

## Contents

|     |                                  |
|-----|----------------------------------|
| 1   | Basic                            |
| 1.1 | PyMath . . . . .                 |
| 2   | Math                             |
| 2.1 | formula . . . . .                |
| 2.2 | extended gcd . . . . .           |
| 3   | Tree                             |
| 3.1 | SegmentTree . . . . .            |
| 3.2 | HLD . . . . .                    |
| 3.3 | PST . . . . .                    |
| 4   | Graph                            |
| 4.1 | cut vertex AND bridges . . . . . |
| 4.2 | SCC - Tarjan . . . . .           |
| 4.3 | BCC - Tarjan . . . . .           |
| 4.4 | Convex . . . . .                 |
| 5   | String                           |
| 5.1 | KMP . . . . .                    |
| 5.2 | Trie . . . . .                   |
| 5.3 | ACAM . . . . .                   |
| 6   | Algorithm                        |
| 6.1 | LCA . . . . .                    |
| 6.2 | MST . . . . .                    |
| 6.3 | SG . . . . .                     |
| 6.4 | Max Flow . . . . .               |
| 6.5 | min cut max flow . . . . .       |
| 7   | DP                               |
| 7.1 | 輪廓線 DP . . . . .                 |
| 7.2 | 數位 DP . . . . .                  |
| 7.3 | 樹 DP . . . . .                   |

## 1 Basic

### 1.1 PyMath

```

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

```

## 2 Math

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

#### 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

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

### 2.2 extended gcd

給定  $a, b, c$ , 求  $ax + by = c$  的解

```

1  ll extgcd(ll a, ll b, llc, 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 =
8      x ;
9    x = y ;
10   y = tmp - (a/b)*y ;
11   return d ;
}

```

## 3 Tree

### 3.1 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   void AddTag(int id, LazyTag &tag){
25     if(tag.type == 0){
26       seg[id].sum += tag.val *
27         seg[id].sz ;
28       seg[id].tag.val += tag.val ;
29     }
30     else{
31       seg[id].sum = tag.val *
32         seg[id].sz ;
33       seg[id].tag = {1, tag.val} ;
34     }
35   }
36   void push(int id){
37     AddTag(lc, seg[id].tag) ;
38     AddTag(rc, seg[id].tag) ;
39     seg[id].tag = {0, 0} ;
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,
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   }
72
73   void print(int id, int L=1, int R=n){
74     if(L == R){
75       cout << arr[L] << endl ;
76     }
77     else{
78       int M = (L + R) >> 1 ;
79       print(lc, L, M) ;
80       print(rc, M+1, R) ;
81     }
82   }
83
84   ll query(int l, int r, int id=1, int L=1, int R=n){
85     if(l > R || r < L) return 0 ;
86     if(l <= L && r >= R) return seg[id].sum ;
87
88     int M = (L + R) >> 1 ;
89     return query(l, r, lc, L, M) +
90       query(l, r, rc, M+1, R) ;
91   }
92
93   void print(int id, int L=1, int R=n){
94     if(L == R){
95       cout << arr[L] << endl ;
96     }
97     else{
98       int M = (L + R) >> 1 ;
99       print(lc, L, M) ;
100      print(rc, M+1, R) ;
101    }
102  }
103
104  void print(int id, int L=1, int R=n){
105    if(L == R){
106      cout << arr[L] << endl ;
107    }
108    else{
109      int M = (L + R) >> 1 ;
110      print(lc, L, M) ;
111      print(rc, M+1, R) ;
112    }
113  }
114
115  void print(int id, int L=1, int R=n){
116    if(L == R){
117      cout << arr[L] << endl ;
118    }
119    else{
120      int M = (L + R) >> 1 ;
121      print(lc, L, M) ;
122      print(rc, M+1, R) ;
123    }
124  }
125
126  void print(int id, int L=1, int R=n){
127    if(L == R){
128      cout << arr[L] << endl ;
129    }
130    else{
131      int M = (L + R) >> 1 ;
132      print(lc, L, M) ;
133      print(rc, M+1, R) ;
134    }
135  }
136
137  void print(int id, int L=1, int R=n){
138    if(L == R){
139      cout << arr[L] << endl ;
140    }
141    else{
142      int M = (L + R) >> 1 ;
143      print(lc, L, M) ;
144      print(rc, M+1, R) ;
145    }
146  }
147
148  void print(int id, int L=1, int R=n){
149    if(L == R){
150      cout << arr[L] << endl ;
151    }
152    else{
153      int M = (L + R) >> 1 ;
154      print(lc, L, M) ;
155      print(rc, M+1, R) ;
156    }
157  }
158
159  void print(int id, int L=1, int R=n){
160    if(L == R){
161      cout << arr[L] << endl ;
162    }
163    else{
164      int M = (L + R) >> 1 ;
165      print(lc, L, M) ;
166      print(rc, M+1, R) ;
167    }
168  }
169
170  void print(int id, int L=1, int R=n){
171    if(L == R){
172      cout << arr[L] << endl ;
173    }
174    else{
175      int M = (L + R) >> 1 ;
176      print(lc, L, M) ;
177      print(rc, M+1, R) ;
178    }
179  }
180
181  void print(int id, int L=1, int R=n){
182    if(L == R){
183      cout << arr[L] << endl ;
184    }
185    else{
186      int M = (L + R) >> 1 ;
187      print(lc, L, M) ;
188      print(rc, M+1, R) ;
189    }
190  }
191
192  void print(int id, int L=1, int R=n){
193    if(L == R){
194      cout << arr[L] << endl ;
195    }
196    else{
197      int M = (L + R) >> 1 ;
198      print(lc, L, M) ;
199      print(rc, M+1, R) ;
200    }
201  }
202
203  void print(int id, int L=1, int R=n){
204    if(L == R){
205      cout << arr[L] << endl ;
206    }
207    else{
208      int M = (L + R) >> 1 ;
209      print(lc, L, M) ;
210      print(rc, M+1, R) ;
211    }
212  }
213
214  void print(int id, int L=1, int R=n){
215    if(L == R){
216      cout << arr[L] << endl ;
217    }
218    else{
219      int M = (L + R) >> 1 ;
220      print(lc, L, M) ;
221      print(rc, M+1, R) ;
222    }
223  }
224
225  void print(int id, int L=1, int R=n){
226    if(L == R){
227      cout << arr[L] << endl ;
228    }
229    else{
230      int M = (L + R) >> 1 ;
231      print(lc, L, M) ;
232      print(rc, M+1, R) ;
233    }
234  }
235
236  void print(int id, int L=1, int R=n){
237    if(L == R){
238      cout << arr[L] << endl ;
239    }
240    else{
241      int M = (L + R) >> 1 ;
242      print(lc, L, M) ;
243      print(rc, M+1, R) ;
244    }
245  }
246
247  void print(int id, int L=1, int R=n){
248    if(L == R){
249      cout << arr[L] << endl ;
250    }
251    else{
252      int M = (L + R) >> 1 ;
253      print(lc, L, M) ;
254      print(rc, M+1, R) ;
255    }
256  }
257
258  void print(int id, int L=1, int R=n){
259    if(L == R){
260      cout << arr[L] << endl ;
261    }
262    else{
263      int M = (L + R) >> 1 ;
264      print(lc, L, M) ;
265      print(rc, M+1, R) ;
266    }
267  }
268
269  void print(int id, int L=1, int R=n){
270    if(L == R){
271      cout << arr[L] << endl ;
272    }
273    else{
274      int M = (L + R) >> 1 ;
275      print(lc, L, M) ;
276      print(rc, M+1, R) ;
277    }
278  }
279
280  void print(int id, int L=1, int R=n){
281    if(L == R){
282      cout << arr[L] << endl ;
283    }
284    else{
285      int M = (L + R) >> 1 ;
286      print(lc, L, M) ;
287      print(rc, M+1, R) ;
288    }
289  }
290
291  void print(int id, int L=1, int R=n){
292    if(L == R){
293      cout << arr[L] << endl ;
294    }
295    else{
296      int M = (L + R) >> 1 ;
297      print(lc, L, M) ;
298      print(rc, M+1, R) ;
299    }
300  }
301
302  void print(int id, int L=1, int R=n){
303    if(L == R){
304      cout << arr[L] << endl ;
305    }
306    else{
307      int M = (L + R) >> 1 ;
308      print(lc, L, M) ;
309      print(rc, M+1, R) ;
310    }
311  }
312
313  void print(int id, int L=1, int R=n){
314    if(L == R){
315      cout << arr[L] << endl ;
316    }
317    else{
318      int M = (L + R) >> 1 ;
319      print(lc, L, M) ;
320      print(rc, M+1, R) ;
321    }
322  }
323
324  void print(int id, int L=1, int R=n){
325    if(L == R){
326      cout << arr[L] << endl ;
327    }
328    else{
329      int M = (L + R) >> 1 ;
330      print(lc, L, M) ;
331      print(rc, M+1, R) ;
332    }
333  }
334
335  void print(int id, int L=1, int R=n){
336    if(L == R){
337      cout << arr[L] << endl ;
338    }
339    else{
340      int M = (L + R) >> 1 ;
341      print(lc, L, M) ;
342      print(rc, M+1, R) ;
343    }
344  }
345
346  void print(int id, int L=1, int R=n){
347    if(L == R){
348      cout << arr[L] << endl ;
349    }
350    else{
351      int M = (L + R) >> 1 ;
352      print(lc, L, M) ;
353      print(rc, M+1, R) ;
354    }
355  }
356
357  void print(int id, int L=1, int R=n){
358    if(L == R){
359      cout << arr[L] << endl ;
360    }
361    else{
362      int M = (L + R) >> 1 ;
363      print(lc, L, M) ;
364      print(rc, M+1, R) ;
365    }
366  }
367
368  void print(int id, int L=1, int R=n){
369    if(L == R){
370      cout << arr[L] << endl ;
371    }
372    else{
373      int M = (L + R) >> 1 ;
374      print(lc, L, M) ;
375      print(rc, M+1, R) ;
376    }
377  }
378
379  void print(int id, int L=1, int R=n){
380    if(L == R){
381      cout << arr[L] << endl ;
382    }
383    else{
384      int M = (L + R) >> 1 ;
385      print(lc, L, M) ;
386      print(rc, M+1, R) ;
387    }
388  }
389
390  void print(int id, int L=1, int R=n){
391    if(L == R){
392      cout << arr[L] << endl ;
393    }
394    else{
395      int M = (L + R) >> 1 ;
396      print(lc, L, M) ;
397      print(rc, M+1, R) ;
398    }
399  }
400
401  void print(int id, int L=1, int R=n){
402    if(L == R){
403      cout << arr[L] << endl ;
404    }
405    else{
406      int M = (L + R) >> 1 ;
407      print(lc, L, M) ;
408      print(rc, M+1, R) ;
409    }
410  }
411
412  void print(int id, int L=1, int R=n){
413    if(L == R){
414      cout << arr[L] << endl ;
415    }
416    else{
417      int M = (L + R) >> 1 ;
418      print(lc, L, M) ;
419      print(rc, M+1, R) ;
420    }
421  }
422
423  void print(int id, int L=1, int R=n){
424    if(L == R){
425      cout << arr[L] << endl ;
426    }
427    else{
428      int M = (L + R) >> 1 ;
429      print(lc, L, M) ;
430      print(rc, M+1, R) ;
431    }
432  }
433
434  void print(int id, int L=1, int R=n){
435    if(L == R){
436      cout << arr[L] << endl ;
437    }
438    else{
439      int M = (L + R) >> 1 ;
440      print(lc, L, M) ;
441      print(rc, M+1, R) ;
442    }
443  }
444
445  void print(int id, int L=1, int R=n){
446    if(L == R){
447      cout << arr[L] << endl ;
448    }
449    else{
450      int M = (L + R) >> 1 ;
451      print(lc, L, M) ;
452      print(rc, M+1, R) ;
453    }
454  }
455
456  void print(int id, int L=1, int R=n){
457    if(L == R){
458      cout << arr[L] << endl ;
459    }
460    else{
461      int M = (L + R) >> 1 ;
462      print(lc, L, M) ;
463      print(rc, M+1, R) ;
464    }
465  }
466
467  void print(int id, int L=1, int R=n){
468    if(L == R){
469      cout << arr[L] << endl ;
470    }
471    else{
472      int M = (L + R) >> 1 ;
473      print(lc, L, M) ;
474      print(rc, M+1, R) ;
475    }
476  }
477
478  void print(int id, int L=1, int R=n){
479    if(L == R){
480      cout << arr[L] << endl ;
481    }
482    else{
483      int M = (L + R) >> 1 ;
484      print(lc, L, M) ;
485      print(rc, M+1, R) ;
486    }
487  }
488
489  void print(int id, int L=1, int R=n){
490    if(L == R){
491      cout << arr[L] << endl ;
492    }
493    else{
494      int M = (L + R) >> 1 ;
495      print(lc, L, M) ;
496      print(rc, M+1, R) ;
497    }
498  }
499
500  void print(int id, int L=1, int R=n){
501    if(L == R){
502      cout << arr[L] << endl ;
503    }
504    else{
505      int M = (L + R) >> 1 ;
506      print(lc, L, M) ;
507      print(rc, M+1, R) ;
508    }
509  }
510
511  void print(int id, int L=1, int R=n){
512    if(L == R){
513      cout << arr[L] << endl ;
514    }
515    else{
516      int M = (L + R) >> 1 ;
517      print(lc, L, M) ;
518      print(rc, M+1, R) ;
519    }
520  }
521
522  void print(int id, int L=1, int R=n){
523    if(L == R){
524      cout << arr[L] << endl ;
525    }
526    else{
527      int M = (L + R) >> 1 ;
528      print(lc, L, M) ;
529      print(rc, M+1, R) ;
530    }
531  }
532
533  void print(int id, int L=1, int R=n){
534    if(L == R){
535      cout << arr[L] << endl ;
536    }
537    else{
538      int M = (L + R) >> 1 ;
539      print(lc, L, M) ;
540      print(rc, M+1, R) ;
541    }
542  }
543
544  void print(int id, int L=1, int R=n){
545    if(L == R){
546      cout << arr[L] << endl ;
547    }
548    else{
549      int M = (L + R) >> 1 ;
550      print(lc, L, M) ;
551      print(rc, M+1, R) ;
552    }
553  }
554
555  void print(int id, int L=1, int R=n){
556    if(L == R){
557      cout << arr[L] << endl ;
558    }
559    else{
560      int M = (L + R) >> 1 ;
561      print(lc, L, M) ;
562      print(rc, M+1, R) ;
563    }
564  }
565
566  void print(int id, int L=1, int R=n){
567    if(L == R){
568      cout << arr[L] << endl ;
569    }
570    else{
571      int M = (L + R) >> 1 ;
572      print(lc, L, M) ;
573      print(rc, M+1, R) ;
574    }
575  }
576
577  void print(int id, int L=1, int R=n){
578    if(L == R){
579      cout << arr[L] << endl ;
580    }
581    else{
582      int M = (L + R) >> 1 ;
583      print(lc, L, M) ;
584      print(rc, M+1, R) ;
585    }
586  }
587
588  void print(int id, int L=1, int R=n){
589    if(L == R){
590      cout << arr[L] << endl ;
591    }
592    else{
593      int M = (L + R) >> 1 ;
594      print(lc, L, M) ;
595      print(rc, M+1, R) ;
596    }
597  }
598
599  void print(int id, int L=1, int R=n){
600    if(L == R){
601      cout << arr[L] << endl ;
602    }
603    else{
604      int M = (L + R) >> 1 ;
605      print(lc, L, M) ;
606      print(rc, M+1, R) ;
607    }
608  }
609
610  void print(int id, int L=1, int R=n){
611    if(L == R){
612      cout << arr[L] << endl ;
613    }
614    else{
615      int M = (L + R) >> 1 ;
616      print(lc, L, M) ;
617      print(rc, M+1, R) ;
618    }
619  }
620
621  void print(int id, int L=1, int R=n){
622    if(L == R){
623      cout << arr[L] << endl ;
624    }
625    else{
626      int M = (L + R) >> 1 ;
627      print(lc, L, M) ;
628      print(rc, M+1, R) ;
629    }
630  }
631
632  void print(int id, int L=1, int R=n){
633    if(L == R){
634      cout << arr[L] << endl ;
635    }
636    else{
637      int M = (L + R) >> 1 ;
638      print(lc, L, M) ;
639      print(rc, M+1, R) ;
640    }
641  }
642
643  void print(int id, int L=1, int R=n){
644    if(L == R){
645      cout << arr[L] << endl ;
646    }
647    else{
648      int M = (L + R) >> 1 ;
649      print(lc, L, M) ;
650      print(rc, M+1, R) ;
651    }
652  }
653
654  void print(int id, int L=1, int R=n){
655    if(L == R){
656      cout << arr[L] << endl ;
657    }
658    else{
659      int M = (L + R) >> 1 ;
660      print(lc, L, M) ;
661      print(rc, M+1, R) ;
662    }
663  }
664
665  void print(int id, int L=1, int R=n){
666    if(L == R){
667      cout << arr[L] << endl ;
668    }
669    else{
670      int M = (L + R) >> 1 ;
671      print(lc, L, M) ;
672      print(rc, M+1, R) ;
673    }
674  }
675
676  void print(int id, int L=1, int R=n){
677    if(L == R){
678      cout << arr[L] << endl ;
679    }
680    else{
681      int M = (L + R) >> 1 ;
682      print(lc, L, M) ;
683      print(rc, M+1, R) ;
684    }
685  }
686
687  void print(int id, int L=1, int R=n){
688    if(L == R){
689      cout << arr[L] << endl ;
690    }
691    else{
692      int M = (L + R) >> 1 ;
693      print(lc, L, M) ;
694      print(rc, M+1, R) ;
695    }
696  }
697
698  void print(int id, int L=1, int R=n){
699    if(L == R){
700      cout << arr[L] << endl ;
701    }
702    else{
703      int M = (L + R) >> 1 ;
704      print(lc, L, M) ;
705      print(rc, M+1, R) ;
706    }
707  }
708
709  void print(int id, int L=1, int R=n){
710    if(L == R){
711      cout << arr[L] << endl ;
712    }
713    else{
714      int M = (L + R) >> 1 ;
715      print(lc, L, M) ;
716      print(rc, M+1, R) ;
717    }
718  }
719
720  void print(int id, int L=1, int R=n){
721    if(L == R){
722      cout << arr[L] << endl ;
723    }
724    else{
725      int M = (L + R) >> 1 ;
726      print(lc, L, M) ;
727      print(rc, M+1, R) ;
728    }
729  }
730
731  void print(int id, int L=1, int R=n){
732    if(L == R){
733      cout << arr[L] << endl ;
734    }
735    else{
736      int M = (L + R) >> 1 ;
737      print(lc, L, M) ;
738      print(rc, M+1, R) ;
739    }
740  }
741
742  void print(int id, int L=1, int R=n){
743    if(L == R){
744      cout << arr[L] << endl ;
745    }
746    else{
747      int M = (L + R) >> 1 ;
748      print(lc, L, M) ;
749      print(rc, M+1, R) ;
750    }
751  }
752
753  void print(int id, int L=1, int R=n){
754    if(L == R){
755      cout << arr[L] << endl ;
756    }
757    else{
758      int M = (L + R) >> 1 ;
759      print(lc, L, M) ;
760      print(rc, M+1, R) ;
761    }
762  }
763
764  void print(int id, int L=1, int R=n){
765    if(L == R){
766      cout << arr[L] << endl ;
767    }
768    else{
769      int M = (L + R) >> 1 ;
770      print(lc, L, M) ;
771      print(rc, M+1, R) ;
772    }
773  }
774
775  void print(int id, int L=1, int R=n){
776    if(L == R){
777      cout << arr[L] << endl ;
778    }
779    else{
780      int M = (L + R) >> 1 ;
781      print(lc, L, M) ;
782      print(rc, M+1, R) ;
783    }
784  }
785
786  void print(int id, int L=1, int R=n){
787    if(L == R){
788      cout << arr[L] << endl ;
789    }
790    else{
791      int M = (L + R) >> 1 ;
792      print(lc, L, M) ;
793      print(rc, M+1, R) ;
794    }
795  }
796
797  void print(int id, int L=1, int R=n){
798    if(L == R){
799      cout << arr[L] << endl ;
800    }
801    else{
802      int M = (L + R) >> 1 ;
803      print(lc, L, M) ;
804      print(rc, M+1, R) ;
805    }
806  }
807
808  void print(int id, int L=1, int R=n){
809    if(L == R){
810      cout << arr[L] << endl ;
811    }
812    else{
813      int M = (L + R) >> 1 ;
814      print(lc, L, M) ;
815      print(rc, M+1, R) ;
816    }
817  }
818
819  void print(int id, int L=1, int R=n){
820    if(L == R){
821      cout << arr[L] << endl ;
822    }
823    else{
824      int M = (L + R) >> 1 ;
825      print(lc, L, M) ;
826      print(rc, M+1, R) ;
827    }
828  }
829
830  void print(int id, int L=1, int R=n){
831    if(L == R){
832      cout << arr[L] << endl ;
833    }
834    else{
835      int M = (L + R) >> 1 ;
836      print(lc, L, M) ;
837      print(rc, M+1, R) ;
838    }
839  }
840
841  void print(int id, int L=1, int R=n){
842    if(L == R){
843      cout << arr[L] << endl ;
844    }
845    else{
846      int M = (L + R) >> 1 ;
847      print(lc, L, M) ;
848      print(rc, M+1, R) ;
849    }
850  }
851
852  void print(int id, int L=1, int R=n){
853    if(L == R){
854      cout << arr[L] << endl ;
855    }
856    else{
857      int M = (L + R) >> 1 ;
858      print(lc, L, M) ;
859      print
```

```

    else if(l > M) modify(l, r, tag, M+1,
                           R, rc) ;
    else{
        modify(l, r, tag, L, M, lc) ;
        modify(l, r, tag, M+1, R, rc) ;
    }
    pull(id) ;
}

ll query(int l, int r, int L=1, int R=n,
         int id=1){
    if(l <= L && R <= r) return
        seg[id].sum ;

    push(id) ;
    int M = (L + R) >> 1 ;
    if(r <= M) return query(l, r, L, M,
                            lc) ;
    else if(l > M) return query(l, r,
                                 M+1, R, rc) ;
    else return query(l, r, L, M, lc) +
        query(l, r, M+1, R, rc) ;
}
}tree ;

```

### 3.2 HLD

```

/* HLD */
int fa[Maxn], top[Maxn], son[Maxn],
sz[Maxn], dep[Maxn] = {0}, dfn[Maxn]
rnk[Maxn], dfscnt = 0 ;

void dfs1(int u, int from){
    fa[u] = from ;
    dep[u] = dep[from] + 1 ;
    sz[u] = 1 ;

    for ( auto v : g[u] ) if(v != from){
        dfs1(v, u) ;
        sz[u] += sz[v] ;
        if(son[u] == -1 || sz[v] > sz[son[u]])
            son[u] = v ;
    }
}

void dfs2(int u, int t){
    top[u] = t ;
    dfn[u] = ++dfscnt ;
    rnk[dfscnt] = u ;

    if(son[u] == -1) return ;

    dfs2(son[u], t) ;

    for ( auto v : g[u] ) if(v != fa[u] &&
        != son[u]){
        dfs2(v, v) ;
    }
}

/* Segment Tree */
#define lc (id << 1)
#define rc ((id << 1) | 1)

struct ColorSeg{
    int left, right, tot ;

    ColorSeg operator+(const ColorSeg &o)
        const {
        if(tot == 0) return o ;
        if(o.tot == 0) return *this ;

        ColorSeg tmp ;
        tmp.left = left ;
        tmp.right = o.right ;
        tmp.tot = tot + o.tot - (right ==
            o.left) ;
    }
}

```

```

15     return tmp ;
16 }
17 }
18 } ;
19
20 struct Node{
21     ColorSeg color ;
22     int tag ;
23 }seg[Maxn << 2] ;
24
25 class SegmentTree{
26 private:
27     void pull(int id){
28         // normal pull
29     }
30
31     void AddTag(int id, int tag){
32         // normal AddTag
33     }
34
35     void push(int id){
36         // normal push
37     }
38
39     void modify(int l, int r, int tag, int
40                 L=1, int R=n, int id=1){
41         // normal modify
42     }
43
44     ColorSeg query(int l, int r, int L=1, int
45                    R=n, int id=1){
46         // normal query
47     }
48 public:
49     void build(int L=1, int R=n, int id=1){
50         // normal build
51     }
52
53     // update val from u to v (simple path)
54     void update(int u, int v, int val){
55         while(top[u] != top[v]){
56             if(dep[top[u]] < dep[top[v]]) swap(u,
57                                         v) ;
58             modify(dfn[top[u]], dfn[u], val) ;
59             u = fa[top[u]] ;
60         }
61
62         if(dep[u] < dep[v]) swap(u, v) ;
63         modify(dfn[v], dfn[u], val) ;
64     }
65
66     // get sum from u to v (simple path)
67     int get(int u, int v){
68         pair<int, ColorSeg> U, V ;
69         ColorSeg M ;
70         U = {u, {0, 0, 0}} ;
71         V = {v, {0, 0, 0}} ;
72
73         while(top[U.first] != top[V.first]){
74             if(dep[top[U.first]] <
75                dep[top[V.first]]) swap(U, V) ;
76             U.second = query(dfn[top[U.first]],
77                               dfn[U.first]) + U.second ;
78             U.first = fa[top[U.first]] ;
79         }
80
81         if(dep[U.first] < dep[V.first]) swap(U,
82                                         V) ;
83
84         M = query(dfn[V.first], dfn[U.first]) ;
85
86         return (U.second.tot + V.second.tot +
87                 M.tot) - (U.second.left == M.right
88                             - (V.second.left == M.left) ;
89     }
90 }
91 tree ;
92
93
94 void init(){}

```

```

115     memset(son, -1, sizeof(son)) ;
116 }

3.3 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 ;
49
50            if(idx <= M) lc = self(self,
51                            seg[cur].left, L, M) ;
52            else rc = self(self, seg[cur].right,
53                            M+1, R) ;
54
55            seg[cur_cnt] = {seg[lc].sum +
56                            seg[rc].sum, lc, rc} ;
57
58            return cur_cnt ;
59        };
56
57        int root = upd(upd, version[ver], 1, n) ;
58        version.push_back(root) ;
59    }
60
61    int query(int verL, int verR, int k){
62        auto qry = [&](auto &&self, const int
63                        cur_old, const int cur_new, int L,
64                        int R){
65            if(L == R) return L ;
66
67            int old_l = seg[cur_old].left, old_r =
68                        seg[cur_old].right ;

```

```

65     int new_l = seg[cur_new].left, new_r = 16
66         seg[cur_new].right ;
67
68     int dl = seg[new_l].sum -
69         seg[old_l].sum ;
70     int dr = seg[new_r].sum -
71         seg[old_r].sum ;
72
73     int M = (L + R) >> 1 ;
74
75     if(dl >= k) return self(self, old_l,
76         new_l, L, M) ;
77     k -= dl ;
78     return self(self, old_r, new_r, M+1,
79         R) ;
80 }
81
82     int idx = qry(qry, version[verL-1],
83         version[verR], 1, n) ;
84     return idx ;
85 }
86
87 }
```

## 4 Graph

### 4.1 cut vertex AND bridges

```

1 int dfn[Maxn] = {-1}, low[Maxn] = {-1},
2     dfscnt ;
3
4 void dfs(int u, int fa){
5     dfn[u] = low[u] = ++dfscnt ;
6     int child = 0 ;
7
8     for ( auto v : g[u] ) if(v != fa){
9         if(dfn[v] == -1){
10             child++ ;
11             dfs(v, u) ;
12             low[u] = min(low[u], low[v]) ;
13
14             if(low[v] >= dfn[u]){
15                 // this edge is a bridge
16             }
17
18             if(u != fa && low[v] >= dfn[u]){
19                 // this node v is a articulation point
20             }
21             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],
3     dfscnt = 0, cnt_scc = 0 ;
4 stack<int> st ;
5 bitset<Maxn> inSt, vis ;
6
7 void dfs(int u, int from){
8     dfn[u] = low[u] = ++dfscnt ;
9     st.push(u) ;
10    inSt[u] = 1 ;
11
12    for ( auto v : g[u] ){
13        if(!inSt[v] && dfn[v] != -1) continue ;
14        if(dfn[v] == -1) dfs(v, u) ;
15        low[u] = min(low[u], low[v]) ;
16    }
17 }
```

```

16     if(dfn[u] == low[u]){
17         cnt_scc++ ;
18         int x ;
19         do{
20             x = st.top() ;
21             st.pop() ;
22
23             inSt[x] = 0 ;
24             sccId[x] = cnt_scc ;
25             scc[cnt_scc].push_back(x) ;
26         }
27         while(x != u) ;
28     }
29 }
30
31 void dfs2(int u, int id){
32     vis_bcc[u] = id ;
33     bcc[id].push_back(u) ;
34
35     for ( int i=head[u] ; i!=-1 ; i=e[i].next )
36     {
37         int v = e[i].v ;
38
39         if(vis_bcc[v] != -1 || bz[i]) continue ;
40         dfs2(v, id) ;
41     }
42 }
43
44 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     }
63 }
```

### 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 }
```

```

33 void monotone(){
34     sort(nodes.begin(), nodes.end()) ;
35
36     int n = unique(nodes.begin(), nodes.end())
37         - nodes.begin() ;
38
39     vector<Coordinate> ch(n+1) ;
40
41     int m = 0 ;
42
43     for ( int i=0 ; i<n ; i++ ){
44         while(m > 1 && cross(ch[m-2], ch[m-1],
45             nodes[i]) < 0) m-- ;
46         ch[m++] = nodes[i] ;
47     }
48
49     for ( int i=n-2, t=m ; i>=0 ; i-- ){
50         while(m > t && cross(ch[m-2], ch[m-1],
51             nodes[i]) < 0) m-- ;
52         ch[m++] = nodes[i] ;
53     }
54
55     if(n > 1) m-- ;
56     cout << m << endl ;
57
58     for ( int i=0 ; i<m ; i++ ) cout <<
59         ch[i].x << " " << ch[i].y << endl ;
56

```

## 5 String

### 5.1 KMP

```

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

```

### 5.2 Trie

```

1 class TrieNode{
2 public:
3     set<int> end ;
4     TrieNode *next[26] ;
5
6     TrieNode(){
7         for ( int i=0 ; i<26 ; i++ ) next[i]
8             = nullptr ;
9     }
10
11 class Trie{
12 private:
13     int cnt ;
14     TrieNode *root ;
15 public:
16     Trie() : cnt(0) {
17         root = new TrieNode() ;
18     }
19
20     void insert(string &str, int n){
21         TrieNode* node = root ;
22         for ( auto s : str ){
23             int path = s - 'a' ;
24
25             if(node->next[path] == nullptr)
26                 node->next[path] = new
27                     TrieNode() ;
28             node = node->next[path] ;
29
30             node->end.insert(n) ;
31         }
32
33         void search(string &str){
34             TrieNode* node = root ;
35             for ( auto s : str ){
36                 int path = s - 'a' ;
37                 if(node->next[path] == nullptr)
38                     return ;
39                 node = node->next[path] ;
40
41                 int flg = 0 ;
42                 for ( auto n : node->end ){
43                     if(flg) cout << " " ;
44                     else flg = 1 ;
45
46                     cout << n ;
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                         delete node ;
56                     }
57
58                     ~Trie(){
59                         clear(root) ;
60                     }
61                 };
62
63             }
64
65             ~Trie(){
66                 clear(root) ;
67             }
68
69             vector<int> count_per_pattern(const string
70                 &text) const {
71                 int u = 0 ;
72                 vector<int> vis(tree.size(), 0) ;
73
74                 for ( char ch : text ){
75                     u = tree[u][ch - base] ;
76                     vis[u]++;
77                 }
78
79                 for ( int i=order.size()-1 ; i>=1 ; i-- )
80                 int x = order[i] ;
81                 vis[fail[x]] += vis[x] ;
82
83                 vector<int> ans(end.size(), 0) ;
84                 for ( int id=0 ; id<end.size() ; id++ )
85                     ans[id] = vis[end[id]] ;
86
87             }
88
89             void add_pattern(const string &pattern){
90                 int u = 0 ;
91                 for ( auto ch : pattern ){
92                     int v = ch - base ;
93
94                     if(tree[u][v] == 0) tree[u][v] =
95                         new_node() ;
96                     u = tree[u][v] ;
97
98                 }
99
100                end.push_back(u) ;
101
102                void build(){
103                    queue<int> q ;
104                    order.clear() ;
105                    order.push_back(0) ;
106
107                    for ( int i=0 ; i<alpha ; i++ )
108                        if(tree[0][i] > 0){
109                            q.push(tree[0][i]) ;
110                        }
111
112                    while(!q.empty()){
113                        int u = q.front() ; q.pop() ;
114                        order.push_back(u) ;
115
116                        for ( int i=0 ; i<alpha ; i++ ){
117                            if(tree[u][i] == 0) tree[u][i] =
118                                tree[fail[u]][i] ;
119                            else{
120                                fail[tree[u][i]] = tree[fail[u]][i]
121                                    ;
122                                q.push(tree[u][i]) ;
123                            }
124                        }
125
126                    }
127
128                    vector<int> count_per_pattern(const string
129                        &text) const {
130                        int u = 0 ;
131                        vector<int> vis(tree.size(), 0) ;
132
133                        for ( char ch : text ){
134                            u = tree[u][ch - base] ;
135                            vis[u]++;
136                        }
137
138                        for ( int i=order.size()-1 ; i>=1 ; i-- )
139                        int x = order[i] ;
140                        vis[fail[x]] += vis[x] ;
141
142                        vector<int> ans(end.size(), 0) ;
143                        for ( int id=0 ; id<end.size() ; id++ )
144                            ans[id] = vis[end[id]] ;
145
146                    }
147
148                    void clear(){
149                        fail.assign(1, 0) ;
150                        order.clear() ;
151                        end.clear() ;
152                        tree.assign(1, vector<int>(alpha, 0)) ;
153
154                }
155
156                void add_pattern(const string &pattern){
157                    int u = 0 ;
158                    for ( auto ch : pattern ){
159                        int v = ch - base ;
160
161                        if(tree[u][v] == 0) tree[u][v] =
162                            new_node() ;
163                        u = tree[u][v] ;
164
165                    }
166
167                    end.push_back(u) ;
168
169                    void build(){
170                        queue<int> q ;
171                        order.clear() ;
172                        order.push_back(0) ;
173
174                        for ( int i=0 ; i<alpha ; i++ )
175                            if(tree[0][i] > 0){
176                                q.push(tree[0][i]) ;
177                            }
178
179                        while(!q.empty()){
180                            int u = q.front() ; q.pop() ;
181                            order.push_back(u) ;
182
183                            for ( int i=0 ; i<alpha ; i++ ){
184                                if(tree[u][i] == 0) tree[u][i] =
185                                    tree[fail[u]][i] ;
186                                else{
187                                    fail[tree[u][i]] = tree[fail[u]][i]
188                                        ;
189                                    q.push(tree[u][i]) ;
190                                }
191                            }
192
193                        }
194
195                        vector<int> count_per_pattern(const string
196                            &text) const {
197                            int u = 0 ;
198                            vector<int> vis(tree.size(), 0) ;
199
200                            for ( char ch : text ){
201                                u = tree[u][ch - base] ;
202                                vis[u]++;
203                            }
204
205                            for ( int i=order.size()-1 ; i>=1 ; i-- )
206                            int x = order[i] ;
207                            vis[fail[x]] += vis[x] ;
208
209                            vector<int> ans(end.size(), 0) ;
210                            for ( int id=0 ; id<end.size() ; id++ )
211                                ans[id] = vis[end[id]] ;
212
213                        }
214
215                        void clear(){
216                            fail.assign(1, 0) ;
217                            order.clear() ;
218                            end.clear() ;
219                            tree.assign(1, vector<int>(alpha, 0)) ;
220
221                    }
222
223                    void add_pattern(const string &pattern){
224                        int u = 0 ;
225                        for ( auto ch : pattern ){
226                            int v = ch - base ;
227
228                            if(tree[u][v] == 0) tree[u][v] =
229                                new_node() ;
230                            u = tree[u][v] ;
231
232                        }
233
234                        end.push_back(u) ;
235
236                        void build(){
237                            queue<int> q ;
238                            order.clear() ;
239                            order.push_back(0) ;
240
241                            for ( int i=0 ; i<alpha ; i++ )
242                                if(tree[0][i] > 0){
243                                    q.push(tree[0][i]) ;
244                                }
245
246                            while(!q.empty()){
247                                int u = q.front() ; q.pop() ;
248                                order.push_back(u) ;
249
250                                for ( int i=0 ; i<alpha ; i++ ){
251                                    if(tree[u][i] == 0) tree[u][i] =
252                                        tree[fail[u]][i] ;
253                                    else{
254                                        fail[tree[u][i]] = tree[fail[u]][i]
255                                            ;
256                                        q.push(tree[u][i]) ;
257                                    }
258                                }
259
260                            }
261
262                            vector<int> count_per_pattern(const string
263                                &text) const {
264                                int u = 0 ;
265                                vector<int> vis(tree.size(), 0) ;
266
267                                for ( char ch : text ){
268                                    u = tree[u][ch - base] ;
269                                    vis[u]++;
270                                }
271
272                                for ( int i=order.size()-1 ; i>=1 ; i-- )
273                                int x = order[i] ;
274                                vis[fail[x]] += vis[x] ;
275
276                                vector<int> ans(end.size(), 0) ;
277                                for ( int id=0 ; id<end.size() ; id++ )
278                                    ans[id] = vis[end[id]] ;
279
280                            }
281
282                            void clear(){
283                                fail.assign(1, 0) ;
284                                order.clear() ;
285                                end.clear() ;
286                                tree.assign(1, vector<int>(alpha, 0)) ;
287
288                            }
289
290                            void add_pattern(const string &pattern){
291                                int u = 0 ;
292                                for ( auto ch : pattern ){
293                                    int v = ch - base ;
294
295                                    if(tree[u][v] == 0) tree[u][v] =
296                                        new_node() ;
297                                    u = tree[u][v] ;
298
299                                }
300
301                                end.push_back(u) ;
302
303                                void build(){
304                                    queue<int> q ;
305                                    order.clear() ;
306                                    order.push_back(0) ;
307
308                                    for ( int i=0 ; i<alpha ; i++ )
309                                        if(tree[0][i] > 0){
310                                            q.push(tree[0][i]) ;
311                                        }
312
313                                    while(!q.empty()){
314                                        int u = q.front() ; q.pop() ;
315                                        order.push_back(u) ;
316
317                                        for ( int i=0 ; i<alpha ; i++ ){
318                                            if(tree[u][i] == 0) tree[u][i] =
319                                                tree[fail[u]][i] ;
320                                            else{
321                                                fail[tree[u][i]] = tree[fail[u]][i]
322                                                    ;
323                                                q.push(tree[u][i]) ;
324                                            }
325                                        }
326
327                                    }
328
329                                    vector<int> count_per_pattern(const string
330                                        &text) const {
331                                        int u = 0 ;
332                                        vector<int> vis(tree.size(), 0) ;
333
334                                        for ( char ch : text ){
335                                            u = tree[u][ch - base] ;
336                                            vis[u]++;
337                                        }
338
339                                        for ( int i=order.size()-1 ; i>=1 ; i-- )
340                                        int x = order[i] ;
341                                        vis[fail[x]] += vis[x] ;
342
343                                        vector<int> ans(end.size(), 0) ;
344                                        for ( int id=0 ; id<end.size() ; id++ )
345                                            ans[id] = vis[end[id]] ;
346
347                                    }
348
349                                    void clear(){
350                                        fail.assign(1, 0) ;
351                                        order.clear() ;
352                                        end.clear() ;
353                                        tree.assign(1, vector<int>(alpha, 0)) ;
354
355                                    }
356
357                                    void add_pattern(const string &pattern){
358                                        int u = 0 ;
359                                        for ( auto ch : pattern ){
360                                            int v = ch - base ;
361
362                                            if(tree[u][v] == 0) tree[u][v] =
363                                                new_node() ;
364                                            u = tree[u][v] ;
365
366                                        }
367
368                                        end.push_back(u) ;
369
370                                        void build(){
371                                            queue<int> q ;
372                                            order.clear() ;
373                                            order.push_back(0) ;
374
375                                            for ( int i=0 ; i<alpha ; i++ )
376                                                if(tree[0][i] > 0){
377                                                    q.push(tree[0][i]) ;
378                                                }
379
380                                            while(!q.empty()){
381                                                int u = q.front() ; q.pop() ;
382                                                order.push_back(u) ;
383
384                                                for ( int i=0 ; i<alpha ; i++ ){
385                                                    if(tree[u][i] == 0) tree[u][i] =
386                                                        tree[fail[u]][i] ;
387                                                    else{
388                                                        fail[tree[u][i]] = tree[fail[u]][i]
389                                                            ;
390                                                        q.push(tree[u][i]) ;
391                                                    }
392                                                }
393
394                                            }
395
396                                            vector<int> count_per_pattern(const string
397                                                &text) const {
398                                                int u = 0 ;
399                                                vector<int> vis(tree.size(), 0) ;
400
401                                                for ( char ch : text ){
402                                                    u = tree[u][ch - base] ;
403                                                    vis[u]++;
404                                                }
405
406                                                for ( int i=order.size()-1 ; i>=1 ; i-- )
407                                                int x = order[i] ;
408                                                vis[fail[x]] += vis[x] ;
409
410                                                vector<int> ans(end.size(), 0) ;
411                                                for ( int id=0 ; id<end.size() ; id++ )
412                                                    ans[id] = vis[end[id]] ;
413
414                                            }
415
416                                            void clear(){
417                                                fail.assign(1, 0) ;
418                                                order.clear() ;
419                                                end.clear() ;
420                                                tree.assign(1, vector<int>(alpha, 0)) ;
421
422                                            }
423
424                                            void add_pattern(const string &pattern){
425                                                int u = 0 ;
426                                                for ( auto ch : pattern ){
427                                                    int v = ch - base ;
428
429                                                    if(tree[u][v] == 0) tree[u][v] =
430                                                        new_node() ;
431                                                    u = tree[u][v] ;
432
433                                                }
434
435                                                end.push_back(u) ;
436
437                                                void build(){
438                                                    queue<int> q ;
439                                                    order.clear() ;
440                                                    order.push_back(0) ;
441
442                                                    for ( int i=0 ; i<alpha ; i++ )
443                                                        if(tree[0][i] > 0){
444                                                            q.push(tree[0][i]) ;
445                                                        }
446
447                                                    while(!q.empty()){
448                                                        int u = q.front() ; q.pop() ;
449                                                        order.push_back(u) ;
450
451                                                        for ( int i=0 ; i<alpha ; i++ ){
452                                                            if(tree[u][i] == 0) tree[u][i] =
453                                                                tree[fail[u]][i] ;
454                                                            else{
455                                                                fail[tree[u][i]] = tree[fail[u]][i]
456                                                                    ;
457                                                                q.push(tree[u][i]) ;
458                                                            }
459                                                        }
460
461                                                    }
462
463                                                    vector<int> count_per_pattern(const string
464                                                        &text) const {
465                                                        int u = 0 ;
466                                                        vector<int> vis(tree.size(), 0) ;
467
468                                                        for ( char ch : text ){
469                                                            u = tree[u][ch - base] ;
470                                                            vis[u]++;
471                                                        }
472
473                                                        for ( int i=order.size()-1 ; i>=1 ; i-- )
474                                                        int x = order[i] ;
475                                                        vis[fail[x]] += vis[x] ;
476
477                                                        vector<int> ans(end.size(), 0) ;
478                                                        for ( int id=0 ; id<end.size() ; id++ )
479                                                            ans[id] = vis[end[id]] ;
480
481                                                    }
482
483                                                    void clear(){
484                                                        fail.assign(1, 0) ;
485                                                        order.clear() ;
486                                                        end.clear() ;
487                                                        tree.assign(1, vector<int>(alpha, 0)) ;
488
489                                                    }
490
491                                                    void add_pattern(const string &pattern){
492                                                        int u = 0 ;
493                                                        for ( auto ch : pattern ){
494                                                            int v = ch - base ;
495
496                                                            if(tree[u][v] == 0) tree[u][v] =
497                                                                new_node() ;
498                                                            u = tree[u][v] ;
499
500                                                        }
501
502                                                        end.push_back(u) ;
503
504                                                        void build(){
505                                                            queue<int> q ;
506                                                            order.clear() ;
507                                                            order.push_back(0) ;
508
509                                                            for ( int i=0 ; i<alpha ; i++ )
510                                                                if(tree[0][i] > 0){
511                                                                    q.push(tree[0][i]) ;
512                                                                }
513
514                                                            while(!q.empty()){
515                                                                int u = q.front() ; q.pop() ;
516                                                                order.push_back(u) ;
517
518                                                                for ( int i=0 ; i<alpha ; i++ ){
519                                                                    if(tree[u][i] == 0) tree[u][i] =
520                                                                        tree[fail[u]][i] ;
521                                                                    else{
522                                                                        fail[tree[u][i]] = tree[fail[u]][i]
523                                                                            ;
524                                                                        q.push(tree[u][i]) ;
525                                                                    }
526                                                                }
527
528                                                            }
529
530                                                            vector<int> count_per_pattern(const string
531                                                                &text) const {
532                                                                int u = 0 ;
533                                                                vector<int> vis(tree.size(), 0) ;
534
535                                                                for ( char ch : text ){
536                                                                    u = tree[u][ch - base] ;
537                                                                    vis[u]++;
538                                                                }
539
540                                                                for ( int i=order.size()-1 ; i>=1 ; i-- )
541                                                                int x = order[i] ;
542                                                                vis[fail[x]] += vis[x] ;
543
544                                                                vector<int> ans(end.size(), 0) ;
545                                                                for ( int id=0 ; id<end.size() ; id++ )
546                                                                    ans[id] = vis[end[id]] ;
547
548                                                            }
549
550                                                            void clear(){
551                                                                fail.assign(1, 0) ;
552                                                                order.clear() ;
553                                                                end.clear() ;
554                                                                tree.assign(1, vector<int>(alpha, 0)) ;
555
556                                                            }
557
558                                                            void add_pattern(const string &pattern){
559                                                                int u = 0 ;
560                                                                for ( auto ch : pattern ){
561                                                                    int v = ch - base ;
562
563                                                                    if(tree[u][v] == 0) tree[u][v] =
564                                                                        new_node() ;
565                                                                    u = tree[u][v] ;
566
567                                                                }
568
569                                                                end.push_back(u) ;
570
571                                                                void build(){
572                                                                    queue<int> q ;
573                                                                    order.clear() ;
574                                                                    order.push_back(0) ;
575
576                                                                    for ( int i=0 ; i<alpha ; i++ )
577                                                                        if(tree[0][i] > 0){
578                                                                            q.push(tree[0][i]) ;
579                                                                        }
580
581                                                                    while(!q.empty()){
582                                                                        int u = q.front() ; q.pop() ;
583                                                                        order.push_back(u) ;
584
585                                                                        for ( int i=0 ; i<alpha ; i++ ){
586                                                                            if(tree[u][i] == 0) tree[u][i] =
587                                                                                tree[fail[u]][i] ;
588                                                                            else{
589                                                                                fail[tree[u][i]] = tree[fail[u]][i]
590                                                                                    ;
591                                                                                q.push(tree[u][i]) ;
592                                                                            }
593                                                                        }
594
595                                                                    }
596
597                                                                    vector<int> count_per_pattern(const string
598                                                                        &text) const {
599                                                                        int u = 0 ;
600                                                                        vector<int> vis(tree.size(), 0) ;
601
602                                                                        for ( char ch : text ){
603                                                                            u = tree[u][ch - base] ;
604                                                                            vis[u]++;
605                                                                        }
606
607                                                                        for ( int i=order.size()-1 ; i>=1 ; i-- )
608                                                                        int x = order[i] ;
609                                                                        vis[fail[x]] += vis[x] ;
610
611                                                                        vector<int> ans(end.size(), 0) ;
612                                                                        for ( int id=0 ; id<end.size() ; id++ )
613                                                                            ans[id] = vis[end[id]] ;
614
615                                                                    }
616
617                                                                    void clear(){
618                                                                        fail.assign(1, 0) ;
619                                                                        order.clear() ;
620                                                                        end.clear() ;
621                                                                        tree.assign(1, vector<int>(alpha, 0)) ;
622
623                                                                    }
624
625                                                                    void add_pattern(const string &pattern){
626                                                                        int u = 0 ;
627                                                                        for ( auto ch : pattern ){
628                                                                            int v = ch - base ;
629
630                                                                            if(tree[u][v] == 0) tree[u][v] =
631                                                                                new_node() ;
632                                                                            u = tree[u][v] ;
633
634                                                                        }
635
636                                                                        end.push_back(u) ;
637
638                                                                        void build(){
639                                                                            queue<int> q ;
640                                                                            order.clear() ;
641                                                                            order.push_back(0) ;
642
643                                                                            for ( int i=0 ; i<alpha ; i++ )
644                                                                                if(tree[0][i] > 0){
645                                                                                    q.push(tree[0][i]) ;
646                                                                                }
647
648                                                                            while(!q.empty()){
649                                                                                int u = q.front() ; q.pop() ;
650                                                                                order.push_back(u) ;
651
652                                                                                for ( int i=0 ; i<alpha ; i++ ){
653                                                                                    if(tree[u][i] == 0) tree[u][i] =
654                                                                                        tree[fail[u]][i] ;
655                                                                                    else{
656                                                                                        fail[tree[u][i]] = tree[fail[u]][i]
657                                                                                            ;
658                                                                                        q.push(tree[u][i]) ;
659                                                                                    }
660
661                                                                            }
662
663                                                                        }
664
665                                                                        vector<int> count_per_pattern(const string
666                                                                            &text) const {
667                                                                            int u = 0 ;
668                                                                            vector<int> vis(tree.size(), 0) ;
669
670                                                                            for ( char ch : text ){
671                                                                                u = tree[u][ch - base] ;
672                                                                                vis[u]++;
673                                                                            }
674
675                                                                            for ( int i=order.size()-1 ; i>=1 ; i-- )
676                                                                            int x = order[i] ;
677                                                                            vis[fail[x]] += vis[x] ;
678
679                                                                            vector<int> ans(end.size(), 0) ;
680                                                                            for ( int id=0 ; id<end.size() ; id++ )
681                                                                                ans[id] = vis[end[id]] ;
682
683                                                                        }
684
685                                                                        void clear(){
686                                                                        fail.assign(1, 0) ;
687                                                                        order.clear() ;
688                                                                        end.clear() ;
689                                                                        tree.assign(1, vector<int>(alpha, 0)) ;
690
691                                                                        }
692
693                                                                        void add_pattern(const string &pattern){
694                                                                        int u = 0 ;
695                                                                        for ( auto ch : pattern ){
696                                                                            int v = ch - base ;
697
698                                                                            if(tree[u][v] == 0) tree[u][v] =
699                                                                                new_node() ;
700                                                                            u = tree[u][v] ;
701
702                                                                        }
703
704                                                                        end.push_back(u) ;
705
706                                                                        void build(){
707                                                                            queue<int> q ;
708                                                                            order.clear() ;
709                                                                            order.push_back(0) ;
710
711                                                                            for ( int i=0 ; i<alpha ; i++ )
712                                                                                if(tree[0][i] > 0){
713                                                                                    q.push(tree[0][i]) ;
714                                                                                }
715
716                                                                            while(!q.empty()){
717                                                                                int u = q.front() ; q.pop() ;
718                                                                                order.push_back(u) ;
719
720                                                                                for ( int i=0 ; i<alpha ; i++ ){
721                                                                                    if(tree[u][i] == 0) tree[u][i] =
722                                                                                        tree[fail[u]][i] ;
723                                                                                    else{
724                                                                                        fail[tree[u][i]] = tree[fail[u]][i]
725                                                                                            ;
726                                                                                        q.push(tree[u][i]) ;
727                                                                                    }
728
729                                                                            }
730
731                                                                        }
732
733                                                                        vector<int> count_per_pattern(const string
734                                                                            &text) const {
735                                                                            int u = 0 ;
736                                                                            vector<int> vis(tree.size(), 0) ;
737
738                                                                            for ( char ch : text ){
739                                                                                u = tree[u][ch - base] ;
740                                                                                vis[u]++;
741                                                                            }
742
743                                                                            for ( int i=order.size()-1 ; i>=1 ; i-- )
744                                                                            int x = order[i] ;
745                                                                            vis[fail[x]] += vis[x] ;
746
747                                                                            vector<int> ans(end.size(), 0) ;
748                                                                            for ( int id=0 ; id<end.size() ; id++ )
749                                                                                ans[id] = vis[end[id]] ;
750
751                                                                        }
752
753                                                                        void clear(){
754                                                                        fail.assign(1, 0) ;
755                                                                        order.clear() ;
756                                                                        end.clear() ;
757                                                                        tree.assign(1, vector<int>(alpha, 0)) ;
758
759                                                                        }
760
761                                                                        void add_pattern(const string &pattern){
762                                                                        int u = 0 ;
763                                                                        for ( auto ch : pattern ){
764                                                                            int v = ch - base ;
765
766                                                                            if(tree[u][v] == 0) tree[u][v] =
767                                                                                new_node() ;
768                                                                            u = tree[u][v] ;
769
770                                                                        }
771
772                                                                        end.push_back(u) ;
773
774                                                                        void build(){
775                                                                            queue<int> q ;
776                                                                            order.clear() ;
777                                                                            order.push_back(0) ;
778
779                                                                            for ( int i=0 ; i<alpha ; i++ )
780                                                                                if(tree[0][i] > 0){
781                                                                                    q.push(tree[0][i]) ;
782                                                                                }
783
784                                                                            while(!q.empty()){
785                                                                                int u = q.front() ; q.pop() ;
786                                                                                order.push_back(u) ;
787
788                                                                                for ( int i=0 ; i<alpha ; i++ ){
789                                                                                    if(tree[u][i] == 0) tree[u][i] =
790                                                                                        tree[fail[u]][i] ;
791                                                                                    else{
792                                                                                        fail[tree[u][i]] = tree[fail[u]][i]
793                                                                                            ;
794                                                                                        q.push(tree[u][i]) ;
795                                                                                    }
796
797                                                                            }
798
799                                                                        }
800
801                                                                        vector<int> count_per_pattern(const string
802                                                                            &text) const {
803                                                                            int u = 0 ;
804                                                                            vector<int> vis(tree.size(), 0) ;
805
806                                                                            for ( char ch : text ){
807                                                                                u = tree[u][ch - base] ;
808                                                                                vis[u]++;
809                                                                            }
810
811                                                                            for ( int i=order.size()-1 ; i>=1 ; i-- )
812                                                                            int x = order[i] ;
813                                                                            vis[fail[x]] += vis[x] ;
814
815                                                                            vector<int> ans(end.size(), 0) ;
816                                                                            for ( int id=0 ; id<end.size() ; id++ )
817                                                                                ans[id] = vis[end[id]] ;
818
819                                                                        }
820
821                                                                        void clear(){
822                                                                        fail.assign(1, 0) ;
823                                                                        order.clear() ;
824                                                                        end.clear() ;
825                                                                        tree.assign(1, vector<int>(alpha, 0)) ;
826
827                                                                        }
828
829                                                                        void add_pattern(const string &pattern){
830                                                                        int u = 0 ;
831                                                                        for ( auto ch : pattern ){
832                                                                            int v = ch - base ;
833
834                                                                            if(tree[u][v] == 0) tree[u][v] =
835                                                                                new_node() ;
836                                                                            u = tree[u][v] ;
837
838                                                                        }
839
840                                                                        end.push_back(u) ;
841
842                                                                        void build(){
843                                                                            queue<int> q ;
844                                                                            order.clear() ;
845                                                                            order.push_back(0) ;
846
847                                                                            for ( int i=0 ; i<alpha ; i++ )
848                                                                                if(tree[0][i] > 0){
849                                                                                    q.push(tree[0][i]) ;
850                                                                                }
851
852                                                                            while(!q.empty()){
853                                                                                int u = q.front() ; q.pop() ;
854                                                                                order.push_back(u) ;
855
856                                                                                for ( int i=0 ; i<alpha ; i++ ){
857                                                                                    if(tree[u][i] == 0) tree[u][i] =
858                                                                                        tree[fail[u]][i] ;
859                                                                                    else{
860                                                                                        fail[tree[u][i]] = tree[fail[u]][i]
861                                                                                            ;
862                                                                                        q.push(tree[u][i]) ;
863                                                                                    }
864
865                                                                            }
866
867                                                                        }
868
868
869                                                                        vector<int> count_per_pattern(const string
870                                                                            &text) const {
871                                                                            int u = 0 ;
872                                                                            vector<int> vis(tree.size(), 0) ;
873
874                                                                            for ( char ch : text ){
875                                                                                u = tree[u][ch - base] ;
876                                                                                vis[u]++;
877                                                                            }
878
879                                                                            for ( int i=order.size()-1 ; i>=1 ; i-- )
880                                                                            int x = order[i] ;
881                                                                            vis[fail[x]] += vis[x] ;
882
883                                                                            vector<int> ans(end.size(), 0) ;
884                                                                            for ( int id=0 ; id<end.size() ; id++ )
885                                                                                ans[id] = vis[end[id]] ;
886
887                                                                        }
888
889                                                                        void clear(){
890                                                                        fail.assign(1, 0) ;
891                                                                        order.clear() ;
892                                                                        end.clear() ;
893                                                                        tree.assign(1, vector<int>(alpha, 0)) ;
894
895                                                                        }
896
897                                                                        void add_pattern(const string &pattern){
898                                                                        int u = 0 ;
899                                                                        for ( auto ch : pattern ){
900                                                                            int v = ch - base ;
901
902                                                                            if(tree[u][v] == 0) tree[u][v] =
903                                                                                new_node() ;
904                                                                            u = tree[u][v] ;
905
906                                                                        }
907
908                                                                        end.push_back(u) ;
909
910                                                                        void build(){
911                                                                            queue<int> q ;
912                                                                            order.clear() ;
913                                                                            order.push_back(0) ;
914
915                                                                            for ( int i=0 ; i<alpha ; i++ )
916                                                                                if(tree[0][i] > 0){
917                                                                                    q.push(tree[0][i]) ;
918                                                                                }
919
920                                                                            while(!q.empty()){
921                                                                                int u = q.front() ; q.pop() ;
922                                                                                order.push_back(u) ;
923
924                                                                                for ( int i=0 ; i<alpha ; i++ ){
925                                                                                    if(tree[u][i] == 0) tree[u][i] =
926                                                                                        tree[fail[u]][i] ;
927                                                                                    else{
928                                                                                        fail[tree[u][i]] = tree[fail[u]][i]
929                                                                                            ;
930                                                                                        q.push(tree[u][i]) ;
931                                                                                    }
932
933                                                                            }
934
935                                                                        }
936
936
937                                                                        vector<int> count_per_pattern(const string
938                                                                            &text) const {
939                                                                            int u = 0 ;
940                                                                            vector<int> vis(tree.size(), 0) ;
941
942                                                                            for ( char ch : text ){
943                                                                                u = tree[u][ch - base] ;
944                                                                                vis[u]++;
945                                                                            }
946
947                                                                            for ( int i=order.size()-1 ; i>=1 ; i-- )
948                                                                            int x = order[i] ;
949                                                                            vis[fail[x]] += vis[x] ;
950
951                                                                            vector<int> ans(end.size(), 0) ;
952                                                                            for ( int id=0 ; id<end.size() ; id++ )
953                                                                                ans[id] = vis[end[id]] ;
954
955                                                                        }
956
957                                                                        void clear(){
958                                                                        fail.assign(1, 0) ;
959                                                                        order.clear() ;
960                                                                        end.clear() ;
961                                                                        tree.assign(1, vector<int>(alpha, 0)) ;
962
963                                                                        }
964
965                                                                        void add_pattern(const string &pattern){
966                                                                        int u = 0 ;
967                                                                        for ( auto ch : pattern ){
968                                                                            int v = ch - base ;
969
970                                                                            if(tree[u][v] == 0) tree[u][v] =
971                                                                                new_node() ;
972                                                                            u = tree[u][v] ;
973
974                                                                        }
975
976                                                                        end.push_back(u) ;
977
978                                                                        void build(){
979                                                                            queue<int> q ;
980                                                                            order.clear() ;
981                                                                            order.push_back(0) ;
982
983                                                                            for ( int i=0 ; i<alpha ; i++ )
984                                                                                if(tree[0][i] > 0){
985                                                                                    q.push(tree[0][i]) ;
986                                                                                }
987
988                                                                            while(!q.empty()){
989                                                                                int u = q.front() ; q.pop() ;
990                                                                                order.push_back(u) ;
991
992                                                                                for ( int i=0 ; i<alpha ; i++ ){
993                                                                                    if(tree[u][i] == 0) tree[u][i] =
994                                                                                        tree[fail[u]][i] ;
995                                                                                    else{
996                                                                                        fail[tree[u][i]] = tree[fail[u]][i]
997                                                                                            ;
998                                                                                        q.push(tree[u][i]) ;
999                                                                                    }
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150

```

```

79     }
80
81     return ans ;
82 }
83 }
```

## 6 Algorithm

### 6.1 LCA

```

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

### 6.2 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    }
38
39    void MST(){
40        int cnt = 0 ;
41        for ( int i=0 ; i<edge.size() && cnt < n-1
42                ; i++ ){
43            auto [u, v, w] = edge[i] ;
44            if(merge(u, v)){
45                cnt++ ;
46                sum -= w ;
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(),
58                 greater<Edge>()) ;
59        }
60    }
61 }
```

### 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     int SG[30];
31     int vis[Maxn], stone[Maxn];
32
33     void build(){
34         SG[0] = 0;
35         memset(vis, 0, sizeof(vis));
36
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         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     }
64 }
```

### 6.4 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
22    int dfs(int v, int f){
23        if(v == T) return f ;
24
25        for ( int i=cur[v] ; i<e[v].size()
26                ; i++ ){
27            Edge &e = e[v][i] ;
28
29            if(e.cap > 0 && dep[e.v] < dep[v] + 1
30                && dfs(e.v, min(f, e.cap)) > 0){
31                e.cap -= 1 ;
32                cur[v] = i+1 ;
33                return f ;
34            }
35        }
36
37        return 0 ;
38    }
39
40    int maxFlow(){
41        int ans = 0 ;
42
43        while(bfs()){
44            for ( int v=S ; v<T ; v++ ){
45                if(dfs(v, INT_MAX) > 0) ans += 1 ;
46            }
47        }
48
49        return ans ;
50    }
51 }
```

```

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         public:
62         MaxFlow(int n, int s, int t){
63             N = n ; S = s ; T = t ;
64             e.reserve(n*n) ;
65             head.assign(n+1, -1) ;
66             cur.resize(n+1) ;
67             dep.resize(n+1) ;
68         }
69
70         void AddEdge(int u, int v, int cap){
71             e.push_back({v, cap, head[u]}) ;
72             head[u] = e.size() - 1 ;
73             e.push_back({u, 0, head[v]}) ;
74             head[v] = e.size() - 1 ;
75         }
76
77         int run(){
78             int ans = 0 ;
79             while(bfs()){
80                 ans += dfs(S, 0x3f3f3f3f) ;
81             }
82             return ans ;
83         }
84     }

```

## 6.5 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
12    public:
13        MCMF(int n, int _s, int _t){
14            N = n ;
15            s = _s ;
16            t = _t ;
17            e.resize(n*n + 5) ;
18            head.assign(n+5, -1) ;
19            tot = -1 ;
20        }
21
22        void AddEdge(int u, int v, int cap, int
23                      cost){
24            e[++tot] = {v, cap, cost, head[u]} ;
25            head[u] = tot ;
26            e[++tot] = {u, 0, -cost, head[v]} ;
27            head[v] = tot ;
28        }
29
30        int run(){
31            vector<int> dis(N+1), pot(N+1, 0),
32                        preE(N+1) ;
33            int flow = 0, cost = 0 ;
34
35            auto dijkstra = [&](){
36                fill(dis.begin(), dis.end(), INF) ;
37                priority_queue<pii, vector<pii>,
38                                greater<pii>> pq ;
39                dis[s] = 0 ;
40                pq.push({0, s}) ;
41
42                while(!pq.empty()){
43                    auto [d, u] = pq.top() ; pq.pop() ;
44                    if(d > dis[u]) continue ;
45                    for ( int i=head[u] ; i!=-1 ;
46                        i=e[i].next ){
47                        int v = e[i].v, cap = e[i].cap, w =
48                            e[i].cost ;
49                        if(cap && dis[v] > d + w + pot[u] -
50                            pot[v]){
51                            dis[v] = d + w + pot[u] - pot[v] ;
52                            preE[v] = i ;
53                            pq.push({dis[v], v}) ;
54                        }
55                    }
56
57                    return dis[t] != INF ;
58                };
59
60                while(dijkstra()){
61                    for ( int v=1 ; v<=N ; v++ ) if(dis[v]
62                        < INF){
63                        pot[v] += dis[v] ;
64
65                        int aug = INT_MAX ;
66                        for ( int v=t ; v!=s ;
67                            v=e[preE[v]^1].v ){
68                            aug = min(aug, e[preE[v]].cap) ;
69
70                            for ( int v=t ; v!=s ;
71                                v=e[preE[v]^1].v ){
72                                e[preE[v]].cap -= aug ;
73                                e[preE[v]^1].cap += aug ;
74                                cost += aug * e[preE[v]].cost ;
75                            }
76
77                            return cost ;
78                        }
79
80                    }
81                }
82            };
83
84        }

```

## 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            printf("%lld\n", dp[cur][(1 << m) - 1]) ;
36        }
37        return 0 ;
38    }

```

### 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    return d = res ;
27}

```

```

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%d", &a, &b, &K) ;
46         if(K > 90) printf("0\n") ;
47         else printf("%d\n", count(b) -
48             count(a-1)) ;
49     }
50
51     return 0 ;
}

```

```

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", &u, &v, &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     }
69 }

```

### 7.3 樹 DP

```

1 #include <bits/stdc++.h>
2
3 #define N 505
4 #define INF 0x3f3f3f3f
5
6 using namespace std ;
7
8 struct Edge{
9     int v, w ;
10 } ;
11
12 vector<Edge> edge[N] ;
13 int n ;
14 int cnt[N] ;
15 int dp[N][N][2] ;
16
17 void init(){
18     for ( int i=0 ; i<N ; i++ )
19         edge[i].clear() ;
20     memset(cnt, 0, sizeof(cnt)) ;
21     memset(dp, INF, sizeof(dp)) ;
22 }
23
24 void DFS(int u){
25     cnt[u] = 1 ;
26     for ( auto [v, w] : edge[u] ){
27         DFS(v) ;
28         cnt[u] += cnt[v] ;
29     }
30
31     dp[u][1][0] = dp[u][1][1] = 0 ;
32
33     for ( auto [v, w] : edge[u] ){
34         for ( int i=cnt[u] ; i>1 ; i-- ) for (
35             int j=1 ; j<i && j<=cnt[v] ; j++ ){
36             dp[u][i][1] = min(dp[u][i][1],
37                 dp[u][i-j][1] + dp[v][j][1] + 2 *
38                 w) ;
39             dp[u][i][0] = min(dp[u][i][0],
40                 dp[u][i-j][1] + dp[v][j][0] + w) ;
41             dp[u][i][0] = min(dp[u][i][0],
42                 dp[u][i-j][0] + dp[v][j][1] + 2 *
43                 w) ;
44         }
45     }
46 }

```