
944        while(bfs()){
945            ans += dfs(S, 0x3f3f3f3f) ;
946        }
947    }
948
949    int run(){
950        int ans = 0 ;
951        while(bfs()){
952            ans += dfs(S, 0x3f3f3f3f) ;
953        }
954    }
955
956    int run(){
957        int ans = 0 ;
958        while(bfs()){
959            ans += dfs(S, 0x3f3f3f3f) ;
960        }
961    }
962
963    int run(){
964        int ans = 0 ;
965        while(bfs()){
966            ans += dfs(S, 0x3f3f3f3f) ;
967        }
968    }
969
970    int run(){
971        int ans = 0 ;
972        while(bfs()){
973            ans += dfs(S, 0x3f3f3f3f) ;
974        }
975    }
976
977    int run(){
978        int ans = 0 ;
979        while(bfs()){
980            ans += dfs(S, 0x3f3f3f3f) ;
981        }
982    }
983
984    int run(){
985        int ans = 0 ;
986        while(bfs()){
987            ans += dfs(S, 0x3f3f3f3f) ;
988        }
989    }
990
991    int run(){
992        int ans = 0 ;
993        while(bfs()){
994            ans += dfs(S, 0x3f3f3f3f) ;
995        }
996    }
997
998    int run(){
999        int ans = 0 ;
1000       while(bfs()){
1001            ans += dfs(S, 0x3f3f3f3f) ;
1002        }
1003
1004    int run(){
1005        int ans = 0 ;
1006        while(bfs()){
1007            ans += dfs(S, 0x3f3f3f3f) ;
1008        }
1009    }
1010
1011    int run(){
1012        int ans = 0 ;
1013        while(bfs()){
1014            ans += dfs(S, 0x3f3f3f3f) ;
1015        }
1016    }
1017
1018    int run(){
1019        int ans = 0 ;
1020        while(bfs()){
1021            ans += dfs(S, 0x3f3f3f3f) ;
1022        }
1023    }
1024
1025    int run(){
1026        int ans = 0 ;
1027        while(bfs()){
1028            ans += dfs(S, 0x3f3f3f3f) ;
1029        }
1030    }
1031
1032    int run(){
1033        int ans = 0 ;
1034        while(bfs()){
1035            ans += dfs(S, 0x3f3f3f3f) ;
1036        }
1037    }
1038
1039    int run(){
1040        int ans = 0 ;
1041        while(bfs()){
1042            ans += dfs(S, 0x3f3f3f3f) ;
1043        }
1044    }
1045
1046    int run(){
1047        int ans = 0 ;
1048        while(bfs()){
1049            ans += dfs(S, 0x3f3f3f3f) ;
1050        }
1051    }
1052
1053    int run(){
1054        int ans = 0 ;
1055        while(bfs()){
1056            ans += dfs(S, 0x3f3f3f3f) ;
1057        }
1058    }
1059
1060    int run(){
1061        int ans = 0 ;
1062        while(bfs()){
1063            ans += dfs(S, 0x3f3f3f3f) ;
1064        }
1065    }
1066
1067    int run(){
1068        int ans = 0 ;
1069        while(bfs()){
1070            ans += dfs(S, 0x3f3f3f3f) ;
1071        }
1072    }
1073
1074    int run(){
1075        int ans = 0 ;
1076        while(bfs()){
1077            ans += dfs(S, 0x3f3f3f3f) ;
1078        }
1079    }
1080
1081    int run(){
1082        int ans = 0 ;
1083        while(bfs()){
1084            ans += dfs(S, 0x3f3f3f3f) ;
1085        }
1086    }
1087
1088    int run(){
1089        int ans = 0 ;
1090        while(bfs()){
1091            ans += dfs(S, 0x3f3f3f3f) ;
1092        }
1093    }
1094
1095    int run(){
1096        int ans = 0 ;
1097        while(bfs()){
1098            ans += dfs(S, 0x3f3f3f3f) ;
1099        }
1100    }
1101
1102    int run(){
1103        int ans = 0 ;
1104        while(bfs()){
1105            ans += dfs(S, 0x3f3f3f3f) ;
1106        }
1107    }
1108
1109    int run(){
1110        int ans = 0 ;
1111        while(bfs()){
1112            ans += dfs(S, 0x3f3f3f3f) ;
1113        }
1114    }
1115
1116    int run(){
1117        int ans = 0 ;
1118        while(bfs()){
1119            ans += dfs(S, 0x3f3f3f3f) ;
1120        }
1121    }
1122
1123    int run(){
1124        int ans = 0 ;
1125        while(bfs()){
1126            ans += dfs(S, 0x3f3f3f3f) ;
1127        }
1128    }
1129
1130    int run(){
1131        int ans = 0 ;
1132        while(bfs()){
1133            ans += dfs(S, 0x3f3f3f3f) ;
1134        }
1135    }
1136
1137    int run(){
1138        int ans = 0 ;
1139        while(bfs()){
1140            ans += dfs(S, 0x3f3f3f3f) ;
1141        }
1142    }
1143
1144    int run(){
1145        int ans = 0 ;
1146        while(bfs()){
1147            ans += dfs(S, 0x3f3f3f3f) ;
1148        }
1149    }
1150
1151    int run(){
1152        int ans = 0 ;
1153        while(bfs()){
1154            ans += dfs(S, 0x3f3f3f3f) ;
1155        }
1156    }
1157
1158    int run(){
1159        int ans = 0 ;
1160        while(bfs()){
1161            ans += dfs(S, 0x3f3f3f3f) ;
1162        }
1163    }
1164
1165    int run(){
1166        int ans = 0 ;
1167        while(bfs()){
1168            ans += dfs(S, 0x3f3f3f3f) ;
1169        }
1170    }
1171
1172    int run(){
1173        int ans = 0 ;
1174        while(bfs()){
1175            ans += dfs(S, 0x3f3f3f3f) ;
1176        }
1177    }
1178
1179    int run(){
1180        int ans = 0 ;
1181        while(bfs()){
1182            ans += dfs(S, 0x3f3f3f3f) ;
1183        }
1184    }
1185
1186    int run(){
1187        int ans = 0 ;
1188        while(bfs()){
1189            ans += dfs(S, 0x3f3f3f3f) ;
1190        }
1191    }
1192
1193    int run(){
1194        int ans = 0 ;
1195        while(bfs()){
1196            ans += dfs(S, 0x3f3f3f3f) ;
1197        }
1198    }
1199
1200    int run(){
1201        int ans = 0 ;
1202        while(bfs()){
1203            ans += dfs(S, 0x3f3f3f3f) ;
1204        }
1205    }
1206
1207    int run(){
1208        int ans = 0 ;
1209        while(bfs()){
1210            ans += dfs(S, 0x3f3f3f3f) ;
1211        }
1212    }
1213
1214    int run(){
1215        int ans = 0 ;
1216        while(bfs()){
1217            ans += dfs(S, 0x3f3f3f3f) ;
1218        }
1219    }
1220
1221    int run(){
1222        int ans = 0 ;
1223        while(bfs()){
1224            ans += dfs(S, 0x3f3f3f3f) ;
1225        }
1226    }
1227
1228    int run(){
1229        int ans = 0 ;
1230        while(bfs()){
1231            ans += dfs(S, 0x3f3f3f3f) ;
1232        }
1233    }
1234
1235    int run(){
1236        int ans = 0 ;
1237        while(bfs()){
1238            ans += dfs(S, 0x3f3f3f3f) ;
1239        }
1240    }
1241
1242    int run(){
1243        int ans = 0 ;
1244        while(bfs()){
1245            ans += dfs(S, 0x3f3
```

```

78     ans[id] = vis[end[id]] ;
79 }
80
81     return ans ;
82 }
83 };

```

6 Techniques

6.1 二分搜

```

1 // xxxxxxxx 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     }
8
9     return l ;
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 l ;
21 }
22
23 // 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]){

```

```

36         u = up[u][i] ;
37         v = up[v][i] ;
38     }
39
40     return up[u][0] ;
41 }
42
43 int main(){
44     int n, q, root ;
45     scanf("%d%d%d", &n, &q, &root) ;
46     MaxLog = __lg(n) ;
47
48     for ( int i=0 ; i<n-1 ; i++ ){
49         int u, v ;
50         scanf("%d%d", &u, &v) ;
51         e[u].push_back(v) ;
52         e[v].push_back(u) ;
53     }
54
55     dfs(root, root, 0) ;
56
57     while(q--){
58         int u, v ;
59         scanf("%d%d", &u, &v) ;
60         printf("%d\n", lca(u, v)) ;
61     }
62
63     return 0 ;
64 }

```

6.3 SG

```

1 long long SG(long long k){
2
3     if(k % 2 == 0){
4         return k / 2;
5     }
6     else{
7         return SG(k / 2);
8     }
9 }
10
11 int main(){
12     int cas, n;
13
14     scanf("%d", &cas);
15     while(cas--){
16         scanf("%d", &n);
17
18         long long s, v = 0;
19
20         for(int i = 0; i < n; i++){
21             scanf("%lld", &s);
22             v ^= SG(s); //XOR
23         }
24
25         if(v) printf("YES\n");
26         else printf("NO\n");
27     }
28
29     int SG[30] ;
30     int vis[Maxn], stone[Maxn] ;
31
32     void build(){
33         SG[0] = 0 ;
34         memset(vis, 0, sizeof(vis)) ;
35
36         for ( int i=1 ; i<30 ; i++ ){
37             int cur = 0 ;
38             for ( int j=0 ; j<i ; j++ ) for ( int
39                 k=0 ; k<=j ; k++ ){
40                 vis[SG[j] ^ SG[k]] = i ;
41             }
42             while(vis[cur] == i) cur++ ;
43             SG[i] = cur ;
44         }
45     }
46 }
47
48 int main(){
49     build() ;
50
51     int T = 0 ;
52     while(~scanf("%d", &n) && n){
53         int ans = 0 ;
54
55         for ( int i=1 ; i<=n ; i++ ) scanf("%d",
56             &stone[i]) ;
57
58         for ( int i=1 ; i<=n ; i++ ) if(stone[i]
59             & 1){
60             ans ^= SG[n-i] ;
61         }
62     }
63 }

```

7 DP

7.1 輪廓線 DP

```

1 #include <bits/stdc++.h>
2
3 using namespace std ;
4 using ll = long long ;
5
6 ll 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 }

```

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 int count(int n){
30    memset(dp, -1, sizeof(dp)) ;
31    dig.clear() ;
32
33    while(n > 0){
34        dig.push_back(n % 10) ;
35        n /= 10 ;
36    }
37
38    return solve(dig.size() - 1, 0, 0, 1) ;
39
40 int main(){
41    int T ;
42    scanf("%d", &T) ;
43
44    int a, b ;
45    while(T--){
46        scanf("%d%d%d", &a, &b, &K) ;
47        if(K > 90) printf("0\n") ;
48        else printf("%d\n", count(b) -
49                    count(a-1)) ;
50    }
51
52    return 0 ;
53}
54
55 DFS(v) ;
56 cnt[u] += cnt[v] ;
57
58 dp[u][1][0] = dp[u][1][1] = 0 ;
59
60 for ( auto [v, w] : edge[u] ){
61    for ( int i=cnt[u] ; i>1 ; i-- ) for (
62        int j=1 ; j<i && j<=cnt[v] ; j++ )
63        dp[u][i][1] = min(dp[u][i][1],
64                            dp[u][i-j][1] + dp[v][j][1] + 2 *
65                            w) ;
66    dp[u][i][0] = min(dp[u][i][0],
67                        dp[u][i-j][1] + dp[v][j][0] + w) ;
68    dp[u][i][0] = min(dp[u][i][0],
69                        dp[u][i-j][0] + dp[v][j][1] + 2 *
70                        w) ;
71
72    }
73
74    }
75
76 int main(){
77    int t = 0 ;
78
79    while(~scanf("%d", &n) && n){
80        init() ;
81        for ( int i=0 ; i<n-1 ; i++ ){
82            int u, v, w ;
83            scanf("%d%d%d", &v, &u, &w) ;
84            edge[u].push_back({v, w}) ;
85        }
86
87        DFS(0) ;
88        printf("Case %d:\n", ++t) ;
89
90        int q, e ;
91        scanf("%d", &q) ;
92
93        while(q--){
94            scanf("%d", &e) ;
95
96            for ( int i=n ; i>=1 ; i-- )
97                if(dp[0][i][0] <= e){
98                    printf("%d\n", i) ;
99                    break ;
100                }
101            }
102        }
103
104        return 0 ;
105    }

```

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] ) if ( !cnt[v] ) DFS(v) ;
27 }
```