

Contents

1 Basic	1.1 PyMath
2 Tree	2.1 SegmentTree
	2.2 HLD
	2.3 Trie
3 String	3.1 KMP
4 Algorithm	4.1 LCA
	4.2 MST
	4.3 SG
	4.4 Convex

1 Basic

1.1 PyMath

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

2 Tree

2.1 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
25     void AddTag(int id, LazyTag &tag){
26         if(tag.type == 0){
27             seg[id].sum += tag.val *
28                         seg[id].sz ;
29             seg[id].tag.val += tag.val
30         }
31         else{
32             seg[id].sum = tag.val *
33                         seg[id].sz ;
34             seg[id].tag = {1, tag.val}
35         }
36     }
37
38     void push(int id){
39         AddTag(lc, seg[id].tag) ;
40         AddTag(rc, seg[id].tag) ;
41         seg[id].tag = {0, 0} ;
42     }
43
44     void update(int id, int l, int r, int val, int index, int tag)
45     {
46         if(l > r) return;
47         if(l == r)
48         {
49             seg[id].tag = {1, val};
50             seg[id].sum = val;
51             seg[id].sz = 1;
52             return;
53         }
54         int mid = (l + r) / 2;
55         push(id * 2, l, mid, val, index, tag);
56         push(id * 2 + 1, mid + 1, r, val, index, tag);
57         seg[id].sum = seg[id * 2].sum + seg[id * 2 + 1].sum;
58     }
59
60     void query(int id, int l, int r, int index)
61     {
62         if(l > r) return;
63         if(l == r)
64         {
65             cout << seg[id].sum << endl;
66             return;
67         }
68         int mid = (l + r) / 2;
69         query(id * 2, l, mid, index);
70         query(id * 2 + 1, mid + 1, r, index);
71     }
72
73     void print(int id)
74     {
75         cout << "id: " << id << endl;
76         cout << "sum: " << seg[id].sum << endl;
77         cout << "sz: " << seg[id].sz << endl;
78         cout << "tag: " << seg[id].tag << endl;
79     }
80
81     void printAll()
82     {
83         for(int i = 0; i < Maxn << 2; i++)
84             print(i);
85     }
86
87     void printAll()
88     {
89         for(int i = 0; i < Maxn << 2; i++)
90             print(i);
91     }
92
93     void printAll()
94     {
95         for(int i = 0; i < Maxn << 2; i++)
96             print(i);
97     }
98
99     void printAll()
100    {
101        for(int i = 0; i < Maxn << 2; i++)
102            print(i);
103    }
104
105    void printAll()
106    {
107        for(int i = 0; i < Maxn << 2; i++)
108            print(i);
109    }
110
111    void printAll()
112    {
113        for(int i = 0; i < Maxn << 2; i++)
114            print(i);
115    }
116
117    void printAll()
118    {
119        for(int i = 0; i < Maxn << 2; i++)
120            print(i);
121    }
122
123    void printAll()
124    {
125        for(int i = 0; i < Maxn << 2; i++)
126            print(i);
127    }
128
129    void printAll()
130    {
131        for(int i = 0; i < Maxn << 2; i++)
132            print(i);
133    }
134
135    void printAll()
136    {
137        for(int i = 0; i < Maxn << 2; i++)
138            print(i);
139    }
140
141    void printAll()
142    {
143        for(int i = 0; i < Maxn << 2; i++)
144            print(i);
145    }
146
147    void printAll()
148    {
149        for(int i = 0; i < Maxn << 2; i++)
150            print(i);
151    }
152
153    void printAll()
154    {
155        for(int i = 0; i < Maxn << 2; i++)
156            print(i);
157    }
158
159    void printAll()
160    {
161        for(int i = 0; i < Maxn << 2; i++)
162            print(i);
163    }
164
165    void printAll()
166    {
167        for(int i = 0; i < Maxn << 2; i++)
168            print(i);
169    }
170
171    void printAll()
172    {
173        for(int i = 0; i < Maxn << 2; i++)
174            print(i);
175    }
176
177    void printAll()
178    {
179        for(int i = 0; i < Maxn << 2; i++)
180            print(i);
181    }
182
183    void printAll()
184    {
185        for(int i = 0; i < Maxn << 2; i++)
186            print(i);
187    }
188
189    void printAll()
190    {
191        for(int i = 0; i < Maxn << 2; i++)
192            print(i);
193    }
194
195    void printAll()
196    {
197        for(int i = 0; i < Maxn << 2; i++)
198            print(i);
199    }
200
201    void printAll()
202    {
203        for(int i = 0; i < Maxn << 2; i++)
204            print(i);
205    }
206
207    void printAll()
208    {
209        for(int i = 0; i < Maxn << 2; i++)
210            print(i);
211    }
212
213    void printAll()
214    {
215        for(int i = 0; i < Maxn << 2; i++)
216            print(i);
217    }
218
219    void printAll()
220    {
221        for(int i = 0; i < Maxn << 2; i++)
222            print(i);
223    }
224
225    void printAll()
226    {
227        for(int i = 0; i < Maxn << 2; i++)
228            print(i);
229    }
230
231    void printAll()
232    {
233        for(int i = 0; i < Maxn << 2; i++)
234            print(i);
235    }
236
237    void printAll()
238    {
239        for(int i = 0; i < Maxn << 2; i++)
240            print(i);
241    }
242
243    void printAll()
244    {
245        for(int i = 0; i < Maxn << 2; i++)
246            print(i);
247    }
248
249    void printAll()
250    {
251        for(int i = 0; i < Maxn << 2; i++)
252            print(i);
253    }
254
255    void printAll()
256    {
257        for(int i = 0; i < Maxn << 2; i++)
258            print(i);
259    }
260
261    void printAll()
262    {
263        for(int i = 0; i < Maxn << 2; i++)
264            print(i);
265    }
266
267    void printAll()
268    {
269        for(int i = 0; i < Maxn << 2; i++)
270            print(i);
271    }
272
273    void printAll()
274    {
275        for(int i = 0; i < Maxn << 2; i++)
276            print(i);
277    }
278
279    void printAll()
280    {
281        for(int i = 0; i < Maxn << 2; i++)
282            print(i);
283    }
284
285    void printAll()
286    {
287        for(int i = 0; i < Maxn << 2; i++)
288            print(i);
289    }
290
291    void printAll()
292    {
293        for(int i = 0; i < Maxn << 2; i++)
294            print(i);
295    }
296
297    void printAll()
298    {
299        for(int i = 0; i < Maxn << 2; i++)
300            print(i);
301    }
302
303    void printAll()
304    {
305        for(int i = 0; i < Maxn << 2; i++)
306            print(i);
307    }
308
309    void printAll()
310    {
311        for(int i = 0; i < Maxn << 2; i++)
312            print(i);
313    }
314
315    void printAll()
316    {
317        for(int i = 0; i < Maxn << 2; i++)
318            print(i);
319    }
320
321    void printAll()
322    {
323        for(int i = 0; i < Maxn << 2; i++)
324            print(i);
325    }
326
327    void printAll()
328    {
329        for(int i = 0; i < Maxn << 2; i++)
330            print(i);
331    }
332
333    void printAll()
334    {
335        for(int i = 0; i < Maxn << 2; i++)
336            print(i);
337    }
338
339    void printAll()
340    {
341        for(int i = 0; i < Maxn << 2; i++)
342            print(i);
343    }
344
345    void printAll()
346    {
347        for(int i = 0; i < Maxn << 2; i++)
348            print(i);
349    }
350
351    void printAll()
352    {
353        for(int i = 0; i < Maxn << 2; i++)
354            print(i);
355    }
356
357    void printAll()
358    {
359        for(int i = 0; i < Maxn << 2; i++)
360            print(i);
361    }
362
363    void printAll()
364    {
365        for(int i = 0; i < Maxn << 2; i++)
366            print(i);
367    }
368
369    void printAll()
370    {
371        for(int i = 0; i < Maxn << 2; i++)
372            print(i);
373    }
374
375    void printAll()
376    {
377        for(int i = 0; i < Maxn << 2; i++)
378            print(i);
379    }
380
381    void printAll()
382    {
383        for(int i = 0; i < Maxn << 2; i++)
384            print(i);
385    }
386
387    void printAll()
388    {
389        for(int i = 0; i < Maxn << 2; i++)
390            print(i);
391    }
392
393    void printAll()
394    {
395        for(int i = 0; i < Maxn << 2; i++)
396            print(i);
397    }
398
399    void printAll()
400    {
401        for(int i = 0; i < Maxn << 2; i++)
402            print(i);
403    }
404
405    void printAll()
406    {
407        for(int i = 0; i < Maxn << 2; i++)
408            print(i);
409    }
410
411    void printAll()
412    {
413        for(int i = 0; i < Maxn << 2; i++)
414            print(i);
415    }
416
417    void printAll()
418    {
419        for(int i = 0; i < Maxn << 2; i++)
420            print(i);
421    }
422
423    void printAll()
424    {
425        for(int i = 0; i < Maxn << 2; i++)
426            print(i);
427    }
428
429    void printAll()
430    {
431        for(int i = 0; i < Maxn << 2; i++)
432            print(i);
433    }
434
435    void printAll()
436    {
437        for(int i = 0; i < Maxn << 2; i++)
438            print(i);
439    }
440
441    void printAll()
442    {
443        for(int i = 0; i < Maxn << 2; i++)
444            print(i);
445    }
446
447    void printAll()
448    {
449        for(int i = 0; i < Maxn << 2; i++)
450            print(i);
451    }
452
453    void printAll()
454    {
455        for(int i = 0; i < Maxn << 2; i++)
456            print(i);
457    }
458
459    void printAll()
460    {
461        for(int i = 0; i < Maxn << 2; i++)
462            print(i);
463    }
464
465    void printAll()
466    {
467        for(int i = 0; i < Maxn << 2; i++)
468            print(i);
469    }
470
471    void printAll()
472    {
473        for(int i = 0; i < Maxn << 2; i++)
474            print(i);
475    }
476
477    void printAll()
478    {
479        for(int i = 0; i < Maxn << 2; i++)
480            print(i);
481    }
482
483    void printAll()
484    {
485        for(int i = 0; i < Maxn << 2; i++)
486            print(i);
487    }
488
489    void printAll()
490    {
491        for(int i = 0; i < Maxn << 2; i++)
492            print(i);
493    }
494
495    void printAll()
496    {
497        for(int i = 0; i < Maxn << 2; i++)
498            print(i);
499    }
500
501    void printAll()
502    {
503        for(int i = 0; i < Maxn << 2; i++)
504            print(i);
505    }
506
507    void printAll()
508    {
509        for(int i = 0; i < Maxn << 2; i++)
510            print(i);
511    }
512
513    void printAll()
514    {
515        for(int i = 0; i < Maxn << 2; i++)
516            print(i);
517    }
518
519    void printAll()
520    {
521        for(int i = 0; i < Maxn << 2; i++)
522            print(i);
523    }
524
525    void printAll()
526    {
527        for(int i = 0; i < Maxn << 2; i++)
528            print(i);
529    }
530
531    void printAll()
532    {
533        for(int i = 0; i < Maxn << 2; i++)
534            print(i);
535    }
536
537    void printAll()
538    {
539        for(int i = 0; i < Maxn << 2; i++)
540            print(i);
541    }
542
543    void printAll()
544    {
545        for(int i = 0; i < Maxn << 2; i++)
546            print(i);
547    }
548
549    void printAll()
550    {
551        for(int i = 0; i < Maxn << 2; i++)
552            print(i);
553    }
554
555    void printAll()
556    {
557        for(int i = 0; i < Maxn << 2; i++)
558            print(i);
559    }
560
561    void printAll()
562    {
563        for(int i = 0; i < Maxn << 2; i++)
564            print(i);
565    }
566
567    void printAll()
568    {
569        for(int i = 0; i < Maxn << 2; i++)
570            print(i);
571    }
572
573    void printAll()
574    {
575        for(int i = 0; i < Maxn << 2; i++)
576            print(i);
577    }
578
579    void printAll()
580    {
581        for(int i = 0; i < Maxn << 2; i++)
582            print(i);
583    }
584
585    void printAll()
586    {
587        for(int i = 0; i < Maxn << 2; i++)
588            print(i);
589    }
590
591    void printAll()
592    {
593        for(int i = 0; i < Maxn << 2; i++)
594            print(i);
595    }
596
597    void printAll()
598    {
599        for(int i = 0; i < Maxn << 2; i++)
600            print(i);
601    }
602
603    void printAll()
604    {
605        for(int i = 0; i < Maxn << 2; i++)
606            print(i);
607    }
608
609    void printAll()
610    {
611        for(int i = 0; i < Maxn << 2; i++)
612            print(i);
613    }
614
615    void printAll()
616    {
617        for(int i = 0; i < Maxn << 2; i++)
618            print(i);
619    }
620
621    void printAll()
622    {
623        for(int i = 0; i < Maxn << 2; i++)
624            print(i);
625    }
626
627    void printAll()
628    {
629        for(int i = 0; i < Maxn << 2; i++)
630            print(i);
631    }
632
633    void printAll()
634    {
635        for(int i = 0; i < Maxn << 2; i++)
636            print(i);
637    }
638
639    void printAll()
640    {
641        for(int i = 0; i < Maxn << 2; i++)
642            print(i);
643    }
644
645    void printAll()
646    {
647        for(int i = 0; i < Maxn << 2; i++)
648            print(i);
649    }
650
651    void printAll()
652    {
653        for(int i = 0; i < Maxn << 2; i++)
654            print(i);
655    }
656
657    void printAll()
658    {
659        for(int i = 0; i < Maxn << 2; i++)
660            print(i);
661    }
662
663    void printAll()
664    {
665        for(int i = 0; i < Maxn << 2; i++)
666            print(i);
667    }
668
669    void printAll()
670    {
671        for(int i = 0; i < Maxn << 2; i++)
672            print(i);
673    }
674
675    void printAll()
676    {
677        for(int i = 0; i < Maxn << 2; i++)
678            print(i);
679    }
680
681    void printAll()
682    {
683        for(int i = 0; i < Maxn << 2; i++)
684            print(i);
685    }
686
687    void printAll()
688    {
689        for(int i = 0; i < Maxn << 2; i++)
690            print(i);
691    }
692
693    void printAll()
694    {
695        for(int i = 0; i < Maxn << 2; i++)
696            print(i);
697    }
698
699    void printAll()
700    {
701        for(int i = 0; i < Maxn << 2; i++)
702            print(i);
703    }
704
705    void printAll()
706    {
707        for(int i = 0; i < Maxn << 2; i++)
708            print(i);
709    }
710
711    void printAll()
712    {
713        for(int i = 0; i < Maxn << 2; i++)
714            print(i);
715    }
716
717    void printAll()
718    {
719        for(int i = 0; i < Maxn << 2; i++)
720            print(i);
721    }
722
723    void printAll()
724    {
725        for(int i = 0; i < Maxn << 2; i++)
726            print(i);
727    }
728
729    void printAll()
730    {
731        for(int i = 0; i < Maxn << 2; i++)
732            print(i);
733    }
734
735    void printAll()
736    {
737        for(int i = 0; i < Maxn << 2; i++)
738            print(i);
739    }
740
741    void printAll()
742    {
743        for(int i = 0; i < Maxn << 2; i++)
744            print(i);
745    }
746
747    void printAll()
748    {
749        for(int i = 0; i < Maxn << 2; i++)
750            print(i);
751    }
752
753    void printAll()
754    {
755        for(int i = 0; i < Maxn << 2; i++)
756            print(i);
757    }
758
759    void printAll()
760    {
761        for(int i = 0; i < Maxn << 2; i++)
762            print(i);
763    }
764
765    void printAll()
766    {
767        for(int i = 0; i < Maxn << 2; i++)
768            print(i);
769    }
770
771    void printAll()
772    {
773        for(int i = 0; i < Maxn << 2; i++)
774            print(i);
775    }
776
777    void printAll()
778    {
779        for(int i = 0; i < Maxn << 2; i++)
780            print(i);
781    }
782
783    void printAll()
784    {
785        for(int i = 0; i < Maxn << 2; i++)
786            print(i);
787    }
788
789    void printAll()
790    {
791        for(int i = 0; i < Maxn << 2; i++)
792            print(i);
793    }
794
795    void printAll()
796    {
797        for(int i = 0; i < Maxn << 2; i++)
798            print(i);
799    }
800
801    void printAll()
802    {
803        for(int i = 0; i < Maxn << 2; i++)
804            print(i);
805    }
806
807    void printAll()
808    {
809        for(int i = 0; i < Maxn << 2; i++)
810            print(i);
811    }
812
813    void printAll()
814    {
815        for(int i = 0; i < Maxn << 2; i++)
816            print(i);
817    }
818
819    void printAll()
820    {
821        for(int i = 0; i < Maxn << 2; i++)
822            print(i);
823    }
824
825    void printAll()
826    {
827        for(int i = 0; i < Maxn << 2; i++)
828            print(i);
829    }
830
831    void printAll()
832    {
833        for(int i = 0; i < Maxn << 2; i++)
834            print(i);
835    }
836
837    void printAll()
838    {
839        for(int i = 0; i < Maxn << 2; i++)
840            print(i);
841    }
842
843    void printAll()
844    {
845        for(int i = 0; i < Maxn << 2; i++)
846            print(i);
847    }
848
849    void printAll()
850    {
851        for(int i = 0; i < Maxn << 2; i++)
852            print(i);
853    }
854
855    void printAll()
856    {
857        for(int i = 0; i < Maxn << 2; i++)
858            print(i);
859    }
860
861    void printAll()
862    {
863        for(int i = 0; i < Maxn << 2; i++)
864            print(i);
865    }
866
867    void printAll()
868    {
869        for(int i = 0; i < Maxn << 2; i++)
870            print(i);
871    }
872
873    void printAll()
874    {
875        for(int i = 0; i < Maxn << 2; i++)
876            print(i);
877    }
878
879    void printAll()
880    {
881        for(int i = 0; i < Maxn << 2; i++)
882            print(i);
883    }
884
885    void printAll()
886    {
887        for(int i = 0; i < Maxn << 2; i++)
888            print(i);
889    }
890
891    void printAll()
892    {
893        for(int i = 0; i < Maxn << 2; i++)
894            print(i);
895    }
896
897    void printAll()
898    {
899        for(int i = 0; i < Maxn << 2; i++)
900            print(i);
901    }
902
903    void printAll()
904    {
905        for(int i = 0; i < Maxn << 2; i++)
906            print(i);
907    }
908
909    void printAll()
910    {
911        for(int i = 0; i < Maxn << 2; i++)
912            print(i);
913    }
914
915    void printAll()
916    {
917        for(int i = 0; i < Maxn << 2; i++)
918            print(i);
919    }
920
921    void printAll()
922    {
923        for(int i = 0; i < Maxn << 2; i++)
924            print(i);
925    }
926
927    void printAll()
928    {
929        for(int i = 0; i < Maxn << 2; i++)
930            print(i);
931    }
932
933    void printAll()
934    {
935        for(int i = 0; i < Maxn << 2; i++)
936            print(i);
937    }
938
939    void printAll()
940    {
941        for(int i = 0; i < Maxn << 2; i++)
942            print(i);
943    }
944
945    void printAll()
946    {
947        for(int i = 0; i < Maxn << 2; i++)
948            print(i);
949    }
950
951    void printAll()
952    {
953        for(int i = 0; i < Maxn << 2; i++)
954            print(i);
955    }
956
957    void printAll()
958    {
959        for(int i = 0; i < Maxn << 2; i++)
960            print(i);
961    }
962
963    void printAll()
964    {
965        for(int i = 0; i < Maxn << 2; i++)
966            print(i);
967    }
968
969    void printAll()
970    {
971        for(int i = 0; i < Maxn << 2; i++)
972            print(i);
973    }
974
975    void printAll()
976    {
977        for(int i = 0; i < Maxn << 2; i++)
978            print(i);
979    }
980
981    void printAll()
982    {
983        for(int i = 0; i < Maxn << 2; i++)
984            print(i);
985    }
986
987    void printAll()
988    {
989        for(int i = 0; i < Maxn << 2; i++)
990            print(i);
991    }
992
993    void printAll()
994    {
995        for(int i = 0; i < Maxn << 2; i++)
996            print(i);
997    }
998
999    void printAll()
1000   {
1001      for(int i = 0; i < Maxn << 2; i++)
1002          print(i);
1003  }

```

```

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         else if(l > M) modify(l, r, tag, M+1,
72                               R, rc) ;
73         else{
74             modify(l, r, tag, L, M, lc) ;
75             modify(l, r, tag, M+1, R, rc) ;
76         }
77         pull(id) ;
78     }
79
80     ll query(int l, int r, int L=1, int R=n,
81               int id=1){
82         if(l <= L && R <= r) return
83                     seg[id].sum ;
84
85         push(id) ;
86         int M = (L + R) >> 1 ;
87         if(r <= M) return query(l, r, L, M,
88                               lc) ;
89         else if(l > M) return query(l, r,
90                                     M+1, R, rc) ;
91         else return query(l, r, L, M, lc) +
92                     query(l, r, M+1, R, rc) ;
93     }
94 }

```

2.2 HLD

```

1  /* HLD */
2  int fa[Maxn], top[Maxn], son[Maxn],
3      sz[Maxn], dep[Maxn] = {0}, dfn[Maxn],
4      rnk[Maxn], dfscnt = 0 ;
5
6  void dfs1(int u, int from){
7      fa[u] = from ;
8      dep[u] = dep[from] + 1 ;
9      sz[u] = 1 ;
10     for ( auto v : g[u] ) if(v != from){
11         dfs1(v, u) ;
12         sz[u] += sz[v] ;
13         if(son[u] == -1 || sz[v] > sz[son[u]])
14             son[u] = v ;
15     }
16     void dfs2(int u, int t){
17         top[u] = t ;

```

```

18 dfn[u] = ++dfscnt ;
19 rnk[dfscnt] = u ;
20
21 if(son[u] == -1) return ;
22
23 dfs2(son[u], t) ;
24
25 for ( auto v : g[u] ) if(v != fa[u] && v
26     != son[u]){
27     dfs2(v, v) ;
28 }
29
30 /* Segment Tree */
31 #define lc (id << 1)
32 #define rc ((id << 1) | 1)
33
34 struct ColorSeg{
35     int left, right, tot ;
36
37     ColorSeg operator+(const ColorSeg &o)
38         const {
39         if(tot == 0) return o ;
40         if(o.tot == 0) return *this ;
41
42         ColorSeg tmp ;
43         tmp.left = left ;
44         tmp.right = o.right ;
45         tmp.tot = tot + o.tot - (right ==
46             o.left) ;
47
48         return tmp ;
49     }
50
51     struct Node{
52         ColorSeg color ;
53         int tag ;
54     }seg[Maxn << 2] ;
55
56 class SegmentTree{
57 private:
58     void pull(int id){
59         // normal pull
60     }
61
62     void AddTag(int id, int tag){
63         // normal AddTag
64     }
65
66     void push(int id){
67         // normal push
68     }
69
70     void modify(int l, int r, int tag, int
71                 L=1, int R=n, int id=1){
72         // normal modify
73     }
74
75     ColorSeg query(int l, int r, int L=1, int
76                    R=n, int id=1){
77         // normal query
78     }
79
80     public:
81     void build(int L=1, int R=n, int id=1){
82         // normal build
83     }
84
85     // update val from u to v (simple path)
86     void update(int u, int v, int val){
87         while(top[u] != top[v]){
88             if(dep[top[u]] < dep[top[v]]) swap(u,
89                 v) ;
90             modify(dfn[top[u]], dfn[u], val) ;
91             u = fa[top[u]] ;
92         }
93
94         if(dep[u] < dep[v]) swap(u, v) ;
95     }
96
97     int get(int u, int v){
98         pair<int, ColorSeg> U, V ;
99         ColorSeg M ;
100        U = {u, {0, 0, 0}} ;
101        V = {v, {0, 0, 0}} ;
102
103        while(top[U.first] != top[V.first]){
104            if(dep[top[U.first]] <
105                dep[top[V.first]]) swap(U, V) ;
106            U.second = query(dfn[top[U.first]],
107                               dfn[U.first]) + U.second ;
108            U.first = fa[top[U.first]] ;
109
110            if(dep[U.first] < dep[V.first]) swap(U,
111                V) ;
112        }
113
114        M = query(dfn[V.first], dfn[U.first]) ;
115
116        return (U.second.tot + V.second.tot +
117                M.tot) - (U.second.left == M.right)
118                - (V.second.left == M.left) ;
119    }
120
121 }tree ;
122
123 void init(){
124     memset(son, -1, sizeof(son)) ;
125 }

```

2.3 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        }

```

```

39
40     int flg = 0 ;
41     for ( auto n : node->end ){
42         if(flg) cout << " " ;
43         else flg = 1 ;
44
45         cout << n ;
46     }
47
48 void clear(TrieNode* node) {
49     if (!node) return ;
50     for (int i = 0; i < 26; i++) {
51         if (node->next[i]) {
52             clear(node->next[i]) ;
53         }
54     }
55     delete node ;
56 }
57
58 ~Trie(){
59     clear(root) ;
60 }
61 };

```

3 String

3.1 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
38            cout << s1 << s2.substr(y) << endl ;
39
40        }
41        return 0 ;
42    }

```

4 Algorithm

4.1 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         if(u == v) return u ;
34
35         for ( int i=MaxLog ; i>=0 ; i-- )
36             if(up[u][i] != up[v][i]){
37                 u = up[u][i] ;
38                 v = up[v][i] ;
39             }
40
41     return up[u][0] ;
42 }
43
44 int main(){
45     int n, q, root ;
46     scanf( "%d%d%d" , &n, &q, &root ) ;
47     MaxLog = __lg(n) ;
48
49     for ( int i=0 ; i<n-1 ; i++ ){
50         int u, v ;
51         scanf( "%d%d" , &u, &v ) ;
52         e[u].push_back(v) ;
53         e[v].push_back(u) ;
54     }
55
56     dfs(root, root, 0) ;
57
58     while(q--){
59         int u, v ;
60         scanf( "%d%d" , &u, &v ) ;
61         printf( "%d\n" , lca(u, v)) ;
62     }
63
64 }
```

4.2 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    vector<Edge> edge ;
11
12    void init(){
13        edge.clear() ;
14        for ( int i=0 ; i<N ; i++ ){
15            par[i] = i ;
16            sz[i] = 1 ;
17        }
18        sum = 0 ;
19    }
20
21    int find(int x){
22        if(x == par[x]) return x ;
23        return par[x] = find(par[x]) ;
24    }
25
26    int merge(int x, int y){
27        x = find(x) ;
28        y = find(y) ;
29
30        if(x == y) return 0 ;
31        if(sz[x] > sz[y]) swap(x, y) ;
32        par[x] = y ;
33        sz[y] += sz[x] ;
34
35        return 1 ;
36    }
37
38    void MST(){
39        int cnt = 0 ;
40        for ( int i=0 ; i<edge.size() && cnt < n-1
41               ; i++ ){
42            auto [u, v, w] = edge[i] ;
43            if(merge(u, v)){
44                cnt++ ;
45                sum -= w ;
46            }
47        }
48    }
49
50    int main(){
51        for ( int i=0 ; i<m ; i++ ){
52            scanf( "%d%d%d" , &u, &v, &w ) ;
53            edge.push_back({u, v, w}) ;
54            sum += w ;
55        }
56
57        sort(edge.begin(), edge.end(),
58              greater<Edge>()) ;
59        MST() ;
60    }
61 }
```

4.3 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    int main(){
11        int cas, n;
12        scanf( "%d" , &cas);
13 }
```

```

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        int main(){
48            build();
49
50            int T = 0 ;
51            while(~scanf( "%d" , &n) && n){
52                int ans = 0 ;
53
54                for ( int i=1 ; i<=n ; i++ ) scanf( "%d" ,
55                                              &stone[i] );
56
57                for ( int i=1 ; i<=n ; i++ ) if(stone[i]
58                                              & 1){
59                    ans ^= SG[n-i] ;
60                }
61            }
62        }
63    }
64 }
```

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 }
```

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