



Notebook - Maratona de Programação

DSUm balão da cor sim cor não

Contents

1 Algoritmos	2	5.5 Path Queries	12
1.1 Busca Binaria	2	5.6 Bellman Ford	13
1.2 Busca Binaria Double	2	5.7 Bfs	13
1.3 Busca Ternaria	2	5.8 Bridges	13
1.4 Delta	2	5.9 Dfs	14
1.5 Fast Exponentiaton	3	5.10 Dfs Tree	14
1.6 Psum	3	5.11 Dijkstra	14
1.7 Psum2d	3	5.12 Euler Path	14
2 DP	3	5.13 Floyd Warshall	15
2.1 Convex Hull Opt	3	5.14 Kosaraju	15
2.2 Digit Dp	4	5.15 Topo Sort	16
2.3 Dp	4	6 Math	16
2.4 Knapsack	5	6.1 Combinatoria	16
2.5 Lis	5	6.2 Dec To Bin	16
2.6 Mochila Iterativa	5	6.3 Divisibilidade	17
2.7 Mochila Recursiva	6	6.4 Divisores	17
3 ED	6	6.5 Fatora	17
3.1 Bitwise	6	6.6 Mdc	17
3.2 Delayed	6	6.7 Mmc	17
3.3 Dsu	6	6.8 Pa	17
3.4 Merge Sort	7	6.9 Pg	17
3.5 Mo	7	6.10 Pollard-rho	17
3.6 Ordered Set	8	6.11 Primos	18
3.7 Segtree	8	7 Strings	18
3.8 Sqrt Decomposition	8	7.1 Suffix Array	18
3.9 Xortrie	9	7.2 Trie	19
4 Geometria	9	7.3 Zfunction	19
4.1 Geometria	9	8 Template	20
5 Grafos	10	8.1 Template	20
5.1 Binary Lifting	10	9 zExtra	20
5.2 Diametro	10	9.1 Formulasmat	20
5.3 Kruskall	10	9.2 Getline	21
5.4 Lca	11		

1 Algoritmos

1.1 Busca Binaria

```
1 #include <bits/stdc++.h>
2 using namespace std;
3
4 bool check(int valor, int x) {
5     return valor <= x;
6 }
7
8 int bb(int a, int b, int x){
9     int l = a;
10    int r = b;
11    while (l < r) {
12        int mid = (l + r) / 2;
13        if (check(mid, x)) r = mid;
14        else l = mid + 1;
15    }
16    return l;
17 }
18
19 bool check(int valor) {
20     return valor <= 10;
21 }
22
23 int bb_menor(int a, int b){
24     int l = a;
25     int r = b;
26     while (l < r) {
27         int mid = (l + r) / 2;
28         if (check(mid)) r = mid;
29         else l = mid + 1;
30     }
31
32     return l;
33 }
34
35
36 int bb_maior(int a, int b){
37     int l = a;
38     int r = b;
39     while (l < r) {
40         int mid = (l + r) / 2;
41         if (!check(mid)) r = mid;
42         else l = mid + 1;
43     }
44 }
```

1.2 Busca Binaria Double

```
1 //
2 // Complexidade : O(NlogN)
3
4 #include <bits/stdc++.h>
5 using namespace std;
6
7 typedef long long ll;
8 typedef long double ld;
9 const ld EPS = 1e-9;
10
11 ll check(ld x, vector<int> &v){
12     ll sum = 0;
13     for(int i=0; i<n; i++){
14         sum += (v[i]/x);
15     }
16     return sum;
17 }
18
19 int main(){
20     int n, k;
21     cin>>n>>k;
```

```
22     vector<int> v(n);
23     for(int i=0; i<n; i++)cin>>v[i];
24
25     ld l=0.0000000, r=10000000.0000000;
26     ld mid;
27     while(r-l>EPS){
28         mid = (ld)((l + r)/2);
29         if (check(mid, v)>=k){
30             l=mid;
31         }
32         else{
33             r = mid;
34         }
35     }
36     cout<<fixed<<setprecision(7)<<mid<<endl;
37
38     return 0;
39 }
```

1.3 Busca Ternaria

```
1 // Uma busca em uma curva, avaliando dois pontos
   diferentes
2 // Complexidade: O(Nlog3N)
3
4 double check(vector<int> v, vector<int> t, double x){
5     double ans = 0;
6     for(int i=0; i<v.size(); i++){
7         ans = max(ans, (double)(abs(v[i]-x) + t[i]));
8     }
9     return ans;
10 }
11
12 int32_t main(){ sws;
13
14     int t; cin>>t;
15     while(t--){
16         int n; cin>>n;
17         vector<int> v(n);
18         vector<int> t(n);
19         input(v);
20         input(t);
21
22         double ans = 0.0;
23         double l=0.0, r=1e9;
24         while(r-l >= EPS){
25
26             double mid1 = (double) l + (r - l) / 3;
27             double mid2 = (double) r - (r - l) / 3;
28
29             double x1 = check(v, t, mid1);
30             double x2 = check(v, t, mid2);
31
32             if(x1 < x2){
33                 r = mid2;
34             }else{
35                 l = mid1;
36                 ans = l;
37             }
38         }
39         cout<<fixed<<setprecision(7);
40         cout<<ans<<endl;
41     }
42     return 0;
43 }
```

1.4 Delta

```
1 #include <bits/stdc++.h>
2 using namespace std;
3
4 int main(){
```

```

5     int n, q;
6     cin >> n >> q;
7     vector<int> v(n,0);
8     vector<int> delta(n+2, 0);
9
10    while(q--){
11        int l, r, x;
12        cin >> l >> r >> x;
13        delta[l] += x;
14        delta[r+1] -= x;
15    }
16
17    int atual = 0;
18    for(int i=0; i < n; i++){
19        atual += delta[i];
20        v[i] = atual;
21    }
22
23    for(int i=0; i < n; i++){
24        cout << v[i] << " ";
25    }
26    cout << endl;
27
28    return 0;
29 }

```

1.5 Fast Exponentiation

```

1 // recursivo
2 int fast_exp(int base, int e, int m){
3     if(!e) return 1;
4     int ans = fast_exp(base * base % m, e/2, m);
5     if(e % 2) return base * ans % m;
6     else return ans;
7 }
8 //iterativo
9 int fast_exp(int base, int e, int m) {
10    int ret = 1;
11    while (e) {
12        if (e & 1) ret = (ret * base) % m;
13        e >>= 1;
14        base = (base * base) % m;
15    }
16    return ret;
17 }

```

1.6 Psum

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 #define input(x) for (auto &it : x) cin >> it
5 typedef long long ll;
6 vector<ll> psum(1e5);
7
8 int solve(int l, int r){
9     if(l==0) return psum[r];
10    else return psum[r] - psum[l-1];
11 }
12
13 int main(){
14
15     int n, q;
16     cin>>n>>q;
17
18     vector<int> v(n);
19     input(v);
20     for(int i=0; i<n; i++){
21         if(i==0) psum[i] = v[i];
22         else psum[i] = psum[i-1] + v[i];
23     }
24     while(q--){

```

```

25         int l, r;
26         cin>>l>>r;
27
28         cout<<(solve(l,r))<<endl;
29     }
30
31     return 0;
32 }

```

1.7 Psum2d

```

1 int psum[MAX][MAX];
2
3 int32_t main(){ sws;
4     int t; cin>>t;
5     while(t--){
6         memset(psum, 0, sizeof(psum));
7         int n, q; cin>>n>>q;
8
9         for(int i=0; i<n; i++){
10            int x, y;
11            cin>>x>>y;
12
13            psum[x][y] += x*y;
14        }
15
16        for(int i=1; i<MAX; i++){
17            for(int j=1; j<MAX; j++){
18                psum[i][j] += psum[i-1][j];
19            }
20
21            for(int i=1; i<MAX; i++){
22                for(int j=1; j<MAX; j++){
23                    psum[i][j] += psum[i][j-1];
24                }
25            }
26
27            for(int i=0; i<q; i++){
28                int x1, y1, x2, y2;
29                cin>>x1>>y1>>x2>>y2;
30                x2--; y2--;
31
32                int soma = psum[x1][y1] + psum[x2][y2] -
33                    psum[x2][y1] - psum[x1][y2];
34                cout<<soma<<endl;
35            }
36        }
37    }
38    return 0;
39 }

```

2 DP

2.1 Convex Hull Opt

```

1 // utiliza-se convexhull tricky geralmente para dp 0(
2 // 2n), onde para cada elemento, percorre os
3 // elementos anteriores à ele.
4 // o objetivo é iterar pelo j e transformar o i em
5 // constante para criar retas e assim, encontrar o
6 // max. ou min.
7 // convex foi feito para achar o max, caso queira o
8 // min. troque o sinal de todos os j's
9 // reta ax + b, onde x é em função de i. Transforma em
10 // um for ós, onde os i's são atribuídas em dp[i]
11 // e soma-se à ela o cht.eval(x da reta)
12 // logo depois, faz cht.insert_line(a da reta, b da
13 // reta)
14
15 // algoritmo
16 const ll is_query = -(1LL<<62);
17 struct line {
18     ll m, b;

```

```

11 mutable function<const line*> succ;
12 bool operator<(const line& rhs) const {
13     if (rhs.b != is_query) return m < rhs.m;
14     const line* s = succ();
15     if (!s) return 0;
16     ll x = rhs.m;
17     return b - s->b < (s->m - m) * x;
18 }
19 };
20
21 struct dynamic_hull : public multiset<line> { // will
    maintain upper hull for maximum
22     const ll inf = LLONG_MAX;
23     bool bad(iterator y) {
24         auto z = next(y);
25         if (y == begin()) {
26             if (z == end()) return 0;
27             return y->m == z->m && y->b <= z->b;
28         }
29         auto x = prev(y);
30         if (z == end()) return y->m == x->m && y->b
<= x->b;
31
32         /* compare two lines by slope, make sure
    denominator is not 0 */
33         ll v1 = (x->b - y->b);
34         if (y->m == x->m) v1 = x->b > y->b ? inf : -
inf;
35         else v1 /= (y->m - x->m);
36         ll v2 = (y->b - z->b);
37         if (z->m == y->m) v2 = y->b > z->b ? inf : -
inf;
38         else v2 /= (z->m - y->m);
39         return v1 >= v2;
40     }
41     void insert_line(ll m, ll b) {
42         auto y = insert({ m, b });
43         y->succ = [=] { return next(y) == end() ? 0 :
&*next(y); };
44         if (bad(y)) { erase(y); return; }
45         while (next(y) != end() && bad(next(y)))
erase(next(y));
46         while (y != begin() && bad(prev(y))) erase(
prev(y));
47     }
48     ll eval(ll x) {
49         auto l = *lower_bound((line) { x, is_query })
;
50         return l.m * x + l.b;
51     }
52 };
53
54 // antes do convex
55 vll dp(n+1, LLINF);
56 for(int i=1; i<=n; i++){
57     ll x, a, b; tie(x, a, b) = v[i];
58     ll ans = LLINF; dp[i] = x*b + a;
59     for(int j=i-1; j>=1; j--){
60         ll x_bef, a_bef, b_bef; tie(x_bef, a_bef,
b_bef) = v[j];
61         ll val = -x_bef * b;
62         ans = min(ans, val + dp[j]);
63     }
64     dp[i] = min(dp[i], ans + x*b + a);
65 }
66
67 return 0;
68 }
69
70 // depois do convex
71 cht.insert_line(0, 0); // primeiro valor (no caso
72

```

dessa questao exemplo eh 0, 0)

```

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

```

2.2 Digit Dp

2.3 Dp

```

1 // DP - Dynamic Programming
2
3 #include <bits/stdc++.h>
4 using namespace std;
5
6 typedef long long ll;
7 const int MAX = 110;
8
9 int n;
10 int tab[MAX];
11 vector<int> v;
12
13 ll dp(int i){
14     if(i>=n) return 0;
15     if(tab[i] != -1) return tab[i];
16
17     int pega = v[i] + dp(i+2);
18     int npega = dp(i+1);
19
20     tab[i] = max(pega, npega);
21     return tab[i];
22 }
23

```

```

24 int main(){
25     memset(tab, -1, sizeof(tab));
26     cin>>n;
27
28     v.assign(n, 0);
29
30     cout<<dp(0)<<endl;
31
32     return 0;
33 }

```

2.4 Knapsack

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 #define int long long
5 #define ll long long
6 #define sws ios::sync_with_stdio(false);cin.tie( NULL
7 );cout.tie(NULL);
8 #define pb(x) push_back(x);
9 #define pii pair<int,int>
10 const int N = 1e3+5;
11
12 int n, t;
13 int tab[N][N];
14 bool pegou[N][N];
15 vector<pair<int,int>> v;
16
17 vector<int> resposta;
18
19 int dp(int idx, int dias){
20     if(idx >= n) return 0;
21     if(tab[idx][dias] != -1) return tab[idx][dias];
22
23     int pega=0;
24     if(dias+v[idx].first <= t){
25         pega = dp(idx+1, dias+v[idx].first)+v[idx].
26         second;
27     }
28
29     int npega = dp(idx+1, dias);
30
31     if(pega>npega) pegou[idx][dias] = true;
32
33     return tab[idx][dias] = max(pega, npega);
34 }
35
36 int32_t main(){
37     memset(tab, -1, sizeof(tab));
38     cin>>n>>t;
39     for(int i=0; i<n; i++){
40         int ti, di;
41         cin>>ti>>di;
42         v.push_back({ti, di});
43     }
44     dp(0, 0);
45     int i = 0, j = 0;
46     vector<int> ans;
47     // retornar os valores
48     while(i < n){
49         if(pegou[i][j]){
50             j += v[i].first;
51             ans.push_back(i+1);
52         }
53         i++;
54     }
55     cout<<ans.size()<<endl;
56     for(int i=0; i<ans.size(); i++){
57         cout<<ans[i]<<" ";
58     }

```

```

59 }

```

2.5 Lis

```

1 // Longest increase sequence
2 // O(nlogn)
3 multiset<int> S;
4 for(int i=0;i<n;i++){
5     auto it = S.upper_bound(vet[i]); // upper -
6     longest strictly increase sequence
7     if(it != S.end())
8         S.erase(it);
9     S.insert(vet[i]);
10 }
11 // size of the lis
12 int ans = S.size();
13
14 // return the elements in LIS
15 // see that later
16 // https://codeforces.com/blog/entry/13225?comment
17 // -180208
18
19 vi LIS(const vi &elements){
20     auto compare = [&](int x, int y) {
21         return elements[x] < elements[y];
22     };
23     set< int, decltype(compare) > S(compare);
24
25     vi previous( elements.size(), -1 );
26     for(int i=0; i<int( elements.size() ); ++i){
27         auto it = S.insert(i).first;
28         if(it != S.begin())
29             previous[i] = *prev(it);
30         if(*it == i and next(it) != S.end())
31             S.erase(next(it));
32     }
33
34     vi answer;
35     answer.push_back( *S.rbegin() );
36     while ( previous[answer.back()] != -1 )
37         answer.push_back( previous[answer.back()] );
38     reverse( answer.begin(), answer.end() );
39     return answer;
40 }

```

2.6 Mochila Iterativa

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 const int maxn = 110, maxp = 1e5+10;
5 const long long inf = 0x3f3f3f3f3f3f3f3f; // ~= 10^18
6
7 int v[maxn], p[maxn];
8 long long dp[maxn][maxp];
9
10 int main() {
11     int n, C; scanf("%d %d", &n, &C);
12     for(int i = 1; i <= n; i++)
13         scanf("%d %d", &p[i], &v[i]);
14
15     long long ans = 0;
16     // inicializando o vetor
17     for(int i = 1; i <= n; i++)
18         for(int P = p[i]; P <= C; P++)
19             dp[i][P] = -inf;
20     // definindo o caso base
21     dp[0][0] = 0;
22
23     for(int i = 1; i <= n; i++) {
24         for(int P = 0; P <= C; P++) {
25             dp[i][P] = dp[i-1][P];
26             if(P >= p[i])

```

```

27         dp[i][P] = max(dp[i][P], dp[i-1][P-p[
28         i]] + v[i]);
29         ans = max(ans, dp[i][P]);
30     }
31 }
32 printf("%lld\n", ans);
33 }

```

2.7 Mochila Recursiva

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 const int maxn = 110, maxp = 1e5+10;
5
6 int v[maxn], p[maxn], n;
7 long long dp[maxn][maxp];
8 bool vis[maxn][maxp];
9
10 long long solve(int i, int P) {
11     if(i == n+1) return 0; // caso base, nao ha mais
12     itens para se considerar
13     if(vis[i][P]) return dp[i][P];
14     vis[i][P] = 1;
15
16     // primeira possibilidade, nao adicionar o
17     elemento
18     dp[i][P] = solve(i+1, P);
19
20     // segunda possibilidade, adicionar o elemento.
21     // Lembrar de tirar o maximo com o valor ja
22     calculado da primeira possibilidade
23     if(P >= p[i])
24         dp[i][P] = max(dp[i][P], solve(i+1, P - p[i])
25         + v[i]);
26
27     return dp[i][P];
28 }
29
30 int main() {
31     int C; scanf("%d %d", &n, &C);
32     for(int i = 1; i <= n; i++)
33         scanf("%d %d", &p[i], &v[i]);
34     printf("%lld\n", solve(1, C));
35 }

```

3 ED

3.1 Bitwise

```

1 // Bitwise Operations
2
3 #include <bits/stdc++.h>
4 using namespace std;
5
6 // Verificar se o bit esta ligado
7 bool isSet(int bitPosition, int number) {
8     bool ret = ((number & (1 << bitPosition)) != 0);
9     return ret;
10 }
11
12 // Ligar o bit
13 bool setBit(int bitPosition, int number) {
14     return (number | (1 << bitPosition));
15 }
16
17 // Gerando todos os subconjuntos de um conjunto em
18 binario
19 void possibleSubsets(char S[], int N) {

```

```

for(int i = 0; i < (1 << N); ++i) { // i = [0, 2^
N - 1]
    for(int j = 0; j < N; ++j)
        if(i & (1 << j)) // se o j-esimo bit de
i esta setado, printamos S[j]
            cout << S[j] << " ";
        cout << endl;
    }
}
// x & (~x+1) -> first set bit

```

3.2 Delayed

```

1 // adiciona elementos em um multiset, e calcula o
2 numero de elementos menor que x no set
3 // O(raiz(QlogQ))
4 class Delayed{
5     ll q;
6     vector<ll> a, delayed;
7 public:
8     void merge(){
9         for(auto x : delayed){
10             a.pb(x);
11         }
12         sort(all(a));
13         delayed = {};
14     }
15
16     void add(ll x){
17         delayed.pb(x);
18         if(delayed.size() * delayed.size() > q){
19             merge();
20         }
21     }
22
23     ll get(ll x){
24         ll ans = 0;
25         ll pos = lower_bound(a.begin(), a.end(), x) -
26         a.begin();
27         if(!pos){ans = 0;} else{ans = pos;}
28         for(auto it: delayed){
29             if(it < x){ans++;}
30         }
31         return ans;
32     }
33
34     Delayed(ll q){
35         this->q = q;
36     };
37 };

```

3.3 Dsu

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 // Complexidade
5 // build : O(N)
6 // find : O(logN)
7 class DSU{
8     vector<int> parent, sz;
9 public:
10     void make(int v){
11         parent[v] = v;
12         sz[v] = 1;
13     }
14
15     int find(int v){
16         if (v == parent[v]) return v;
17         return parent[v] = find(parent[v]);
18     }
19 };

```

```

18     }
19
20     void union_(int a, int b){
21         a = find(a), b = find(b);
22
23         if(sz[b]>sz[a]) swap(a,b);
24         if (a != b){
25             sz[a] += sz[b];
26             parent[b] = a;
27         }
28     }
29
30     bool same(int a, int b){
31         a = find(a), b = find(b);
32         return a == b;
33     }
34
35     DSU(int n): parent(n+1), sz(n+1){
36         for(int i=1; i<=n; i++) make(i);
37     }
38 };
39
40
41 int main(){
42     DSU dsu(10);
43     return 0;
44 }

```

3.4 Merge Sort

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 #define INF 1000000000
5
6 void merge_sort(vector<int> &v){
7     if(v.size()==1)return;
8
9     vector<int> v1, v2;
10
11     for(int i=0; i<v.size()/2; i++) v1.push_back(v[i]);
12     for(int i=v.size()/2; i<v.size(); i++) v2.push_back(v[i]);
13
14     merge_sort(v1);
15     merge_sort(v2);
16
17     v1.push_back(INF);
18     v2.push_back(INF);
19
20     int ini1=0, ini2=0;
21
22     for(int i=0; i<v.size(); i++){
23         if(v1[ini1]<v2[ini2]){
24             v[i] = v1[ini1];
25             ini1++;
26         }else{
27             v[i] = v2[ini2];
28             ini2++;
29         }
30     }
31     return;
32 }

```

3.5 Mo

```

1 // Contar uma certa ocorrencia em queries de L a R
2 // O(K*(N+Q)), onde K = raiz(N)
3
4 // Problema: quantos numeros x existem tal que
5 // x ocorre exatamente x vezes no subarray

```

```

6
7 int block;
8 bool comp(tuple<int,int,int> a, tuple<int,int,int> b)
9 {
10     int l, r, idx, ll, rr, idx2;
11     tie(l, r, idx) = a;
12     tie(ll, rr, idx2) = b;
13
14     if(l/block != ll/block){
15         return l/block < ll/block;
16     }
17     return (l/block & 1) ? r < rr : r > rr;
18 }
19
20 class MO{
21 public:
22     vector<int> a;
23     int ans = 0;
24     unordered_map<int, int> cnt;
25     vector<tuple<int,int,int>> queries;
26
27     void add(int x){
28         if(cnt[x] == x) ans--;
29         cnt[x]++;
30         if(cnt[x] == x) ans++;
31     }
32
33     void del(int x){
34         if(cnt[x] == x) ans--;
35         cnt[x]--;
36         if(cnt[x] == x) ans++;
37     }
38
39     vector<int> get(){
40         vector<int> qans(queries.size());
41         sort(all(queries), comp);
42         int l=0, r=-1;
43         for(auto q: queries){
44             int ll, rr, idx;
45             tie(ll, rr, idx) = q;
46             while(r < rr) add(a[++r]);
47             while(l > ll) add(a[--l]);
48             while(r > rr) del(a[r--]);
49             while(l < ll) del(a[l--]);
50             qans[idx] = ans;
51         }
52         return qans;
53     }
54
55     MO(vector<int> a, vector<tuple<int,int,int>> queries){
56         this->a = a;
57         this->queries = queries;
58         block = (int)sqrt((int)a.size());
59     }
60 };
61
62 int32_t main(){ sws;
63     int n, m;
64     cin>>n>>m;
65     vector<int> a(n);
66     for(int i=0; i<n; i++)cin>>a[i];
67     vector<tuple<int,int,int>> queries;
68     for(int i=0; i<m; i++){
69         int l, r;
70         cin>>l>>r;
71         queries.push_back({l-1, r-1, i});
72     }
73     MO mo(a, queries);
74     vector<int> ans = mo.get();
75     for(int i=0; i<m; i++){
76         cout<<ans[i]<<endl;
77     }
78     return 0;
79 }

```

3.6 Ordered Set

```
1 // disable define int long long
2 #include <ext/pb_ds/assoc_container.hpp>
3 #include <ext/pb_ds/tree_policy.hpp>
4 using namespace __gnu_pbds;
5 template <class T>
6     using ord_set = tree<T, null_type, less<T>,
7         rb_tree_tag,
8         tree_order_statistics_node_update>;
9 // k-th maior elemento - O(logN) - idx em 0
10 s.find_by_order(k)
11
12 // qtd elementos < k - O(logN)
13 s.order_of_key(k)
14
15 ord_set<int> s;
```

3.7 Segtree

```
1 // Build: O(N)
2 // Queries: O(log N)
3 // Update: O(log N)
4
5 // indexada em 0
6
7 class SegTree{
8     int n, elem_neutro = 0;
9     vector<int> tree, lazy, v;
10
11     int merge(int a, int b){
12         return a+b; //seg de soma
13     }
14
15     void build(int l, int r, int no){
16         if(l==r){
17             tree[no] = v[l];
18             return;
19         }
20         int mid = (l+r)/2;
21         build(l, mid, 2*no);
22         build(mid+1, r, 2*no+1);
23
24         tree[no] = merge(tree[2*no], tree[2*no+1]);
25     }
26
27     void update(int A, int B, int x, int l, int r,
28         int no){
29         prop(l, r, no);
30         if(B<l or r<A) return;
31         if(A<=l and r<=B){
32             lazy[no] += x; //update de soma
33             prop(l, r, no);
34             return;
35         }
36         int mid = (l+r)/2;
37
38         update(A, B, x, l, mid, 2*no);
39         update(A, B, x, mid+1, r, 2*no+1);
40
41         tree[no] = merge(tree[2*no], tree[2*no+1]);
42     }
43
44     void prop(int l, int r, int no){
45         if(lazy[no]!=0){
46             tree[no] += (r-l+1)*lazy[no]; //update de
47             soma
48             if(l!=r){
49                 lazy[2*no] += lazy[no]; //update de
50                 soma
51                 lazy[2*no+1] += lazy[no]; //update de
52                 soma

```

```

53         }
54         lazy[no] = 0;
55     }
56 }
57
58 int query(int A, int B, int l, int r, int no){
59     prop(l, r, no);
60     if(B<l or r<A) return elem_neutro;
61     if(A<=l and r<=B) return tree[no];
62     int mid = (l+r)/2;
63
64     return merge(query(A, B, l, mid, 2*no),
65         query(A, B, mid+1, r, 2*no+1));
66 }
67
68 public:
69     SegTree(vector<int> &v){
70         this->n=v.size();
71         this->v=v;
72         tree.assign(4*n, 0);
73         lazy.assign(4*n, 0);
74         build(0, n-1, 1);
75     }
76     int query(int l, int r){return query(l, r, 0,
77         n-1, 1);}
78     void update(int l, int r, int val){update(l,
79         r, val, 0, n-1, 1);}
80     void out(){for(int i=0; i<n; i++){cout<<query
81         (i, i)<<" ";cout<<endl;}}
82 };
83
84 int32_t main(){
85     int n, q;
86     cin>>n>>q;
87     vector<int> v(n);
88     for(int i=0; i<n; i++)cin>>v[i];
89     SegTree seg(v);
90     while(q--){
91         int op; cin>>op;
92         if(op == 1){
93             int l, r, val;
94             cin>>l>>r>>val;
95             l--; r--;
96             seg.update(l, r, val);
97         }else{
98             int idx;
99             cin>>idx;
100             idx--;
101             cout<<seg.query(idx, idx)<<endl;
102         }
103     }
104     return 0;
105 }
```

3.8 Sqrt Decomposition

```
1 // Acha o elemento minimo do segmento de l a r
2 // O(N/K + K), onde K = raiz(N)
3
4 class Sqrt{
5     vector<int> a, b;
6     int n, k;
7
8     public:
9     void build(){
10         b.resize((n/k)+1);
11         for(int i=0; i<=(n/k); i++){
12             b[i] = LLINF;
13         }
14         for(int i=0; i<n; i++){
15             b[i/k] = min(b[i/k], a[i]);
16         }
17     }

```



```

18
19 void update(int idx, int val){
20     a[idx] = val;
21     int blockId = idx/k;
22     b[blockId] = LLINF;
23     for(int i=blockId*k; i<min(blockId+k, n); i
++){
24         b[blockId] = min(b[blockId], a[i]);
25     }
26 }
27
28 int query(int l, int r){
29     int ans = LLINF;
30     int i = l;
31     while(i <= r){
32         if(i%k==0 and i+k-1<=r){
33             ans = min(ans, b[i/k]);
34             i+=k;
35         }else{
36             ans = min(ans, a[i]);
37             i++;
38         }
39     }
40     return ans;
41 }
42
43 Sqrt(vector<int> a){
44     this->a = a;
45     this->n = (int)a.size();
46     this->k = sqrt(n);
47     build();
48 }
49 };

```

3.9 Xortrie

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 const int MAX = (2e5+5)*30;
5
6
7 class Trie{
8     int trie[MAX][2], pref[MAX];
9
10    int node = 1;
11
12    public:
13    void add(int num){
14        int cur = 1;
15        for(int i=30; i>=0; i--){
16            int bit = ((num &(1<<i)) >= 1);
17            if(!pref[trie[cur][bit]]) trie[cur][bit]
= ++node;
18            cur = trie[cur][bit];
19            pref[cur]++;
20        }
21    }
22
23    void erase(int num){
24        int cur = 1;
25        for(int i=30; i>=0; i--){
26            int bit = ((num &(1<<i)) >= 1);
27            cur = trie[cur][bit];
28            pref[cur]--;
29        }
30    }
31
32    int find(int num){
33        int cur = 1;
34        int ans = 0;
35        for(int i=30; i>=0; i--){
36            int bit = ((num &(1<<i)) >= 1);

```

```

37         if(pref[trie[cur][bit^1]]){
38             cur = trie[cur][bit^1];
39             ans += 1<<i;
40         }else
41             cur = trie[cur][bit];
42     }
43     return ans;
44 }
45 };

```

4 Geometria

4.1 Geometria

```

1 const long double EPS = 1e-9;
2 typedef long double ld;
3
4 // point p(x, y);
5 struct point {
6     ld x, y;
7     int id;
8     point(ld x=0, ld y=0): x(x), y(y){}
9
10    point operator+(const point &o) const{ return {x+
o.x, y+o.y}; }
11    point operator-(const point &o) const{ return {x-
o.x, y-o.y}; }
12    point operator*(ld t) const{ return {x*t, y*t}; }
13    point operator/(ld t) const{ return {x/t, y/t}; }
14    ld operator*(const point &o) const{ return x * o.
x + y * o.y; }
15    ld operator^(const point &o) const{ return x * o.
y - y * o.x; }
16 };
17
18 // line l(point(x1, y1), point(x2, y2));
19 struct line{
20     point a, b;
21     line(){}
22     line(point a, point b) : a(a), b(b){}
23 };
24
25 // ponto e em relacao a linha l
26 // counterclockwise
27 int ccw(line l, point e){
28     // -1=dir; 0=colinear; 1=esq;
29     point a = l.b-l.a, b=e-l.a;
30     ld tmp = a ^ b;
31     return (tmp > EPS) - (tmp < -EPS);
32 }
33
34 // se o ponto ta em cima da linha
35 bool isinseg(point p, line l){
36     point a = l.a-p, b = l.b-p;
37     return ccw(l, p) == 0 and (a * b) <= 0;
38 }
39
40 // se o seg de r intersecta o seg de s
41 bool interseg(line r, line s) {
42     if (isinseg(r.a, s) or isinseg(r.b, s)
43         or isinseg(s.a, r) or isinseg(s.b, r)) return
true;
44
45     return (ccw(r, s.a)>0) != (ccw(r, s.b)>0) and
46         (ccw(s, r.a)>0) != (ccw(s, r.b)>0);
47 }
48
49 // area do poligono
50 ld area_polygon(vector<point> vp){
51     ld area = 0;
52     for(int i=1; i<vp.size()-1; i++){
53         area += (vp[0]-vp[i]) ^ (vp[0]-vp[i+1]);

```

```

54     }
55     return (abs(area)/2);
56 }
57
58 // localizacao do ponto no poligono
59 int point_polygon(vector<point> vp, point p){
60     // -1=outside; 0=boundary; 1=inside;
61     int sz = vp.size();
62     int inter = 0;
63     for(int i=0; i<sz; i++){
64         int j = (i+1)%sz;
65         line l(vp[i], vp[j]);
66         if(isinseg(p, l)) return 0;
67
68         if(vp[i].x <= p.x and p.x < vp[j].x and ccw(l
, p) == 1) inter++;
69         else if(vp[j].x <= p.x and p.x < vp[i].x and
ccw(l, p) == -1) inter++;
70     }
71
72     if(inter%2==0) return -1;
73     else return 1;
74 }

```

5 Grafos

5.1 Binary Lifting

```

1 vector<int> adj[MAX];
2 const int LOG = 30;
3 int up[MAX][LOG], parent[MAX];
4
5 void process(int n){
6     for(int v=1; v<=n; v++){
7         up[v][0] = parent[v];
8         for(int i=1; i<LOG; i++){
9             up[v][i] = up[ up[v][i-1] ][i-1];
10        }
11    }
12 }
13
14 int jump(int n, int k){
15     for(int i=0; i<LOG; i++){
16         if(k & (1 << i)){
17             n = up[n][i];
18         }
19     }
20     if(n == 0) return -1;
21     return n;
22 }
23
24 int32_t main(){
25
26     int n, q; cin>>n>>q;
27
28     parent[1] = 0;
29     for(int i=1; i<=n-1; i++){
30         int x;
31         cin>>x;
32         parent[i+1] = x;
33
34         adj[i+1].pb(x);
35         adj[x].pb(i+1);
36     }
37     process(n);
38     for(int i=0; i<q; i++){
39         int a, b;
40         cin>>a>>b;
41
42         cout<<(jump(a,b))<<endl;
43     }
44 }

```

5.2 Diametro

```

1 // Acha o caminho mais longo de uma ponta ate outra
   ponta de uma arvore
2 // Complexidade: O(N)
3 // Lembrar de checar N == 1, diametro = 0
4 #include <bits/stdc++.h>
5 using namespace std;
6 const int MAX = 1e5+10;
7
8 vector<int> adj[MAX];
9 /*pair<int, int> bfs(int s, int N){
10     vi dist(N + 1, MAX); dist[s] = 0;
11     queue<int> q; q.push(s);
12     int last = s;
13
14     while(!q.empty()){
15         auto u = q.front(); q.pop();
16         last = u;
17         for(auto v: adj[u]){
18             if(dist[v]==MAX){
19                 dist[v]=dist[u]+1;
20                 q.push(v);
21             }
22         }
23     }
24     return {last, dist[last]};
25 }
26
27 int diameter_bfs(int N){
28     auto [v, _] = bfs(1, N);
29     auto [w, D] = bfs(v, N);
30
31     return D;
32 }*/
33
34 void dfs(int u, int p, vector<int> &dist){
35     for(auto v : adj[u]){
36         if(v == p) continue;
37         dist[v] = dist[u] + 1;
38         dfs(v, u, dist);
39     }
40 }
41
42 int diameter(int n){
43     vector<int> dist(n+1);
44     dfs(1, 0, dist);
45     // get farthest node from root
46     auto v = (int)(max_element(dist.begin(), dist.end()
) - dist.begin());
47     // start from farthest node
48     dist[v] = 0;
49     dfs(v, 0, dist);
50     return *max_element(dist.begin(), dist.end());
51 }
52
53 int32_t main(){ sws;
54     int n; cin>>n;
55     for(int i=0; i<n-1; i++){
56         int a, b;
57         cin>>a>>b;
58         adj[a].pb(b);
59         adj[b].pb(a);
60     }
61     if(n == 1) cout<<0<<endl;
62     else cout<<diameter(n)<<endl;
63     return 0;
64 }

```

5.3 Kruskall

```

1 // Arvore geradora minima (arvore conexa com peso
   minimo)

```

```

2 // O(MlogN)
3
4 #include <bits/stdc++.h>
5 using namespace std;
6
7 int n;
8 class DSU{
9     vector<int> parent, sz;
10 public:
11     void make(int v){
12         parent[v] = v;
13         sz[v] = 1;
14     }
15
16     int find(int v){
17         if (v == parent[v]) return v;
18         return parent[v] = find(parent[v]);
19     }
20
21     void union_(int a, int b){
22         a = find(a), b = find(b);
23
24         if(sz[b]>sz[a]) swap(a,b);
25         if (a != b){
26             sz[a] += sz[b];
27             parent[b] = a;
28         }
29     }
30
31     bool same(int a, int b){
32         a = find(a), b = find(b);
33         return a == b;
34     }
35
36     DSU(int n): parent(n+1), sz(n+1){
37         for(int i=1; i<=n; i++) make(i);
38     }
39 };
40
41 // {a, b, weight}
42 vector<tuple<int,int,int>> MST(vector<tuple<int,int,
43     int>> &v){
44     DSU dsu(n);
45     sort(v.begin(), v.end());
46     vector<tuple<int,int,int>> ans;
47     for(int i=0; i<v.size(); i++){
48         int w, a, b;
49         tie(w, a, b) = v[i];
50         if(!dsu.same(a, b)){
51             dsu.union_(a, b);
52             ans.push_back({a, b, w});
53         }
54     }
55     return ans;
56 }
57
58 int32_t main(){
59     int m;
60     cin>>n>>m;
61     DSU dsu(n);
62     vector<tuple<int,int,int>> vt;
63     for(int i=0; i<m; i++){
64         int a, b, w;
65         cin>>a>>b>>w;
66         // {weight, a, b}
67         vt.push_back({w, a, b});
68     }
69     vector<tuple<int,int,int>> ans = MST(vt);
70     return 0;
71 }

```

5.4 Lca

```

1 /*
2 Lowest Common ancestor (LCA) - dado uma Arvore cuja
3 raiz eh um vertice arbitrario e dois vertices u,v
4 que a pertencem, diga qual eh o no mais baixo(
5 relativo a raiz) que eh ancestral de u,v.
6
7 */
8 // Complexidades:
9 // build - O(n log(n))
10 // lca - O(log(n))
11
12 #include <bits/stdc++.h>
13 using namespace std;
14 #define ll long long
15 const int SIZE = 2e5+5;
16 const int LOG = 30; // log2(SIZE)+1;
17 int depth[SIZE];
18 vector<pair<int,int>> adj[SIZE];
19 int up[SIZE][LOG];
20
21 void dfs(int u, int p) {
22     for(auto edge : adj[u]) {
23         int v, w;
24         tie(v, w) = edge;
25         if(v != p){
26             up[v][0] = u;
27             //weight[v] = weight[u] + w;
28             depth[v] = depth[u] + 1;
29             for(int i=1; i<LOG; i++){
30                 up[v][i] = up[ up[v][i-1] ][i-1];
31             }
32             dfs(v, u);
33         }
34     }
35 }
36
37 int lca(int a, int b) {
38     if(depth[a] < depth[b]) swap(a,b);
39     int k = depth[a] - depth[b];
40     for(int i=0; i<LOG; i++){
41         if(k & (1 << i)){
42             a = up[a][i];
43         }
44     }
45     if(a == b) return a;
46     for (int i = LOG-1; i >= 0; i--) {
47         if(up[a][i] != up[b][i]) {
48             a = up[a][i];
49             b = up[b][i];
50         }
51     }
52     return up[a][0];
53 }
54
55 ll dist(int u, int v){
56     return depth[u] + depth[v] - 2*depth[lca(u,v)];
57     // return weight[u] + weight[v] -2*weight[lca(u,v)];
58 }
59
60 int main() {
61     int n; cin>>n;
62
63     for(int i=0; i<n-1; i++){
64         int x, y, z;
65         cin>>x>>y>>z;
66         adj[x].push_back({y, z});
67         adj[y].push_back({x, z});
68     }
69     // raiz
70     dfs(1, 0);
71
72     int q; cin>>q;
73     while(q--){

```

```

70     int a, b, c;
71     cin>>a>>b>>c;
72     long long x = dist(a, b) + dist(b, c);
73     cout<<x<<endl;
74 }
75 }

```

5.5 Path Queries

```

1  #include <bits/stdc++.h>
2  using namespace std;
3
4  #define ll long long
5  #define sws ios::sync_with_stdio(false);cin.tie( NULL
6  );cout.tie(NULL);
7  #define loop(i,a,n) for(int i=a; i < n; i++)
8  #define pb(x) push_back(x);
9  #define vi vector<int>
10 #define mp(x,y) make_pair(x,y)
11 #define print(x,y) loop(i,0,y){cout << x[i] << " ";}
12     cout << "\n";
13 #define pii pair<int,int>
14
15 const int N = 2e5+10;
16 const ll MOD = 1e9+7;
17 const int INF = 0x3f3f3f3f;
18 const ll LLINF = 0x3f3f3f3f3f3f3f3f;
19 const ll MAX = 2e5+10;
20
21 ll n, T = 0, a[MAX], path[MAX], valor[MAX], in[MAX],
22     out[MAX], preorder[MAX];
23
24 vector<ll> tree[MAX];
25
26 class SegTree{
27 private:
28     ll st[4*MAX], lazy[4*MAX];
29
30     ll merge(ll a, ll b){
31         return max(a,b);
32     }
33
34     void push(int i, long long x = 0){
35         st[i] += (lazy[i]+x);
36         if(2*i < 4*MAX) lazy[2*i] += (lazy[i]+x);
37         if(2*i+1 < 4*MAX) lazy[2*i+1] += (lazy[i]
38         ]+x);
39         lazy[i] = 0;
40     }
41
42 public:
43     void build(int i = 1, int l = 0, int r = n-1)
44     {
45         if(l == r){
46             st[i] = a[l]; //leaf node.
47             lazy[i] = 0;
48         }
49         else{
50             int mid = (r+l)/2;
51             lazy[i] = 0;
52             build(2*i, l, mid);
53             build(2*i + 1, mid+1, r);
54             st[i] = merge(st[2*i], st[2*i + 1]);
55             //parent node.
56         }
57         return;
58     }
59
60     ll query(int l, int r, int i = 1, int auxl =
61     0, int auxr = n-1){
62         if(l <= auxl && r >= auxr){ //total
63             overlap.
64             if(lazy[i]){

```

```

57         push(i);
58     }
59     return st[i];
60 }
61 else if(auxr < l || auxl > r){ //no
62     overlap.
63     return 0;
64 }
65 else{ //partial overlap
66     int auxmid = (auxr+auxl)/2;
67     push(i);
68     return merge(query(l, r, 2*i, auxl,
69     auxmid), query(l, r, 2*i + 1, auxmid+1, auxr));
70 }
71
72 void update(int l, int r, ll x, int i = 1,
73 int auxl = 0, int auxr = n-1){
74     if(l <= auxl && auxr <= r){ //total
75     overlap.
76         push(i,x);
77     }
78     else if(auxr < l || auxl > r){ //no
79     overlap.
80         return;
81     }
82     else{ //partial overlap
83         int auxmid = (auxr+auxl)/2;
84         update(l, r, x, 2*i, auxl, auxmid);
85         update(l, r, x, 2*i + 1, auxmid+1,
86         auxr);
87         st[i] = merge(st[2*i],st[2*i+1]);
88     }
89 }
90
91 void dfs(int u, int p){
92     path[u] += (path[p] + valor[u]);
93     preorder[++T] = u; //preorder
94     a[T-1] = path[u];
95     in[u] = T;
96     for(auto v: tree[u]){
97         if(v != p){
98             dfs(v,u);
99         }
100     }
101     out[u] = T;
102 }
103
104 int main(){
105     sws;
106     int q; cin >> n >> q;
107     SegTree seg;
108     loop(i,1,n+1){
109         int a; cin >> a;
110         valor[i] = a;
111     }
112     loop(i,0,n-1){
113         int u, v; cin >> u >> v;
114         tree[u].pb(v);
115         tree[v].pb(u);
116     }
117     dfs(1,0);
118     seg.build();
119     while(q--){
120         int op; cin >> op;
121         if(op == 1){
122             ll u, x; cin >> u >> x;
123             seg.update(in[u]-1,out[u]-1,x-valor[u]);
124             valor[u] = x;
125         }
126         else{
127             int u; cin >> u;

```

```

124         cout << seg.query(in[u]-1,in[u]-1) << "\n";
125     }
126 }
127 }

```

5.6 Bellman Ford

```

1 /*
2 Algoritmo de busca de caminho minimo em um digrafo (
3 grafo orientado ou dirigido) ponderado, ou seja,
4 cujas arestas tem peso, inclusive negativo.
5 */
6
7 int d[MAX];
8 int parent[MAX];
9 vector<pair<int,int>> adj[MAX];
10
11 int32_t main(){ sws;
12     int n, m;
13     cin>>n>>m;
14     for(int i=1; i<=n; i++){
15         d[i] = LLINF;
16     }
17     for(int i=0; i<m; i++){
18         int a, b, c;
19         cin>>a>>b>>c;
20         adj[a].pb({b,c});
21     }
22     d[1] = 0;
23
24     int src_cycle = -1;
25     for(int j=1; j<=n and src_cycle; j++){
26         src_cycle = 0;
27         for(int u=1; u <= n; u++){
28             for(auto [v, w]: adj[u]){
29                 if(d[u] + w < d[v]){
30                     d[v] = d[u] + w;
31                     parent[v] = u;
32                     src_cycle = v;
33                 }
34             }
35         }
36     }
37     // there is no negative cycle
38     if(!src_cycle){cout<<"NO"<<endl;}
39     else {
40         // there is negative cycle
41         cout<<"YES"<<endl;
42         vector<int> v;
43         int a = src_cycle;
44         for(int i = 0; i < n; i++)
45             src_cycle = parent[src_cycle];
46
47         int atual=src_cycle;
48         while(true){
49             v.pb(atual);
50             if(atual == src_cycle && v.size()>1)break;
51             atual = parent[atual];
52         }
53         reverse(all(v));
54         print(v, (int)v.size());
55     }
56
57     return 0;
58 }

```

5.7 Bfs

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 //-----
5 #define MAXN 50050
6
7 int n, m;
8 bool visited[MAXN];
9 vector<int> lista[MAXN];
10 //-----
11
12 void bfs(int x){
13
14     queue<int> q;
15     q.push(x);
16     while(!q.empty()){
17         int v = q.front();
18         q.pop();
19         visited[v] = true;
20         for(auto i : lista[v]){
21             if(!visited[i]){
22                 q.push(i);
23             }
24         }
25     }
26 }

```

5.8 Bridges

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 #define endl "\n"
5 #define sws std::ios::sync_with_stdio(false); cin.tie
6 (NULL); cout.tie(NULL);
7 #define pb push_back
8 const int MAX = 1e5+5;
9 vector<int> adj[MAX];
10 int timer=0;
11 int low[MAX], tin[MAX];
12 bool bridge=false;
13 bool visited[MAX];
14
15 void dfs(int v, int p = -1) {
16     visited[v] = true;
17     tin[v] = low[v] = timer++;
18     for (int to : adj[v]) {
19         if (to == p) continue;
20         if (visited[to]) {
21             low[v] = min(low[v], tin[to]);
22         } else {
23             dfs(to, v);
24             low[v] = min(low[v], low[to]);
25             if (low[to] > tin[v]){
26                 //IS_BRIDGE(v, to);
27             }
28         }
29     }
30 }
31
32 int32_t main(){ sws;
33     int n, m;
34     cin>>n>>m;
35
36     for(int i=0; i<m; i++){
37         int a, b;
38         cin>>a>>b;
39
40         adj[a].pb(b);
41         adj[b].pb(a);
42     }
43     for(int i=1; i<=n; i++){
44         if(!visited[i]) dfs(i);
45     }

```

```

44     }
45     if(bridge) cout<<"YES"<<endl;
46     else cout<<"NO"<<endl;
47
48     return 0;
49 }

```

5.9 Dfs

```

1  #include <iostream>
2  #include <vector>
3  #include <stack>
4
5  using namespace std;
6
7  //-----
8  #define MAXN 50050
9
10 int n, m;
11 bool visited[MAXN];
12 vector<int> lista[MAXN];
13 //-----
14
15 void dfs(int x){
16     visited[x] = true;
17     for(auto i : lista[x]){
18         if(!visited[i]){
19             dfs(i);
20         }
21     }
22 }
23
24 void dfsStack(int x){
25     stack<int> s;
26     s.push(x);
27     while(!s.empty()){
28         int v = s.top();
29         s.pop();
30         visited[v] = true;
31         for(auto i : lista[v]){
32             if(!visited[i]){
33                 s.push(i);
34             }
35         }
36     }
37 }

```

5.10 Dfs Tree

```

1  const int MAX = 1e5;
2  int desce[MAX], sobe[MAX], vis[MAX], h[MAX];
3  int backedges[MAX], pai[MAX];
4
5  // backedges[u] = backedges que comecam embaixo de (
   // ou =) u e sobem pra cima de u; backedges[u] == 0
   // => u eh ponte
6  void dfs(int u, int p) {
7      if(vis[u]) return;
8      pai[u] = p;
9      h[u] = h[p]+1;
10     vis[u] = 1;
11
12     for(auto v : g[u]) {
13         if(p == v or vis[v]) continue;
14         dfs(v, u);
15         backedges[u] += backedges[v];
16     }
17     for(auto v : g[u]) {
18         if(h[v] > h[u]+1)
19             desce[u]++;
20         else if(h[v] < h[u]-1)
21             sobe[u]++;

```

```

22     }
23     backedges[u] += sobe[u] - desce[u];
24 }

```

5.11 Dijkstra

```

1  // Acha o menor caminho de um ponto inicial para
   // todos os outros
2  // Complexidade: O(|V|+|E|*log|V|)
3
4  #include <bits/stdc++.h>
5  using namespace std;
6  #define ll long long
7  typedef pair<int,int> pii;
8
9  const int N = 100005;
10 const ll oo = 1e18;
11
12 ll d[N]; // vetor onde guardamos as distancias
13
14 int n; // numeros de vertices
15 vector<pair<int, ll>> adj[N];
16
17 void dijkstra(int start){
18     for(int u = 1; u <= n; u++){
19         d[u] = oo;
20
21         priority_queue<pii,
22             vector<pii>,
23             greater<pii> > pq;
24
25         d[start] = 0;
26         pq.push({d[start], start});
27
28         ll dt, w;
29         int u, v;
30         while(!pq.empty()){
31             auto [dt, u] = pq.top(); pq.pop();
32             if(dt > d[u]) continue;
33             for(auto [v, w] : adj[u]){
34                 if(d[v] > d[u] + w){
35                     d[v] = d[u] + w;
36                     pq.push({d[v], v});
37                 }
38             }
39         }
40     }
41
42     int main(){
43
44         // le o input, qnt de vertices, arestas
45         // e vertice inicial(start)
46         int start = 0; // inicial
47         dijkstra(start);
48
49         for(int u = 1; u <= n; u++){
50             printf("Distancia de %d para %d: %lld\n",
51                 start, u, d[u]);
52         }
53 }

```

5.12 Euler Path

```

1  // Acha um caminho em que visita todas as arestas
   // somente uma vez
2
3  class EulerPath{
4      int n, m, id=0;
5      bool impossible=false, directed;
6      vector<int> in, out, deg;
7      vector<pair<int,int>> adj[MAX], path;

```

```

8     vector<bool> visited;
9     int src = -1;
10    public:
11    void add(int a, int b){
12        if(directed){
13            adj[a].pb({b, id});
14            out[a]++, in[b]++;
15        }else{
16            adj[a].pb({b, id}), adj[b].pb({a, id});
17            deg[a]++, deg[b]++;
18        }
19        id++;
20    }
21
22    void dfs(int p, int u){
23        while(!adj[u].empty()){
24            pair<int, int> p = adj[u].back(); adj[u].
25            pop_back();
26            int v, id; tie(v, id) = p;
27            if(visited[id]) continue;
28            visited[id] = true;
29            dfs(u, v);
30        }
31        if(path.size() and path.back().first != u)
32            impossible=true;
33        path.pb({p, u});
34    }
35    // exists, path
36    vector<int> findEulerPath(){
37        for(int i=1; i<=n; i++) if(deg[i]%2 != 0)
38            return {};
39        dfs(-1, src);
40        if((path.size() != m+1) or impossible) return
41        {};
42        vector<int> ans;
43        reverse(all(path));
44        for(int i=0; i<path.size(); i++){
45            ans.pb(path[i].second);
46        }
47        return ans;
48    }
49
50    EulerPath(int _n, int _m, bool _directed, int
51    _src=-1):
52    in(n+1), out(n+1), deg(n+1), visited(m, 0),
53    n(_n), m(_m), directed(_directed), src(_src){}
54 };
55
56 int32_t main(){ sws;
57     int n, m;
58     cin>>n>>m;
59     EulerPath ep(n, m, true, 1);
60     for(int i=0; i<m; i++){
61         int a, b;
62         cin>>a>>b;
63         ep.add(a, b);
64     }
65     vector<int> ans = ep.findEulerPath();
66     if(ans.size()){
67         print(ans, ans.size());
68     }else{
69         cout<<"IMPOSSIBLE"<<endl;
70     }
71     return 0;
72 }

```

5.13 Floyd Warshall

```

1 /*
2 Algoritmo de caminho mais curto com todos os pares.
3 Complexidade:  $O(3N)$ 

```

```

4 */
5
6 #include <bits/stdc++.h>
7 using namespace std;
8
9 const int oo = 1000000000; // infinito
10
11 int main(){
12     int n, m; cin>>n>>m;
13
14     vector<vector<int>>> dist(n+1, vector<int> (n+1));
15
16     for(int i=0; i<n+1; i++){
17         for(int j=0; j<n+1; j++){
18             dist[i][j] = oo;
19         }
20     }
21
22     for(int i=0; i<n +1; i++){
23         dist[i][i]=0;
24     }
25
26     for(int i=0; i<m; i++){
27         int comeca, termina, custo;
28         cin>>comeca>>termina>>custo;
29
30         // grafo direcionado
31         dist[comeca][termina] = custo;
32     }
33
34     for(int k=1; k<=n; k++){ // intermediario
35         for(int i=1; i<=n; i++){
36             for(int j=1; j<=n; j++){
37                 //(i,k,j) = ir de i pra j passando
38                 por k;
39
40                 // relaxar distancia de i pra j
41                 dist[i][j] = min(dist[i][j], dist[i][
42                 k] + dist[k][j]);
43             }
44         }
45         return 0;
46     }

```

5.14 Kosaraju

```

1 // Acha componentes fortemente conexas
2 // ou seja, que tem caminho entre todos os pares de
3 // vertices
4 // 0(n+m)
5 // SCC from BenQ
6 class SCC{
7     int N;
8     public:
9     vector<int> adj[MAX], radj[MAX];
10    stack<int> st;
11    vector<bool> visited;
12    // todas as componentes
13    vector<int> comps;
14    // componente do vertice
15    vector<int> comp;
16
17    void add(int x, int y) {
18        adj[x].pb(y), radj[y].pb(x);
19    }
20    void dfs(int u){
21        visited[u] = true;
22        for(auto v: adj[u]) if(!visited[v]) dfs(v);
23        st.push(u);
24    }

```

```

25 void dfs2(int u, int c){
26     comp[u] = c;
27     for(auto v: radj[u]) if(comp[v] == -1) dfs2(v
28 , c);
29 }
30 void gen() {
31     for(int i=1; i<=N; i++) if(!visited[i]) dfs(i
32 );
33     while(!st.empty()){
34         int u = st.top(); st.pop();
35         if(comp[u] == -1){
36             dfs2(u, u);
37             comps.pb(u);
38         }
39     }
40 }
41 SCC(int n){
42     N = n+1;
43     comp.assign(N, -1);
44     visited.assign(N, false);
45 }
46 };
47
48 int32_t main(){ sws;
49     int n, m;
50     cin>>n>>m;
51     SCC scc(n);
52     for(int i=0; i<m; i++){
53         int a, b;
54         cin>>a>>b;
55         scc.add(a, b);
56     }
57     int comp=0;
58     vector<int> ans(n+1);
59     scc.gen();
60     cout<<scc.comps.size()<<endl;
61     for(int i=1; i<=n; i++){
62         if(!ans[scc.comp[i]]){
63             ans[scc.comp[i]] = ++comp;
64         }
65     }
66     for(int i=1; i<=n; i++){
67         cout<<ans[scc.comp[i]]<<" ";
68     }
69     cout<<endl;
70     return 0;
71 }

```

5.15 Topo Sort

```

1 // topological sort
2 // retorna uma ordenacao topologica
3 // caso for um dag, se nao, retorna vazio se tiver
4 // ciclo
5 // 0(n+m)
6 // indexado em 1 os vertices
7
8 int n;
9 int visited[MAX];
10 vector<int> adj[MAX];
11 int pos=0;
12 vector<int> ord;
13 bool has_cycle=false;
14
15 void dfs(int v){
16     visited[v] = 1;
17     for(auto u : adj[v]){
18         if(visited[u] == 1) has_cycle=true;
19         else if(!visited[u]) dfs(u);
20     }
21     ord[pos++] = v;
22     visited[v] = 2;
23 }

```

```

24 vector<int> topo_sort(int n){
25     ord.assign(n, 0);
26     has_cycle = false;
27     pos = n-1;
28     for(int i=1; i<=n; i++){
29         if(!visited[i]) dfs(i);
30     }
31
32     if(has_cycle) return {};
33     else return ord;
34 }
35
36 int main(){
37     int m;
38     cin>>n>>m;
39
40     for(int i=0; i<m; i++){
41         int a, b;
42         cin>>a>>b;
43         adj[a].pb(b);
44     }
45
46     vector<int> ans = topo_sort(n);
47
48     return 0;
49 }

```

6 Math

6.1 Combinatoria

```

1 // quantidade de combinacoes possiveis sem repeticao
2 // de 2 numeros
3 int comb(int k){
4     if(k==1 or k==0) return 0;
5     return (k*(k-1))/2;
6 }
7
8 int fat[MAX], ifat[MAX];
9
10 void factorial(){
11     fat[0] = 1;
12     for(int i=0; i<MAX; i++){
13         if(i > 0) fat[i] = (i * fat[i-1]) % MOD;
14         ifat[i] = fast_exp(fat[i], MOD-2, MOD);
15     }
16 }
17 // N escolhe K
18 int choose(int n, int k){
19     if(k > n or k<0) return 0;
20     return (((fat[n] * ifat[k]) % MOD) * ifat[n-k]) %
21     MOD;
22 }

```

6.2 Dec To Bin

```

1 int binary_to_decimal(long long n) {
2     int dec = 0, i = 0, rem;
3
4     while (n!=0) {
5         rem = n % 10;
6         n /= 10;
7         dec += rem * pow(2, i);
8         ++i;
9     }
10
11     return dec;
12 }
13

```



```

14 long long decimal_to_binary(int n) {
15     long long bin = 0;
16     int rem, i = 1;
17
18     while (n!=0) {
19         rem = n % 2;
20         n /= 2;
21         bin += rem * i;
22         i *= 10;
23     }
24
25     return bin;
26 }
27
28 // copieei da nathalia, tem que ver se funciona
    certinho

```

6.3 Divisibilidade

```

1 // 2 -> se eh par
2 // 3 -> se a soma dos algarismos eh divisivel por 3
3 // 4 -> se os dois ultimos algarismos eh divisivel
    por 4
4 // 5 -> se o ultima algarismo eh 0 ou 5
5 // 6 -> se eh par e a soma dos algarismos eh
    divisivel por 3
6 // 7 -> se o dobro do ultimo algarismo subtraido do
    numero sem o ultimo algarismo eh divisivel por 7
7 // 8 -> se os 3 ultimos algarismos eh divisivel por 8
8 // 9 -> se a soma dos algarismos eh divisivel por 9
9 // 10 -> se o ultimo algarimo eh 0

```

6.4 Divisores

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 vector<long long> get_divisors(long long n){
5     vector<long long> divs;
6     for(long long i = 1; i*i <= n; i++){
7         if(n%i == 0){
8             divs.push_back(i);
9             long long j = n/i;
10            if(j != i)
11                divs.push_back(j);
12        }
13    }
14    return divs;
15 }

```

6.5 Fatora

```

1 map<int,int> fatora(int n){
2     map<int,int> fact;
3     for(int i = 2; i*i <= n; i++){
4         while(n%i == 0){
5             fact[i]++;
6             n /= i;
7         }
8     }
9     if(n > 1)
10         fact[n]++;
11     return fact;
12 }

```

6.6 Mdc

```

1 // Greatest common divisor / MDC
2
3 long long gcd(long long a, long long b){
4     return b ? gcd(b, a % b) : a;

```

```

5 }
6
7 // or just use __gcd(a,b)

```

6.7 Mmc

```

1 // Least Common Multiple - MMC
2 #include <bits/stdc++.h>
3 using namespace std;
4
5 long long lcm(long long a, long long b){
6     return (a/__gcd(a,b)*b);
7 }

```

6.8 Pa

```

1 // Termo Geral
2 // An = A1 + (n-1)*d
3
4 // Soma
5 // Sn = (n/2)(2*A1+(n-1)*d)
6
7 // óSomatrio de 1 a K
8 int pa(int k){
9     return (k*(k+1))/2;
10 }

```

6.9 Pg

```

1 // Termo Geral
2 // An = A1 * r^(n-1)
3
4 // Soma
5 // (A(r^(n)-1))/(r-1)

```

6.10 Pollard-rho

```

1 // O(sqrt(N) * logN)
2
3 ll a[MAX];
4
5 ll mul(ll a, ll b, ll m){
6     ll ret = a*b - (ll)((long double)1/m*a*b+0.5)*m;
7     return ret < 0 ? ret+m : ret;
8 }
9
10 ll pow(ll x, ll y, ll m) {
11     if (!y) return 1;
12     ll ans = pow(mul(x, x, m), y/2, m);
13     return y%2 ? mul(x, ans, m) : ans;
14 }
15
16 bool prime(ll n) {
17     if (n < 2) return 0;
18     if (n <= 3) return 1;
19     if (n % 2 == 0) return 0;
20
21     ll r = __builtin_ctzll(n - 1), d = n >> r;
22     for (int a : {2, 325, 9375, 28178, 450775,
23         9780504, 795265022}) {
24         ll x = pow(a, d, n);
25         if (x == 1 or x == n - 1 or a % n == 0)
26             continue;
27
28         for (int j = 0; j < r - 1; j++) {
29             x = mul(x, x, n);
30             if (x == n - 1) break;
31         }
32         if (x != n - 1) return 0;
33     }
34     return 1;

```

```

33 }
34
35 ll rho(ll n) {
36     if (n == 1 or prime(n)) return n;
37     auto f = [n](ll x) {return mul(x, x, n) + 1;};
38
39     ll x = 0, y = 0, t = 30, prd = 2, x0 = 1, q;
40     while (t % 40 != 0 or __gcd(prd, n) == 1) {
41         if (x==y) x = ++x0, y = f(x);
42         q = mul(prd, abs(x-y), n);
43         if (q != 0) prd = q;
44         x = f(x), y = f(f(y)), t++;
45     }
46     return __gcd(prd, n);
47 }
48
49 vector<ll> fact(ll n) {
50     if (n == 1) return {};
51     if (prime(n)) return {n};
52     ll d = rho(n);
53     vector<ll> l = fact(d), r = fact(n / d);
54     l.insert(l.end(), r.begin(), r.end());
55     return l;
56 }
57
58 int main(){
59     set<ll> primes;
60     int M, N, K; cin >> M >> N >> K;
61     loop(i,0,N){
62         cin >> a[i];
63         vector<ll> aprimes = fact(a[i]);
64         for(auto prime : aprimes){
65             primes.insert(prime);
66         }
67     }
68     int m, n, d;
69     loop(i,0,K) cin >> m >> n >> d;
70     for(auto prime : primes){
71         cout << prime << " ";
72     }
73 }
74 }

```

6.11 Primos

```

1 // PRIMALIDADE
2
3 #include <bits/stdc++.h>
4 using namespace std;
5
6 const int MAX = 1e5+7;
7
8 void crivo(){
9     vector<int> crivo(MAX, 1);
10    for(int i=2; i*i<=MAX; i++){
11        if(crivo[i]==1){
12            for(int j=i+i; j<MAX; j+=i){
13                crivo[j]=0;
14            }
15        }
16    }
17 }
18 bool isPrime[MAX];
19 vector<int> generate_primes() {
20     vector<int> primes;
21     isPrime[1]=isPrime[0]=1;
22     for(int i=2; i<MAX; i++){
23         if(!isPrime[i]){
24             primes.pb(i);
25             for(int j=i+i; j<MAX; j+=i){
26                 isPrime[j]=1;
27             }
28         }
29     }
30 }

```

```

29     }
30     return primes;
31 }
32
33 bool is_prime(int num){
34     for(int i = 2; i*i<= num; i++) {
35         if(num % i == 0) {
36             return false;
37         }
38     }
39     return true;
40 }
41 }

```

7 Strings

7.1 Suffix Array

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 #define ll long long
5 #define sws ios::sync_with_stdio(false);cin.tie( NULL
6 #define print(x) for (auto &it : x) cout<<it<<' ';<<
7 #define loop(i,a,n) for(int i=a; i < n; i++)
8 #define pb(x) push_back(x);
9 #define vi vector<int>
10 #define mp(x,y) make_pair(x,y)
11 #define pii pair<int,int>
12 #define pqi priority_queue<int, vector<int>, greater<
13 const ll MOD = 1e9+7;
14 const int INF = 0x3f3f3f3f;
15 const ll LLINF = 0x3f3f3f3f3f3f3f3f;
16
17 vector<int> suffix_array(string s) {
18     s += "$";
19     int n = s.size(), N = max(n, 260);
20     vector<int> sa(n), ra(n);
21     for (int i = 0; i < n; i++) sa[i] = i, ra[i] = s[
22         i];
23     for (int k = 0; k < n; k ? k *= 2 : k++) {
24         vector<int> nsa(sa), nra(n), cnt(N);
25
26         for (int i = 0; i < n; i++) nsa[i] = (nsa[i]-
27             k+n)%n, cnt[ra[i]]++;
28         for (int i = 1; i < N; i++) cnt[i] += cnt[i
29             -1];
30         for (int i = n-1; i+1; i--) sa[--cnt[ra[nsa[i]
31             ]]] = nsa[i];
32         for (int i = 1, r = 0; i < n; i++) nra[sa[i]]
33             = r += ra[sa[i]] !=
34             ra[sa[i-1]] or ra[(sa[i]+k)%n] != ra[(sa[
35             i-1]+k)%n];
36         ra = nra;
37         if (ra[sa[n-1]] == n-1) break;
38     }
39     return vector<int>(sa.begin()+1, sa.end());
40 }
41
42 vector<int> kasai(string s, vector<int> sa) {
43     int n = s.size(), k = 0;
44     vector<int> ra(n), lcp(n);
45     for (int i = 0; i < n; i++) ra[sa[i]] = i;
46     for (int i = 0; i < n; i++, k -= !!k) {
47         if (ra[i] == n-1) { k = 0; continue; }
48         int j = sa[ra[i]+1];

```

```

46     while (i+k < n and j+k < n and s[i+k] == s[j+
47 k]) k++;
48     lcp[ra[i]] = k;
49 }
50 return lcp;
51 }
52
53 int32_t main(){
54     sws;
55     string s;
56     cin>>s;
57
58     vector<int> suf = suffix_array(s);
59     vector<int> lcp = kasai(s, suf);
60
61     ll ans = 0;
62     for(int i=0; i<s.size(); i++){
63         if(islower(s[suf[i]])){
64             int sz = s.size()-suf[i];
65             ans += (sz - lcp[i]);
66         }
67     }
68     cout<<ans<<endl;
69 }

```

7.2 Trie

```

1 // Constroi e procura por uma string em uma arvore
2 // Trie t;
3 // Trie t(qtd_char, c_min, max_size)
4 // qtd_char = qntd maxima de caracteres
5 // c_min = menor caractere
6 // max_size = tamanho maximo de strings
7
8 // Complexidade - O(N*|s|*qtd_char)
9
10 #include <bits/stdc++.h>
11 using namespace std;
12
13 #define sws std::ios::sync_with_stdio(false); cin.tie
14 (NULL); cout.tie(NULL);
15 const int MAX = 2005;
16
17 class Trie{
18     int node = 1;
19     char c_min;
20     int qtd_char, max_size;
21     vector<vector<int>> trie;
22     vector<int> pref, end;
23
24 public:
25     void add(string s){
26         int cur = 1;
27         for(auto c: s){
28             if(!trie[cur][c-c_min]){
29                 trie[cur][c-c_min] = ++node;
30             }
31             cur = trie[cur][c-c_min];
32             pref[cur]++;
33         }
34         end[cur]++;
35     }
36
37     void erase(string s){
38         int cur = 1;
39         for(auto c: s){
40             cur = trie[cur][c-c_min];
41             pref[cur]--;
42         }
43         end[cur]--;
44     }

```

```

45 int find(string s){
46     int cur = 1;
47     for(auto c: s){
48         if(!trie[cur][c-c_min]) return 0;
49         cur = trie[cur][c-c_min];
50     }
51     return cur;
52 }
53
54 int count_pref(string s){
55     return pref[find(s)];
56 }
57
58 Trie(int qtd_char_=26, char c_min_ = 'a', int
59 max_size_=MAX):
60     c_min(c_min_), qtd_char(qtd_char_), max_size(
61     max_size_){
62     trie.resize(max_size, vector<int>(qtd_char));
63     pref.resize(max_size);
64     end.resize(max_size);
65 }
66
67 int32_t main(){ sws;
68     Trie t;
69     t.add("abcd");
70     t.add("ad");
71     t.erase("ad");
72     cout<<t.count_pref("a")<<endl;
73
74     return 0;
75 }

```

7.3 Zfunction

```

1 // complexidades
2 // z - O(|s|)
3 // match: O(|s|+|p|)
4 vector<int> z_func(string s){
5     int n = s.size();
6     vector<int> z(n);
7     int l=0, r=0;
8     for (int i = 1, i < n; i++) {
9         if (i <= r)
10             z[i] = min(z[i - l], r - i + 1);
11         while (i + z[i] < n && s[z[i]] == s[i + z[i]
12 ]))
13             z[i]++;
14         if (i + z[i] - 1 > r)
15             l = i, r = i + z[i] - 1;
16     }
17     return z;
18 }
19 // string matching
20 // quantas vezes B aparece em A
21 int32_t main(){ sws;
22     string a, b;
23     cin>>a>>b;
24
25     string s = b + '$' + a;
26     vector<int> z = z_func(s);
27     int ans = 0;
28     for(int i=0; i<z.size(); i++){
29         if(z[i] == b.size())ans++;
30     }
31     cout<<ans<<endl;
32
33     return 0;
34 }

```

8 Template

8.1 Template

```
1 #include <bits/stdc++.h>
2 using namespace std;
3 //alias comp='g++ -std=c++17 -g -O2 -Wall -
  Wconversion -Wshadow -fsanitize=address,undefined
  -fno-sanitize-recover -ggdb -o out'
4
5 #define sws std::ios::sync_with_stdio(false); cin.tie
  (NULL); cout.tie(NULL);
6 #define int long long
7 #define endl "\n"
8 #define input(x) for (auto &it : x) cin >> it
9 #define pb push_back
10 #define all(x) x.begin(), x.end()
11 #define ff first
12 #define ss second
13 #define TETO(a, b) ((a) + (b-1))/(b)
14 #define dbg(msg, x) cout << msg << " = " << x << endl
15 #define print(x,y) for (auto &it : x) cout << it
16
17 typedef long long ll;
18 typedef long double ld;
19 typedef vector<int> vi;
20 typedef pair<int,int> pii;
21 typedef priority_queue<int, vector<int>, greater<int>
  >> pqi;
22
23 const ll MOD = 1e9+7;
24 const int MAX = 1e4+5;
25 const ll LLINF = 0x3f3f3f3f3f3f3f3f;
26 const double PI = acos(-1);
27
28 int32_t main(){ sws;
29
30
31     return 0;
32 }
```

9 zExtra

9.1 Formulasmat

```
1 int gcd(int a, int b) {
2     if (b == 0) return a;
3     return gcd(b, a % b);
4 }
5
6 // number of elements
7 long long sum_of_n_first_squares(int n) {
8     return (n * (n - 1) * (2 * n - 1)) / 6;
9 }
10
11 // first element, last element, number of elements
12 long long sum_pa(int a1, int an, int n) {
13     return ((a1 + an) * n) / 2;
14 }
15
16 // first element, number of elements, ratio
17 long long general_term_pa(int a1, int n, int r) {
18     return a1 + (n - 1) * r;
19 }
20
21 // first term, numbers of elements, ratio
22 long long sum_pg(int a1, int n, int q) {
23     return (a1 * (fexp(q, n) - 1)) / (q - 1);
24 }
25
26 // -1 < q < 1
```

```
27 // first term, ratio
28 long long sum_infinite_pg(int a1, double q) {
29     return a1 * (1 - q);
30 }
31
32 // first term, number of elements, ratio
33 long long general_term_pg(int a1, int n, int q) {
34     return a1 * fexp(q, n - 1);
35 }
36
37 // first element of original pa, first element of
  derived pa, number of elements of original pa,
  ratio of derived pa
38 long long sum_second_order_pa(int a1, int b1, int n,
  int r) {
39     return a1 * n + (b1 * n * (n - 1)) / 2 + (r * n * (n
  - 1) * (n - 2)) / 6
40 }
41
42 // log
43 int intlog(double base, double x) {
44     return (int)(log(x) / log(base));
45 }
46
47 // sum from one to n
48 (n * (n + 1)) / 2
49
50 // gcd
51 long long gcd(long long a, long long b){
52     return b ? gcd(b, a % b) : a;
53 }
54
55 // or just use __gcd(a,b)
56
57 // lcm
58 long long lcm(long long a, long long b){
59     return (a / __gcd(a,b) * b);
60 }
61
62 // distancia manhattan
63 // https://vjudge.net/contest/539684#problem/H
64
65 // distancia euclidiana
66
67 // GEOMETRIA
68 // seno
69 a / sen(a) = b / sen(b) = c / sen(c)
70
71 //cosseno
72 a^2 = b^2 + c^2 - 2*b*c*cos(a)
73
74 // area losango
75 A = (1/2) * diagonal_maior * diagonal_menor
76
77 // volume prisma
78 V = B * H
79
80 //volume esfera
81 V = (4/3) * PI * R^3
82
83 //volume piramide
84 V = (1/3) * B * H
85
86 //volume cone
87 V = (1/3) * PI * R^2 * H
88
89 //condicao de existencia
90 a - b | < c < a + b
91
92 // combinacao sem rep.
93 C(n x) = n! / (x! (n-x)!)
94
95 // combinacao com rep.
```

```

96 C(n m) = (m + n - 1)!/(n! (m-1)!)
97
98 // perm sem rep
99 p = n!
100
101 // perm com rep
102 p = n!/(rep1! rep2! ... repn!)
103
104 // perm circ
105 P = (n-1)!

```

9.2 Getline

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 // Sempre usar cin.ignore() entre um cin e um getline
4 int main() {

```

```

5
6 string s1; cin>>s1;
7 cin.ignore();
8 while (true) {
9     string s; getline(cin, s);
10    if (s == "PARO") break;
11    cout<<"A"<<endl;
12 }
13 string s2; cin>>s2;
14 cin.ignore();
15 while (true) {
16    string s3; getline(cin, s3);
17    if (s3 == "PARO") break;
18    cout<<"A"<<endl;
19 }
20 }

```