

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 AND Number Theory | 2 |
| 2.1 | formula | 2 |
| 2.2 | extended gcd | 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 ver1 | 5 |
| 3.7 | BIT ver2 | 5 |
| 3.8 | BIT ver3 | 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 | 9 |
| 7.1 | 輪廓線 DP | 9 |
| 7.2 | 數位 DP | 9 |
| 7.3 | 樹 DP | 9 |

1 Foundations

1.1 PyMath

```
import math

math.ceil(x) #上高斯
math.floor(x) #下高斯
math.factorial(x) #階乘
math.fabs(x) #絕對值
math.fsum(arr) #求和
math.gcd(x, y)
math.exp(x) # e^x
math.log(x, base)
math.log2(x)
math.log10(x)
math.sqrt(x)
math.pow(x, y, mod)
math.sin(x) # cos, tan, asin, acos, atan,
            atan2, sinh ...
math.hypot(x, y) #歐幾里德範數
math.degrees(x) #x從弧度轉角度
math.radians(x) #x從角度轉弧度
math.gamma(x) #x的gamma函數
math.pi #const
math.e #const
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
    radix)
9 static Integer valueOf(int i)
10 static Integer valueOf(String s)
11 static Integer valueOf(String s, int radix)
12 static String toString(int i)
13 static String toString(int i, int radix)
14 static String toUnsignedString(int i)
15 static String toUnsignedString(int i, int
    radix)
16 static long toUnsignedLong(int x)
17 static Integer decode(String nm)
    // 支援 0x/0/# 前綴
18 static Integer getInteger(String nm[, int
    val]) // 從系統屬性讀取整數
19
20 // 比較/雜湊/聚合
21 static int compare(int x, int y)
22 static int compareUnsigned(int x, int y)
23 static int hashCode(int value)
24 static int min(int a, int b)
25 static int max(int a, int b)
26 static int sum(int a, int b)
27
28 // 位元操作
29 static int bitCount(int i) // 設定位數
30 static int highestOneBit(int i)
31 static int lowestOneBit(int i)
32 static int numberOfLeadingZeros(int i)
33 static int numberOfTrailingZeros(int i)
34 static int rotateLeft(int i, int distance)
35 static int rotateRight(int i, int distance)
36 static int reverse(int i)
37 static int reverseBytes(int i)
38
39 // 無號運算
40 static int divideUnsigned(int dividend, int
    divisor)
```

```
41 static int remainderUnsigned(int dividend,
    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
    endIndex)
9 boolean contains(CharSequence s)
10 boolean startsWith(String prefix[, int
    toffset])
11 boolean endsWith(String suffix)
12 int indexOf(String str[, int fromIndex])
13 int lastIndexOf(String str[, int
    fromIndex])
14
15 // 取子字串/子序列
16 String substring(int beginIndex)
17 String substring(int beginIndex, int
    endIndex)
18 CharSequence subSequence(int beginIndex, int
    endIndex)
19
20 // 比較/等價
21 boolean equals(Object obj)
22 boolean equalsIgnoreCase(String
    anotherString)
23 int compareTo(String anotherString)
24 int compareToIgnoreCase(String str)
25 boolean matches(String regex)
26 boolean regionMatches(int toffset, String
    other, int ooffset, int len)
27 boolean regionMatches(boolean ignoreCase,
    int toffset, String other, int ooffset,
    int len)
28
29 // 建構/轉換/連接
30 String concat(String str)
31 String replace(char oldChar, char newChar)
32 String replace(CharSequence target,
    CharSequence replacement)
33 String replaceAll(String regex, String
    replacement)
34 String replaceFirst(String regex, String
    replacement)
35 String[] split(String regex[, int limit])
36 String toLowerCase()
37 String toUpperCase()
38 String trim()
39 String strip() // (since 11)
40 String stripLeading() // (since 11)
41 String stripTrailing() // (since 11)
42 String repeat(int count) // (since 11)
43 InputStream chars()
44 Stream<String> lines() // (since 11)
45 String intern()
46
47 // 靜態工具
48 static String format(String format,
    Object... args)
49 static String join(CharSequence delimiter,
    CharSequence... elements)
50 static String join(CharSequence delimiter,
    Iterable<? extends CharSequence>
    elements)
51 static String
    valueOf(primitive/char[]/Object)
52 static String copyValueOf(char[] data[, int
    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 // 絕對值/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 // 指數對/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)
```

2 Mathematics AND Number Theory

2.1 formula

1. Catalan Number

$$C_n = \frac{1}{n} \binom{2n}{n}, C_{n+1} = \frac{2(2n+1)}{n+2} C_n$$

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

2. Euler’s Formula

對於 v 個點, e 條邊, f 個面, c 個連通分量

$$V + F = E + 2$$
$$V + F = E + C + 1$$

3. Pick’s Theorem

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

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

4. T1

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<long long, long long> crt(long long a1, long long m1, long long a2, long long m2) {
4     long long g = std::gcd(m1, m2);
5     if ((a2 - a1) % g != 0) return {0, -1}; // no solution
6
7     long long lcm = m1 / g * m2;
8     long long m1_reduced = m1 / g;
9     long long m2_reduced = m2 / g;
10
11     long long diff = (a2 - a1) / g % m2_reduced;
12     if (diff < 0) diff += m2_reduced;
13
14     long long inv = mod_pow(m1_reduced, m2_reduced - 1, m2_reduced);
15     long long step = diff * inv % m2_reduced;
16     long long x0 = (a1 + step * m1) % lcm;
17     if (x0 < 0) x0 += lcm;
18     return {x0, lcm};
19 }
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。

5. T2

Combinatorics Cheat Sheet

Binomial Coefficient Identities

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

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：

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

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。

6. T4

Number Theory Essentials

Fermat's Little Theorem: $a^{p-1} \equiv 1 \pmod{p} \quad (\gcd(a, p) = 1)$

Euler's Theorem: $a^{\varphi(n)} \equiv 1 \pmod{n} \quad (\gcd(a, n) = 1)$

Modular Inverse: $a^{-1} \equiv a^{p-2} \pmod{p} \quad (\gcd(a, p) = 1)$

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

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

其中 μ 為莫比烏斯函數；常用於反演及計算互質對數量。

2.2 extended gcd

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 }
```

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
        {return w > o.w ;} ;
5 } ;
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 }
38
39 void MST(){
40     int cnt = 0 ;
41     for ( int i=0 ; i<edge.size() && cnt < n-1
        ; i++ ){
42         auto [u, v, w] = edge[i] ;
43         if(merge(u, v)){
44             cnt++ ;
45             sum += w ;
46         }
47     }
48 }
49
50 int main(){
51     for ( int i=0 ; i<m ; i++ ){
52         scanf("%d%d%d", &u, &v, &w) ;
53         edge.push_back({u, v, w}) ;
54         sum += w ;
55     }
56
57     sort(edge.begin(), edge.end(),
        greater<Edge>()) ;
58     MST() ;
59 }
```

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 +
            seg[rc].sum ;
22     }
23
24     void AddTag(int id, LazyTag &tag){
25         if(tag.type == 0){
26             seg[id].sum += tag.val *
                seg[id].sz ;
27             seg[id].tag.val += tag.val ;
28         }
29         else{
30             seg[id].sum = tag.val *
                seg[id].sz ;
31             seg[id].tag = {1, tag.val} ;
32         }
33     }
34 }
```

```
35 void push(int id){
36     AddTag(lc, seg[id].tag) ;
37     AddTag(rc, seg[id].tag) ;
38     seg[id].tag = {0, 0} ;
39 }
40
41 public:
42 void build(int L=1, int R=n, int id=1){
43     seg[id].sum = 0 ;
44     seg[id].tag = {0, 0} ;
45     seg[id].sz = 1 ;
46
47     if(L == R){
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,
    int L=1, int R=n, int id=1){
61     if(l <= L && R <= r){
62         AddTag(id, tag) ;
63         return ;
64     }
65
66     push(id) ;
67     int M = (L + R) >> 1 ;
68     if(r <= M) modify(l, r, tag, L, M,
        lc) ;
69     else if(l > M) modify(l, r, tag, M+1,
        R, rc) ;
70     else{
71         modify(l, r, tag, L, M, lc) ;
72         modify(l, r, tag, M+1, R, rc) ;
73     }
74     pull(id) ;
75 }
76
77 ll query(int l, int r, int L=1, int R=n,
    int id=1){
78     if(l <= L && R <= r) return
        seg[id].sum ;
79
80     push(id) ;
81     int M = (L + R) >> 1 ;
82     if(r <= M) return query(l, r, L, M,
        lc) ;
83     else if(l > M) return query(l, r,
        M+1, R, rc) ;
84     else return query(l, r, L, M, lc) +
        query(l, r, M+1, R, rc) ;
85 }
86 }tree ;
```

3.3 HLD

```
1 /* HLD */
2 int fa[Maxn], top[Maxn], son[Maxn],
    sz[Maxn], dep[Maxn] = {0}, dfn[Maxn],
    rnk[Maxn], dfsCnt = 0 ;
3
4 void dfs1(int u, int from){
5     fa[u] = from ;
6     dep[u] = dep[from] + 1 ;
7     sz[u] = 1 ;
8
9     for ( auto v : g[u] ) if(v != from){
10         dfs1(v, u) ;
11         sz[u] += sz[v] ;
```

```

12     if(son[u] == -1 || sz[v] > sz[son[u]])
13         son[u] = v ;
14 }
15
16 void dfs2(int u, int t){
17     top[u] = t ;
18     dfn[u] = ++dfscnt ;
19     rnk[dfscnt] = u ;
20
21     if(son[u] == -1) return ;
22
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
52 struct Node{
53     ColorSeg color ;
54     int tag ;
55 }seg[Maxn << 2] ;
56
57 class SegmentTree{
58 private:
59     void pull(int id){
60         // normal pull
61     }
62
63     void AddTag(int id, int tag){
64         // normal AddTag
65     }
66
67     void push(int id){
68         // normal push
69     }
70
71     void modify(int l, int r, int tag, int
72         L=1, int R=n, int id=1){
73         // normal modify
74     }
75
76     ColorSeg query(int l, int r, int L=1, int
77         R=n, int id=1){
78         // normal query
79     }
80 public:
81     void build(int L=1, int R=n, int id=1){
82         // normal build
83     }
84
85     // update val from u to v (simple path)
86     void update(int u, int v, int val){
87         while(top[u] != top[v]){
88             if(dep[top[u]] < dep[top[v]]) swap(u,
89                 v) ;
90             modify(dfn[top[u]], dfn[u], val) ;
91             u = fa[top[u]] ;
92         }
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 ;
98             U = {u, {0, 0, 0}} ;
99             V = {v, {0, 0, 0}} ;
100
101             while(top[U.first] != top[V.first]){
102                 if(dep[top[U.first]] <
103                     dep[top[V.first]]) swap(U, V) ;
104                 U.second = query(dfn[top[U.first]],
105                     dfn[U.first]) + U.second ;
106                 U.first = fa[top[U.first]] ;
107             }
108
109             if(dep[U.first] < dep[V.first]) swap(U,
110                 V) ;
111
112             M = query(dfn[V.first], dfn[U.first]) ;
113
114             return (U.second.tot + V.second.tot +
115                 M.tot) - (U.second.left == M.right)
116                 - (V.second.left == M.left) ;
117         }
118     }tree ;
119
120     void init(){
121         memset(son, -1, sizeof(son)) ;
122     }
123 }
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999

```

```

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         int flg = 0;
42         for ( auto n : node->end ){
43             if(flg) cout << " ";
44             else flg = 1;
45             cout << n;
46         }
47     }
48
49     void clear(TrieNode* node) {
50         if (!node) return;
51         for (int i = 0; i < 26; i++) {
52             if (node->next[i]) {
53                 clear(node->next[i]);
54             }
55         }
56         delete node;
57     }
58
59     ~Trie(){
60         clear(root);
61     }

```

3.6 BIT ver1

```

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 ver2

```

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 ver3

```

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(){

```

```

36     memset(BIT1, 0, sizeof(BIT1));
37     memset(BIT2, 0, sizeof(BIT2));
38 }

```

4 Graph

4.1 cut vertex AND bridges

```

1 int dfn[Maxn] = {-1}, low[Maxn] = {-1},
   dfsCnt;
2
3 void dfs(int u, int fa){
4     dfn[u] = low[u] = ++dfsCnt;
5     int child = 0;
6
7     for ( auto v : g[u] ) if(v != fa){
8         if(dfn[v] == -1){
9             child++;
10            dfs(v, u);
11            low[u] = min(low[u], low[v]);
12
13            if(low[v] >= dfn[u]){
14                // this edge is a bridge
15            }
16
17            if(u != fa && low[v] >= dfn[u]){
18                // this node v is a articulation point
19            }
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 }
46
47 void init(){
48     memset(dfn, -1, sizeof(dfn)) ;
49     memset(low, -1, sizeof(low)) ;
50 }
51
52 int main(){
53     init() ;
54     input() ;
55     for ( int i=1 ; i<=n ; i++ ) if(dfn[i]
56         == -1){
57         dfs(i, i) ;
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
36 void dfs2(int u, int id){
37     vis_bcc[u] = id ;
38     bcc[id].push_back(u) ;
39
40     for ( int i=head[u] ; i!=-1 ; i=e[i].next
41         ){
42         int v = e[i].v ;
43
44         if(vis_bcc[v] != -1 || bz[i]) continue ;
45         dfs2(v, id) ;
46     }
47 }
48 void init(){

```

```

45     memset(dfn, -1, sizeof(dfn)) ;
46     memset(head, -1, sizeof(head)) ;
47     memset(vis_bcc, -1, sizeof(vis_bcc)) ;
48 }
49
50 int main(){
51     init() ;
52     input() ;
53     for ( int i=1 ; i<=n ; i++ ) if(dfn[i]
54         == -1){
55         dfs1(i, 0) ;
56     }
57
58     for ( int i=1 ; i<=n ; i++ ) if(vis_bcc[i]
59         == -1){
60         bcc.push_back(vector<int>()) ;
61         dfs2(i, bcc_cnt++) ;
62     }

```

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    vector<Coordinate> nodes ;
15
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        cin >> n ;
28
29        for ( int i=0 ; i<n ; i++ ){
30            cin >> x >> y >> c ;
31            if(c == 'Y') nodes.push_back({x, y}) ;
32        }
33
34        void monotone(){
35            sort(nodes.begin(), nodes.end()) ;
36
37            int n = unique(nodes.begin(), nodes.end())
38                - nodes.begin() ;
39
40            vector<Coordinate> ch(n+1) ;
41
42            int m = 0 ;
43
44            for ( int i=0 ; i<n ; i++ ){
45                while(m > 1 && cross(ch[m-2], ch[m-1],
46                    nodes[i]) < 0) m-- ;
47                ch[m++] = nodes[i] ;
48            }
49
50            for ( int i=n-2, t=m ; i>=0 ; i-- ){
51                while(m > t && cross(ch[m-2], ch[m-1],
52                    nodes[i]) < 0) m-- ;
53                ch[m++] = nodes[i] ;
54            }

```

```

49     }
50
51     if(n > 1) m-- ;
52     cout << m << endl ;
53
54     for ( int i=0 ; i<m ; i++ ) cout <<
55         ch[i].x << " " << ch[i].y << endl ;

```

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                 if(d > 0){
49                     e[i].cap -= d ;
50                     e[i^1].cap += d ;
51                     rest += d ;
52                     if(rest == flow) break ;
53                 }
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
68 void AddEdge(int u, int v, int cap){
69     e.push_back({v, cap, head[u]});
70     head[u] = e.size() - 1;
71     e.push_back({u, 0, head[v]});
72     head[v] = e.size() - 1;
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; i=e[i].next){
44                    int v = e[i].v, cap = e[i].cap, w = e[i].cost;
45                    if(cap && dis[v] > d + w + pot[u] - pot[v]){
46                        dis[v] = d + w + pot[u] - pot[v];
47                        preE[v] = i;
48                        pq.push({dis[v], v});
49                    }
50                }
51            }
52        };
53
54        while(dijkstra()){
55            for (int v=1; v<=N; v++) if(dis[v] < INF){
56                pot[v] += dis[v];
57            }
58
59            int aug = INT_MAX;
60            for (int v=t; v!=s; v=e[preE[v]^1].v){
61                aug = min(aug, e[preE[v]].cap);
62            }
63
64            for (int v=t; v!=s; v=e[preE[v]^1].v){
65                e[preE[v]].cap -= aug;
66                e[preE[v]^1].cap += aug;
67                cost += aug * e[preE[v]].cost;
68            }
69        }
70
71        return cost;
72    }
73 };

```

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 = Next[check];
13        else Next[cur++] = 0;
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 ACAutomation{
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    ACAutomation(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] = new_node();
33            u = tree[u][v];
34        }
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
50        while(!q.empty()){
51            int u = q.front(); q.pop();
52            order.push_back(u);
53
54            for (int i=0; i<alpha; i++){
55                if(tree[u][i] == 0) tree[u][i] = new_node();
56                else{
57                    fail[tree[u][i]] = tree[fail[u]][i];
58                    q.push(tree[u][i]);
59                }
60            }
61        }
62
63        vector<int> count_per_pattern(const string &text) const {
64            int u = 0;
65            vector<int> vis(tree.size(), 0);
66
67            for (char ch : text){
68                u = tree[u][ch - base];
69                vis[u]++;
70            }
71        }
72    }

```

```

71 for ( int i=order.size()-1 ; i>=1 ; i--
    ){
72     int x = order[i] ;
73     vis[fail[x]] += vis[x] ;
74 }
75 vector<int> ans(end.size(), 0) ;
76 for ( int id=0 ; id<end.size() ; id++){
77     ans[id] = vis[end[id]] ;
78 }
79
80 return ans ;
81 }
82 }
83 };

```

6 Techniques

6.1 二分搜

```

1 // xxxxxxxooo 找最小解
2 bool binary_search(){
3     while(1 < 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(1 < 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     if(u == v) return u ;
34
35 for ( int i=MaxLog ; i>=0 ; i-- )
36     if(up[u][i] != up[v][i]){
37         u = up[u][i] ;
38         v = up[v][i] ;
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
12 int main(){
13     int cas, n ;
14
15     scanf("%d", &cas) ;
16     while(cas--){
17         scanf("%d", &n) ;
18
19         long long s, v = 0 ;
20
21         for(int i = 0 ; i < n ; i++){
22             scanf("%lld", &s) ;
23             v ^= SG(s) ; //XOR
24         }
25
26         if(v) printf("YES\n") ;
27         else printf("NO\n") ;
28     }
29 }
30
31 int SG[30] ;
32 int vis[Maxn], stone[Maxn] ;
33
34 void build(){
35     SG[0] = 0 ;
36     memset(vis, 0, sizeof(vis)) ;

```

```

37 for ( int i=1 ; i<30 ; i++ ){
38     int cur = 0 ;
39     for ( int j=0 ; j<i ; j++ ) for ( int
40         k=0 ; k<=j ; k++ ){
41         vis[SG[j] ^ SG[k]] = i ;
42     }
43     while(vis[cur] == i) cur++ ;
44     SG[i] = cur ;
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
17 int main(){
18     while(~scanf("%d%d", &n, &m)){
19         if(m > n) swap(n, m) ;
20         memset(dp, 0, sizeof(dp)) ;
21         cur = 0 ;
22         dp[cur][(1 << m) - 1] = 1 ;
23         for ( int i=0 ; i<n ; i++ ) for ( int
24             j=0 ; j<m ; j++ ){
25             cur ^= 1 ;
26             memset(dp[cur], 0, sizeof(dp[cur])) ;
27
28             for ( int k=0 ; k<(1 << m) ; k++ ){
29                 update(k, k << 1) ; // not put
30                 if(i && !(k & (1 << (m - 1))))
31                     update(k, (k << 1) | (1 << m) |
32                         1) ; // put up
33                 if(j && !(k & 1)) update(k, (k << 1)
34                     | 3) ; // put left
35             }
36         }
37         printf("%lld\n", dp[cur][(1 << m) - 1]) ;
38     }
39     return 0 ;
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
    lim){
10     if(pos == -1){
11         if(sum == 0 && dsum == 0) return 1 ;
12         return 0 ;
13     }
14
15     int &d = dp[pos][sum][dsum][lim] ;
16     if(d != -1) return d ;
17
18     int up = lim ? dig[pos] : 9 ;
19     int res = 0 ;
20     for ( int i=0 ; i<=up ; i++){
21         res += solve(pos-1, (sum * 10 + i) %
            K, (dsum + i) % K, lim && i==up)
            ;
22     }
23
24     return d = res ;
25 }
26
27 int count(int n){
28     memset(dp, -1, sizeof(dp)) ;
29     dig.clear() ;
30
31     while(n > 0){
32         dig.push_back(n % 10) ;
33         n /= 10 ;
34     }
35
36     return solve(dig.size() - 1, 0, 0, 1) ;
37 }
38
39 int main(){
40     int T ;
41     scanf("%d", &T) ;
42
43     int a, b ;
44     while(T--){
45         scanf("%d%d", &a, &b, &K) ;
46         if(K > 90) printf("0\n") ;
47         else printf("%d\n", count(b) -
            count(a-1)) ;
48     }
49
50     return 0 ;
51 }

```

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 (
            int j=1 ; j<i && j<=cnt[v] ; j++ ){
35             dp[u][i][1] = min(dp[u][i][1],
                dp[u][i-j][1] + dp[v][j][1] + 2 *
                w) ;
36             dp[u][i][0] = min(dp[u][i][0],
                dp[u][i-j][1] + dp[v][j][0] + w) ;
37             dp[u][i][0] = min(dp[u][i][0],
                dp[u][i-j][0] + dp[v][j][1] + 2 *
                w) ;
38         }
39     }
40
41 int main(){
42     int t = 0 ;
43
44     while(~scanf("%d", &n) && n){
45         init() ;
46         for ( int i=0 ; i<n-1 ; i++){
47             int u, v, w ;
48             scanf("%d%d%d", &v, &u, &w) ;
49             edge[u].push_back({v, w}) ;
50         }
51
52         DFS(0) ;
53         printf("Case %d:\n", ++t) ;
54
55         int q, e ;
56         scanf("%d", &q) ;
57
58         while(q--){
59             scanf("%d", &e) ;
60
61             for ( int i=n ; i>=1 ; i-- )
62                 if(dp[0][i][0] <= e){
63                     printf("%d\n", i) ;
64                     break ;
65                 }
66         }
67
68         return 0 ;
69 }

```