

```

1. 1.segmented_seive
2. bitset<mx7>visited;
3. int prime[mx7], sz=0,cs=0;
4. vector<ll>sg;
5. void seive()
6. {
7.     for(ll i=4; i<mx7; i+=2)visited[i]=1;
8.     for(ll i=3; i*i<mx7; i+=2)
9.     {
10.         if(visited[i])continue;
11.         for(ll j=i*i; j<mx7; j+=(2*i))
12.         {
13.             visited[j]=1;
14.         }
15.     }
16.     prime[0]=2;
17.     int cnt=1;
18.     for(int i=3; i<mx7; i+=2)
19.     {
20.         if(!visited[i])prime[cnt++]=i;
21.     }
22.     sz=cnt;
23. }
24. void segmented_seive(ll l,ll r)
25. {
26.     sg.clear();
27.     ll root=sqrt(r)+1;
28.
29.     for(ll i=1; i<=r; i++)sg.pb(i);
30.     if(l==0)sg[1]=0;
31.     else if(l==1)sg[0]=0;

```

https://pastebin.com/print/gK48UqAF

1/42

```

32.
33.     for(ll i=0; prime[i]<=root; i++)
34.     {
35.         ll d=prime[i];
36.         ll start=d*d;
37.         if(start<l)start=((l+d-1)/d)*d;
38.
39.         for(ll j=start; j<=r; j+=d)
40.         {
41.             sg[j-1]=0;
42.         }
43.     }
44. }
45. vector<ll>v;
46. void input()
47. {
48.     ll n,m,cnt=0,d,l,r;
49.     sc2(l,r);
50.     segmented_seive(l,r);
51.     for(ll i=1; i<=r; i++)
52.     {
53.         if(sg[i-1]!=0)v.pb(sg[i-1]);
54.     }
55.     //for(int i=0;i<v.size();i++)cout<<v[i]<<" ";
56.     ll lenh=v.size();
57.     printf("Case %d: %lld\n",++cs,lenh);
58.     v.clear();
59. }
60. 2.segment_tree_with_Ordered_set (L to R kth max)
61. #include<bits/stdc++.h>
62. #include<ext/pb_ds/assoc_container.hpp>

```

https://pastebin.com/print/gK48UqAF

2/42

```

63. using namespace std;
64. #include<ext/pb_ds/tree_policy.hpp>
65. using namespace __gnu_pbds;
66. typedef tree<pair<ll,ll>,null_type,less< pair<ll,ll>
67. >,rb_tree_tag,
68. tree_order_statistics_node_update>ordered_set;
69. ordered_set tr[4*mx5];
70. ll a[mx6];
71. void build(int u,int b,int e)
72. {
73.     if(b==e)
74.     {
75.         tr[u].insert({a[b],b});
76.         return;
77.     }
78.     ll mid=(b+e)/2;
79.     build(2*u,b,mid);
80.     build(2*u+1,mid+1,e);
81.     for(int i=b; i<=e; i++)tr[u].insert({a[i],i});
82. }
83. ll query(int u,int b,int e,int i,int j,int val,int id)
84. {
85.     if(e<i or b>j)return 0;
86.     else if(b>=i and e<=j)
87.     {
88.         ll sz=tr[u].order_of_key({val+1,id});
89.         return sz;
90.     }
91.     int mid=(b+e)/2;
92.     ll left=query(2*u,b,mid,i,j,val,id);
93.     ll right=query((2*u)+1,mid+1,e,i,j,val,id);

```

https://pastebin.com/print/gK48UqAF

3/42

```

93.     return left+right;
94. }
95. void update_delete (int u,int b,int e,int i,int j,int
96. val,int id)
97. {
98.     if(e<i or b>j)return;
99.     else if(b>=i and e<=j)
100.     {
101.         tr[u].erase(tr[u].find({val,id}));
102.         return;
103.     }
104.     int mid=(b+e)/2;
105.     update_delete(2*u,b,mid,i,j,val,id);
106.     update_delete((2*u)+1,mid+1,e,i,j,val,id);
107.     tr[u].erase(tr[u].find({val,id}));
108. }
109. void update_add(int u,int b,int e,int i,int j,int x,int
110. id)
111. {
112.     if(e<i or b>j)return;
113.     else if(b>=i and e<=j)
114.     {
115.         tr[u].insert({x,id});
116.         return;
117.     }
118.     int mid=(b+e)/2;
119.     update_add(2*u,b,mid,i,j,x,id);
120.     update_add((2*u)+1,mid+1,e,i,j,x,id);
121.     tr[u].insert({x,id});

```

https://pastebin.com/print/gK48UqAF

4/42

```

122. void input()
123. {
124.     ll n,m,res=0,l,r;
125.     sc(n);
126.     for(int i=1; i<=n; i++)
127.     {
128.         sc(a[i]);
129.         update_add(1,1,mx5,i,i,a[i],i);
130.     }
131.     sc(m);
132.     while(m--)
133.     {
134.         ll check,l,r,k,x,v;
135.         sc(check);
136.         if(check==1)
137.         {
138.             sc(v);
139.             a[++n]=v;
140.             update_add(1,1,mx5,n,n,v,n);
141.         }
142.         else if(check==2)
143.         {
144.             update_delete(1,1,mx5,n,n,a[n],n);
145.             a[n--]=0;
146.         }
147.         else
148.         {
149.             ll L,R;
150.             sc(L),sc(R),sc(k);
151.             k=(R-L+1)-k+1;
152.             l=1,r=mx9;

```

https://pastebin.com/print/gK48UqAF

5/42

```

153.         ll f=0;
154.         while(l<=r)
155.         {
156.             if(l==r)
157.             {
158.                 f++;
159.                 if(f==2)break;
160.             }
161.             ll mid=(l+r)/2;
162.             if(quary(1,1,mx5,L,R,mid,mx18)>=k) r=mid;
163.
164.             else l=mid+1;
165.         }
166.
167.         printf("%lld\n",l);
168.
169.     }
170.
171. }
172.
173. }
174. 3.segment_tree
175. void build(int u,int b,int e)
176. {
177.     if(b==e)
178.     {
179.         tr[u]=a[b];
180.         return;
181.     }
182.     ll mid=(b+e)/2;
183.     build(2*u,b,mid);

```

https://pastebin.com/print/gK48UqAF

6/42

```

184.     build(2*u+1,mid+1,e);
185.     tr[u]=min(tr[(2*u)],tr[(2*u)+1]);
186. }
187. ll quary(int u,int b,int e,int i,int j)
188. {
189.     if(e<i or b>j)return mx18;
190.     else if(b>=i and e<=j)
191.     {
192.         return tr[u];
193.     }
194.     int mid=(b+e)/2;
195.     ll left=quary(2*u,b,mid,i,j);
196.     ll right=quary((2*u)+1,mid+1,e,i,j);
197.     return min(left,right);
198. }
199. ll bs(ll l,ll r,ll value)
200. {
201.     ll left=l;
202.     ll f=0,ans=-1;
203.     while(l<=r)
204.     {
205.         if(l==r)
206.         {
207.             f++;
208.             if(f==2)break;
209.         }
210.         ll mid=(l+r)/2;
211.         if(quary(1,1,n,left,mid)<value)
212.         {
213.             r=mid;
214.             ans=mid;

```

https://pastebin.com/print/gK48UqAF

7/42

```

215.     }
216.     else l=mid+1;
217. }
218. return ans;
219. }
220.
221. void input()
222. {
223.     ll m,val,ans=0,size=0;
224.     cin>>n>>m;
225.     for(int i=1; i<=n; i++)cin>>a[i];
226.     build(1,1,n);
227.     while(m--)
228.     {
229.         cin>>x;
230.         ll pre_id=bs(1,x-1,a[x]);
231.         //(1.....x-1 indx er modde x theke sob cheyere
232.         dure kun indx a[x] theke chuto
233.         ll suf_id=bs(x+1,n,a[x]);
234.         //(x+1 .....n indx er modde sobh theke kache
235.         kun indx a[i] theke chuto
236.         if(pre_id!=-1)cout<<"age nai cuto value"<<endl;
237.         else cout<<pre_id<<endl;
238.         if(suf_id!=-1)cout<<"pore nai cuto value"<<endl;
239.         else cout<<suf_id<<endl;
240.     }
241. }
242. 4.Optimize_Seive
243. const int N=mx8;
244. bitset<N>visited;

```

https://pastebin.com/print/gK48UqAF

8/42

```

244. int prime[N],sz=0;
245. void seive()
246. {
247.     for(ll i=4;i<N;i+=2)visited[i]=1;
248.     for(ll i=3; i<N; i+=2)
249.     {
250.         if(visited[i])continue;
251.         for(ll j=i*i; j<N; j+=(2*i))
252.         {
253.             visited[j]=1;
254.         }
255.     }
256.     prime[0]=2;
257.     int cnt=1;
258.     for(int i=3;i<N;i+=2)
259.     {
260.         if(!visited[i])prime[cnt++]=i;
261.     }
262.     sz=cnt;
263. }
264. 5.prime power of All possible divisor generator
265. vector< pair<ll,ll>>pr;
266. void f(int i, ll x)
267. {
268.     if(i == pr.size())
269.     {
270.         cout<<x<<" ";
271.         return ;
272.     }
273.     int p=pr[i].second;
274.     ll y=1;

```

https://pastebin.com/print/gK48UqAF

9/42

```

275.     for(int j=0 ; j<=p ; j++)
276.     {
277.         f(i+1, x*y);
278.         y*=pr[i].first;
279.     }
280. }
281. void input()
282. {
283.     f(0,1);
284. }
285. 6.Euler totient phi table
286.
287. void pre_calculation()
288. {
289.     phi[1] = 1 ;
290.     for(int i=2; i<=1000000; i++)
291.     {
292.         if(phi[i]==0)
293.         {
294.             phi[i] = i-1 ;
295.             for(int j=i+1; j<=1000000; j+=i)
296.             {
297.                 if(phi[j]==0) phi[j] = j ;
298.                 phi[j] = phi[j] - phi[j]/i ;
299.             }
300.         }
301.     }
302.     for(int i=1; i<=1000000; i++)
303.     {
304.         for(int j=i; j<=1000000; j+=i)
305.         {

```

https://pastebin.com/print/gK48UqAF

10/42

```

306.         S[j] = S[j] + (ll) (i*phi[i]);
307.     }
308. }
309. }
310. 6.nCr , mod is not prime
311.
312. ll fac[2*mx7],smallprime[2*mx7],largeprime[2*mx7];
313. ll big_mod(ll b,ll p,ll m )
314. {
315.     ll res=1;
316.     while(p!=0)
317.     {
318.         if(p&1)res=(res*b)%m;
319.         b=(b*b)%m;
320.         p=p>>1;
321.     }
322.     res=(res)%m;
323.     return res;
324. }
325. void pre()
326. {
327.     //mod=103003811=103*1000037
328.     fac[0]=1;
329.     for(ll i=1; i<2*mx7; i++)
330.     {
331.         ll cnt=0;
332.         ll num=i;
333.         while(num%103==0)
334.         {
335.             cnt++;
336.             num/=103;

```

https://pastebin.com/print/gK48UqAF

11/42

```

337.     }
338.     ll cnt2=0;
339.     while(num%1000037==0)
340.     {
341.         cnt2++;
342.         num/=1000037 ;
343.     }
344.     fac[i]=(fac[i-1]*num)%mod;
345.     smallprime[i]=smallprime[i-1]+cnt;
346.     largeprime[i]=largeprime[i-1]+cnt2;
347. }
348. }
349. ll ext_euclid(ll a, ll b)
350. {
351.     ll q, ps=1, s=0, pt=0, t=1, r;
352.     if(b==0) while(1);
353.     while(a%b != 0)
354.     {
355.         q = a/b; r = a-q*b;
356.         ll tmps=s, tmppt=t;
357.         s = ps-q*s, t = pt-q*t;
358.         ps = tmps, pt = tmppt;
359.         a=b; b=r;
360.         if(b==0) while(1);
361.     }
362.     return (t+mod)%mod;
363. }
364. ll nCr(ll n,ll r)
365. {
366.
367.     if(r>n)return 0;

```

https://pastebin.com/print/gK48UqAF

12/42

```

368.     if(r==0)return 1;
369.     if(n<=0)return 0;
370.     if(r<0)return 0;
371.
372.     ll res=(fac[n]);
373.     ll temp=fac[n-r]*fac[r];
374.     temp=ext_euclid(mod,temp);
375.     res=(res*temp)%mod;
376.
377.     ll smlprime=smallprime[n]-smallprime[n-r]-
smallprime[r];
378.     ll lrgprime=largeprime[n]-largeprime[n-r]-
largeprime[r];
379.
380.     res=(res*big_mod(103,smlprime,mod))%mod;
381.     res=(res*big_mod(1000037,lrgprime,mod))%mod;
382.     return res;
383. }
384. void input()
385. {
386.     ll n,r,ans=0;
387.     sc2(n,r);
388.     ans=nCr(n,r);
389.     printf("%lld\n",ans);
390. }
391. }
392. 7.Ordered set
393. #include<bits/stdc++.h>
394. #include<ext/pb_ds/assoc_container.hpp>
395. using namespace std;
396. #include<ext/pb_ds/tree_policy.hpp>

```

```

397. using namespace __gnu_pbds;
398. typedef tree<int,null_type,less<int>,rb_tree_tag,
399.     tree_order_statistics_node_update>ordered_set;
400. ordered_set st;
401. void input()
402. {
403.     int n,m,ans=0,sum=0;
404.     sci(n);
405.     for(int i=0;i<n;i++)
406.     {
407.         cin>>m;
408.         st.insert(m);
409.     }
410.     //n=10.....0 1 4 5 6 8 9 10 11 15
411.     cout<<*(st.find_by_order(2))<<endl; //output=3;
412.     cout<<st.order_of_key(4)<<endl;
413.     //strictly less then 4 number of element output 2 ta
414.     int r=7,l=4;
415.     //upper_bound(r)-lower_bound(l)
416.     cout<<st.order_of_key(r+1)- st.order_of_key(l)
<<endl;
417.     int x=2;
418.     if(st.find(x)!=st.end())
419.     {
420.         st.erase(st.find(x)); //value x delete
421.     }
422. }
423. 8.LCS
424. char in[1000][1000];
425. int lcs[1000][1000];
426. void print(string A,int i,int j)

```

```

427. {
428.     if(i==0||j==0)
429.     {
430.         return ;
431.     }
432.     if(in[i][j]=='D')
433.     {
434.         print(A,i-1,j-1);
435.         cout<<A[i-1];
436.     }
437.     else if(in[i][j]=='U')
438.     {
439.         print(A,i-1,j);
440.     }
441.     else print(A,i,j-1);
442. }
443. int lcs_length(string A,string B)
444. {
445.     int m,n,i,j,k;
446.     m=A.size();
447.     n=B.size();
448.     for(i=0; i<=m; i++)
449.     {
450.         for(j=0; j<=n; j++)
451.         {
452.             if(i==0||j==0)
453.             {
454.                 lcs[i][j]=0;
455.             }
456.             else if(A[i-1]==B[j-1])
457.             {

```

```

458.         lcs[i][j]=1+lcs[i-1][j-1];
459.         in[i][j]='D';
460.     }
461.     else if(lcs[i-1][j]>=lcs[i][j-1])
462.     {
463.         lcs[i][j]=lcs[i-1][j];
464.         in[i][j]='U';
465.     }
466.     else
467.     {
468.         lcs[i][j]=lcs[i][j-1];
469.         in[i][j]='L';
470.     }
471.     }
472. }
473. return lcs[m][n];
474. }
475. 9.Rod Cutting
476. void rod_cutting(int n,int price[])
477. {
478.     int tab[n+1][n+1];
479.     for(int i=0; i<=n; i++)
480.     {
481.         for(int j=0; j<=n; j++)
482.         {
483.             if(i==0 or j==0)tab[i][j]=0;
484.             else if(j<i) tab[i][j] = tab[i-1][j];
485.             else tab[i][j] = max(tab[i-1][j], price[i]+tab[i][j-
i]);
486.         }
487.     }

```

```

488.     int i = n, j = n;
489.     vector<int> cut_point;
490.     while(i> 0 and j>0)
491.     {
492.         if(tab[i][j] == tab[i-1][j]) i--;
493.         else
494.         {
495.             cut_point.push_back(i);
496.             j = j-i;
497.         }
498.     }
499. }
500.
501. 10.Knapsack
502. ll knapSack( ll W, ll wt[], ll p[], ll n)
503. {
504.     ll i, w;
505.     ll k[n+1][W+1];
506.     for (i = 0; i <= n; i++)
507.     {
508.         for (w = 0; w <= W; w++)
509.         {
510.             if (i==0 || w==0)k[i][w] = 0;
511.             else if (wt[i-1] <= w)
512.             {
513.                 k[i][w] = max((p[i-1] + k[i-1][w-wt[i-1]]), k[i-
514.                 1][w]);
515.             }
516.             else k[i][w] = k[i-1][w];
517.         }
518.     }

```

https://pastebin.com/print/gK48UqAF

17/42

```

518.     return k[n][W];
519. }
520.
521. 11. Trie with Bit
522. (Binary trie all subrray xor sum
523.     with xor k in maximum,minimum(spoj XORX))
524. ll to[mx6][2],to_node=1,cs=0;
525. void trie_add_string(bitset<32> s)
526. {
527.     ll cur=1;
528.     for(int i=31; i>=0; i--)
529.
530.         int ch=s[i];
531.         if(!to[cur][ch])to[cur][ch]=++to_node;
532.         cur=to[cur][ch];
533.     }
534. }
535. ll trie_quary_max(bitset<32> s)
536. {
537.     ll cur=1,ans=0;
538.     ll d=1<<30;
539.     d*=2LL;
540.     for(int i=31; i>=0; i--)
541.     {
542.         int ch=s[i];
543.         if(to[cur][ch^1])
544.         {
545.             cur=to[cur][ch^1];
546.             ans+=d;
547.         }
548.         else cur=to[cur][ch];

```

https://pastebin.com/print/gK48UqAF

18/42

```

549.         d/=2;
550.     }
551.     return ans;
552. }
553.
554. ll trie_quary_min(bitset<32> s)
555. {
556.     ll cur=1,ans=0;
557.     ll d=1<<30;
558.     d*=2LL;
559.     for(int i=31; i>=0; i--)
560.     {
561.         int ch=s[i];
562.         if(to[cur][ch] )
563.         {
564.             cur=to[cur][ch];
565.         }
566.         else {
567.             ans+=d;
568.             cur=to[cur][ch^1];
569.         }
570.         d/=2;
571.     }
572.     return ans;
573. }
574.
575. void input()
576. {
577.     ll n,l,r,res=0;
578.     sc(n);
579.     sc(r);

```

https://pastebin.com/print/gK48UqAF

19/42

```

580.     bitset<32>ma(0);
581.     trie_add_string(ma);
582.     ll mx=0,pre=0;
583.     ll mn=mx18;
584.
585.     ll ans=0;
586.     for(int i=0;i<n;i++){
587.
588.         sc(1);
589.         pre^=1;
590.
591.         ll d=pre^r;
592.         bitset<32>ma(d);
593.         mx=max(mx,trie_quary_max(ma));
594.         bitset<32>mb(pre);
595.         trie_add_string(mb);
596.     }
597.     printf("%lld\n",mx^r);
598. }
599. 12. Trie template
600. void trie_add_string(string s)
601. {
602.     int cur=1;
603.     cnt[cur]++;
604.     for(int i=0; i<s.size(); i++)
605.     {
606.         int ch=s[i]-'a';
607.         if(!to[cur][ch])to[cur][ch]=++to_node;
608.         cur=to[cur][ch];
609.         cnt[cur]++;
610.     }

```

https://pastebin.com/print/gK48UqAF

20/42

```

611.     track[cur]++;
612. }
613. int trie_quary(string s)
614. {
615.     int cur=1;
616.     for(int i=0; i<s.size(); i++)
617.     {
618.         int ch=s[i]-'a';
619.         if(!to[cur][ch])return cnt[cur];
620.         cur=to[cur][ch];
621.     }
622.     return cnt[cur];
623. }
624. 13.KMP (prefix occurs number of time)
625. ll cnt[mx6],to=0;
626. vector<ll> kmp_prefix_fun(string s)
627. {
628.     ll n=s.size();
629.     vector<ll>pi(n);
630.     //pi[0]=0;
631.     for(ll i=1; i<n; i++)
632.     {
633.         ll j=pi[i-1];
634.         while(j>0 and s[i]!=s[j])j=pi[j-1];
635.         if(s[i]==s[j])++j;
636.         pi[i]=j;
637.         cnt[j]++;
638.     }
639.     return pi;
640. }
641. int t=1;

```

```

642. void input()
643. {
644.     string a,b;
645.     cin>>a;
646.     vector<ll> v=kmp_prefix_fun(a);
647.     ll n=a.size();
648.     for(int i=n;i>=1;i--)
649.     {
650.         cnt[v[i-1]]+=cnt[i];
651.     }
652.     vector<ll>vec;
653.     while(n)
654.     {
655.         vec.pb(n);
656.         n=v[n-1];
657.     }
658.     ll sz=vec.size();
659.     cout<<sz<<endl;
660.     for( int i=sz-1;i>=0;i--)
661.     {
662.         cout<<vec[i]<<" "<<cnt[vec[i]]+1<<endl;
663.     }
664. }
665. 14.Hashing Template
666. #include <bits/stdc++.h>
667. #define ff first
668. #define ss second
669. #define mp make_pair
670. using namespace std;
671. typedef long long LL;
672. typedef pair<LL, LL> PLL;

```

```

673. const PLL M=mp(1e9+7, 1e9+9);    ///Should be large primes
674. const LL base=347;                ///Should be a prime
675. const int N = 1e6+7;              ///Highest length of
676. string
677. ostream& operator<<(ostream& os, PLL hash)
678. {
679.     return os<<"("<<hash.ff<<"", "<<hash.ss<<"");
680. }
681. PLL operator+ (PLL a, LL x)
682. {
683.     return mp(a.ff + x, a.ss + x);
684. }
685. PLL operator- (PLL a, LL x)
686. {
687.     return mp(a.ff - x, a.ss - x);
688. }
689. PLL operator* (PLL a, LL x)
690. {
691.     return mp(a.ff * x, a.ss * x);
692. }
693. PLL operator+ (PLL a, PLL x)
694. {
695.     return mp(a.ff + x.ff, a.ss + x.ss);
696. }
697. PLL operator- (PLL a, PLL x)
698. {
699.     return mp(a.ff - x.ff, a.ss - x.ss);
700. }
701. PLL operator* (PLL a, PLL x)

```

```

702. {
703.     return mp(a.ff * x.ff, a.ss * x.ss);
704. }
705. PLL operator% (PLL a, PLL m)
706. {
707.     return mp(a.ff % m.ff, a.ss % m.ss);
708. }
709. PLL power (PLL a, LL p)
710. {
711.     if (p==0) return mp(1,1);
712.     PLL ans = power(a, p/2);
713.     ans = (ans * ans)%M;
714.     if (p%2) ans = (ans*a)%M;
715.     return ans;
716. }
717. ///Magic!!!!!!
718. PLL inverse(PLL a)
719. {
720.     return power(a, (M.ff-1)*(M.ss-1)-1);
721. }
722. PLL pb[N];        ///powers of base mod M
723. PLL invb;
724. ///Call pre before everything
725. void hashPre()
726. {
727.     pb[0] = mp(1,1);
728.     for (int i=1; i<N; i++)
729.         pb[i] = (pb[i-1] * base)%M;
730.     invb = inverse(pb[1]);
731. }
732. ///Calculates Hash of a string

```

```

733. PLL Hash (string s)
734. {
735.     PLL ans = mp(0,0);
736.     for (int i=0; i<s.size(); i++)
737.         ans=(ans*base + s[i])%M;
738.     return ans;
739. }
740. ///appends c to string
741. PLL append(PLL cur, char c)
742. {
743.     return (cur*base + c)%M;
744. }
745. ///prepends c to string with size k
746. PLL prepend(PLL cur, int k, char c)
747. {
748.     return (pb[k]*c + cur)%M;
749. }
750.
751. ///replaces the i-th (0-indexed) character from right from
    a to b;
752. PLL replace(PLL cur, int i, char a, char b)
753. {
754.     cur = (cur + pb[i] * (b-a))%M;
755.     return (cur + M)%M;
756. }
757. ///Erases c from the back of the string
758. PLL pop_back(PLL hash, char c)
759. {
760.     return (((hash-c)*invb)%M+M)%M;
761. }
762. ///Erases c from front of the string with size len

```

```

763. PLL pop_front(PLL hash, int len, char c)
764. {
765.     return ((hash - pb[len-1]*c)%M+M)%M;
766. }
767.
768. ///concatenates two strings where length of the right is k
769. PLL concat(PLL left, PLL right, int k)
770. {
771.     return (left*pb[k] + right)%M;
772. }
773. ///Calculates hash of string with size len repeated cnt
    times
774. ///This is O(log n). For O(1), pre-calculate inverses
775. PLL repeat(PLL hash, int len, LL cnt)
776. {
777.     PLL mul = (pb[len*cnt] - 1) * inverse(pb[len]-1);
778.     mul = (mul%M+M)%M;
779.     PLL ans = (hash*mul)%M;
780.
781.     if (pb[len].ff == 1)    ans.ff = hash.ff*cnt;
782.     if (pb[len].ss == 1)    ans.ss = hash.ss*cnt;
783.     return ans;
784. }
785. ///Calculates hashes of all prefixes of s including empty
    prefix
786. vector<PLL> hashList(string &s)
787. {
788.     int n = s.size();
789.     vector<PLL> ans(n+1);
790.     ans[0] = mp(0,0);
791.

```

```

792.     for (int i=1; i<=n; i++)
793.         ans[i] = (ans[i-1] * base + s[i-1])%M;
794.     return ans;
795. }
796. ///Calculates hash of substring s[l..r] (1 indexed)
797. PLL substringHash(const vector<PLL> &hashlist, int l, int
    r)
798. {
799.     int len = (r-l+1);
800.     return ((hashlist[r] - hashlist[l-1]*pb[len])%M+M)%M;
801. }
802. vector<PLL> ha,hb;
803. int l1,r1,n;
804. bool ok(int mid)
805. {
806.     map<PLL,int>ma;
807.     for(int i=0; i<n; i++)
808.     {
809.         if(i+mid-1<n)
810.         {
811.             PLL d=substringHash(hb,i+1,i+1+mid-1);
812.             ma[d]++;
813.         }
814.         else break;
815.     }
816.     for(int j=0; j<n; j++)
817.     {
818.         if(j+mid-1<n)
819.         {
820.             PLL d1=substringHash(ha,j+1,j+1+mid-1);
821.             if(ma[d1])

```

```

822.         {
823.             l1=j;
824.             r1=j+mid-1;
825.             return 1;
826.         }
827.     }
828.     else break;
829. }
830. return 0;
831. }
832. char buffer[N];
833. int main()
834. {
835.     hashPre();
836.     int t;
837.     t=1;
838.     for (int cs=1; cs<=t; ++cs)
839.     {
840.         string a,b;
841.         cin>>n>>a>>b;
842.         int na = a.size(), nb = b.size();
843.         hb=hashList(b);
844.         ha = hashList(a);
845.         int l=0,r=nb+1,f=0;
846.         while(l<r)
847.         {
848.             if(l==r)
849.             {
850.                 f++;
851.                 if(f==2)break;
852.             }

```

```

853.         int mid=(l+r+1)/2;
854.         if(ok(mid))l=mid;
855.         else r=mid-1;
856.     }
857.     for(int i=l1; i<=r1; i++)
858.     {
859.         cout<<a[i];
860.     }
861. }
862. }
863.
864.
865. 14.All possible increasing sequence count;
866. ll mn[400005],a[400005];
867. map<ll,ll>ma;
868. vector<ll>v;
869. void setall2()
870. {
871.     ma.clear();
872.     v.clear();
873.     memset(mn,0,sizeof(mn));
874. }
875. void build(ll u,ll b,ll e)
876. {
877.     if(b==e)
878.     {
879.         mn[u]=0;
880.         return ;
881.     }
882.     ll mid=(b+e)/2;
883.     build(2*u,b,mid);

```

```

884.     build((2*u)+1,mid+1,e);
885.     mn[u]=mn[2*u]+mn[(2*u) +1];
886. }
887. ll query (ll u,ll b,ll e,ll i,ll j)
888. {
889.     if(e<i or b>j)return 0;
890.     if(b>=i && e<=j)return mn[u];
891.     ll mid=(b+e)/2;
892.     ll l_mn=query(2*u,b,mid,i,j);
893.     ll r_mn=query(2*u+1,mid+1,e,i,j);
894.     return ((l_mn+r_mn)%mod+mod)%mod;
895. }
896.
897. void update(ll u,ll b,ll e,ll l,ll x)
898. {
899.     if( l>e or l<b)
900.     {
901.         return ;
902.     }
903.     else if(l<=b && e<=l)
904.     {
905.         mn[u]=((mn[u]+x)%mod+mod)%mod;
906.         return ;
907.     }
908.     ll mid=(b+e)/2;
909.     update(2*u,b,mid,l,x);
910.     update(2*u+1,mid+1,e,l,x);
911.     mn[u]=((mn[2*u]+mn[(2*u)+1])%mod+mod)%mod;
912. }
913. void setall()
914. {

```

```

915.
916.     sort(v.begin(),v.end());
917.     v.erase(unique(v.begin(),v.end()),v.end());
918.     for(i=0; i<v.size(); i++)
919.     {
920.         ma[v[i]]=i+1;
921.     }
922. build(1,1,v.size());
923. }
924. void solve()
925. {
926.
927.     for(i=0; i<n; i++)
928.     {
929.         ll s=query(1,1,v.size(),1,ma[a[i]]-1)+1;
930.         update(1,1,v.size(),ma[a[i]],s);
931.     }
932.     printf("Case %lld: %lld\n",++f,mn[1]);
933.
934. }
935. void input()
936. {
937.     sc(t);
938.     while(t--)
939.     {
940.         setall2();
941.         sc(n);
942.         for(i=0; i<n; i++)
943.         {
944.             sc(a[i]);
945.             v.pb(a[i]);

```

```

946.     }
947.     setall();
948.     solve();
949. }
950.
951. }
952. 15.All possible subarray xor sum
953. int main()
954. {
955.     cin>>t;
956.     while(t--)
957.     {
958.         cin>>n;
959.         long long int b[n+1];
960.         b[0]=0;
961.         sum=0;
962.         long long int a[33]= {0};
963.         for(i=1; i<=n; i++)
964.         {
965.             cin>>m;
966.             b[i]=b[i-1]^m;
967.         }
968.
969.         k=1;
970.         for(i=0; i<=32; i++)
971.         {
972.             c=0;
973.             for(j=1; j<=n; j++)
974.             {
975.                 if(b[j]&(1LL<<i))c++;
976.             }

```



```

977.         sum+=(c*(n+1-c)*k);
978.         k=k<<1;
979.     }
980.     cout<<sum<<endl;
981. }
982. }
983. 16.Bigmod()
984. ll big_mod(ll b,ll p,ll m )
985. {
986.     ll res=1;
987.     while(p!=0)
988.     {
989.         if(p&1)res=(res*b)%m;
990.         b=(b*b)%m;
991.         p=p>>1;
992.     }
993.     res=(res)%m;
994.     return res;
995. }
996. 17.ALL pair LCM sum
997.
998. ll cnt[mx6],b[mx6],exact[mx6];
999. void seive()
1000. {
1001.     for(ll i=1; i<mx6; i++)
1002.     {
1003.         for(ll j=i; j<mx6; j+=i)
1004.         {
1005.             b[i]+=(j*cnt[j]);
1006.         }
1007.         b[i]%=mod;

```

https://pastebin.com/print/gK48UqAF

33/42

```

1008.     }
1009. }
1010.
1011. void input()
1012. {
1013.     ll n,m,ans;
1014.     cin>>n;
1015.     for(ll i=0; i<n; i++)
1016.     {
1017.         cin>>m;
1018.         cnt[m]++;
1019.     }
1020.     seive();
1021.     for(ll i=1000000; i>=1; i--)
1022.     {
1023.         exact[i]=(b[i]*b[i])%mod;
1024.         for(int j=i+i; j<=1000000; j+=i)
1025.         {
1026.             exact[i]=(exact[i]-exact[j]);
1027.             if(exact[i]<0)exact[i]+=mod;
1028.
1029.         }
1030.         ans=(ans+exact[i]*big_mod(i,mod-2ll,mod))%mod;
1031.         ans=(ans-(i*cnt[i]))%mod;
1032.         if(ans<0)ans+=mod;
1033.     }
1034.     ans=(ans*big_mod(2,mod-2,mod))%mod;
1035.     ans=(ans+mod)%mod;
1036.
1037.     printf("%lld\n",ans);
1038.

```

https://pastebin.com/print/gK48UqAF

34/42

```

1039. }
1040. //Exact Gcd g
1041. void input()
1042. {
1043.     for(ll i=1000000; i>=1; i--)
1044.     {
1045.         if(b[i]==0)d=0;
1046.         else exact[i]=d;
1047.         for(int j=i+i; j<=1000000; j+=i)
1048.         {
1049.             exact[i]=(exact[i]-exact[j]);
1050.         }
1051.     }
1052.
1053.     printf("%lld\n",exact[G]);
1054.
1055. }
1056. //String stream
1057. //string to num
1058. int x;
1059. string a="375834";
1060. stringstream ss(a);
1061. ss>>x;
1062. cout<<x<<endl;
1063. //number to string
1064. int n=45;
1065. stringstream ss;
1066. ss<<n;
1067. string a=ss.str();
1068. cout<<a<<endl;
1069.

```

https://pastebin.com/print/gK48UqAF

35/42

```

1070. ll count word(string a)
1071. {
1072.     stringstream ss(a);
1073.     string word;
1074.     while(ss>>word)cnt++;
1075.     return cout<<cnt<<endl;
1076. }
1077.
1078. //Catalan Number
1079.
1080. Cn=(2n!)/(((n+1)! * (n!)));
1081. n=0,1,2,3,...
1082. cn=1,1,2,5,14,42,132,429,1430....
1083.
1084. //Increasing sequence
1085. ll b[mx6],a[mx6],i;
1086. void bs(ll m)
1087. {
1088.     ll l=0,r=v.size()-1;
1089.     while(l<r)
1090.     {
1091.         ll mid=(l+r-1)/2;
1092.         if(v[mid]>=m)r=mid;
1093.         else l=mid+1;
1094.     }
1095.     if(v[r]==m)
1096.     {
1097.         b[i]=r;
1098.     }
1099.     else if(v[r]<m)
1100.     {

```

https://pastebin.com/print/gK48UqAF

36/42

```

1101.     v.pb(m);
1102.     b[i]=r+1;
1103. }
1104. else
1105. {
1106.     v[r]=min(m,v[r]);
1107.     b[i]=r;
1108. }
1109. }
1110. void solve()
1111. {
1112.     ll n,ma=0;
1113.     cin>>n;
1114.     for(i=0;i<n;i++)
1115.     {
1116.         cin>>a[i];
1117.     }
1118.     v.pb(a[0]);
1119.     for( i=1;i<n;i++)
1120.     {
1121.         bs(a[i]);
1122.     }
1123.     for(i=0;i<n;i++)ma=max(ma,b[i]);
1124.     cout<<ma+1<<endl;
1125. }
1126.
1127. //COMSOD(n)=i to n sumof( sum of number of divisor)
1128.
1129. for(int i=2;i*i<=n;i++)
1130. {
1131.     j=n/i;

```

https://pastebin.com/print/gK48UqAF

37/42

```

1132.     ans+=((i+j)*(j-i+1))/2;
1133.     ans+=i*(j-i);
1134. }
1135. cout<<ans<<endl;
1136.
1137. //Multiset
1138. int main()
1139. {
1140.     multiset<int> st//increasing order
1141.     for (itr = gquiz1.begin(); itr != gquiz1.end(); ++itr)
1142.     {
1143.         cout << *itr << " ";
1144.     }
1145.     cout << endl;
1146.     // remove all elements up to element
1147.     // with value 30 in gquiz2
1148.     gquiz2.erase(gquiz2.begin(), gquiz2.find(30));
1149.     // remove all elements with value 50 in gquiz2
1150.     int num;
1151.     num = gquiz2.erase(50);
1152.     // lower bound and upper bound for multiset gquiz1
1153.     cout << "\ngquiz1.lower_bound(40) : \n"
1154.           << *gquiz1.lower_bound(40) << endl;
1155.     cout << "gquiz1.upper_bound(40) : \n"
1156.           << *gquiz1.upper_bound(40) << endl;
1157.     return 0;
1158. }
1159.
1160. //Array Range check
1161.

```

https://pastebin.com/print/gK48UqAF

38/42

```

1162. ca=max(a,l);
1163. cb=min(b,r);
1164. if(ca>cb)cout<<out of range;
1165. else cout<<in range;
1166.
1167. //All possible Permutation a[i]!=i
1168. int count(int n)
1169. {
1170.     if(n==1)return 0;
1171.     if(n==2)return 1;
1172.     return (n-2)*((count(n-1)+count(n-2)));
1173. }
1174.
1175. //NEXT elecent after delete using Bitset
1176.
1177. bitset<100000>bit;
1178. bit.flip();
1179. 1111111111111111
1180. bit[1]=0;
1181. bit[4]=
1182. 1011011111
1183. bit._Find_next(b-1);
1184. // b index er soman or pore kothay bit on ache
1185.
1186. //Locas Combination bignumber nCr
1187. mod=99999997;
1188. llu Locascombination(llu n,llu r)
1189. {
1190.     if(r==0)return 1;
1191.     llu ni=n%mod,ri=r%mod;
1192.     if(ni<ri)return 0;

```

https://pastebin.com/print/gK48UqAF

39/42

```

1193.     return (((fac[ni]*big_mod(fac[ri],mod-
1194.     2,mod)%mod)*bigmod(fac[ni-ri],mod-
1195.     2,mod))%mod)*Locascombination(n/mod,r/mod)%mod);
1196. }
1197. //Combinatorics star and vars
1198. 1. x1+x2+x3+x4...+xk=n;
1199. xi>=1;
1200. number of way=(n-1)C (k-1)
1201.
1202. 2.
1203. x1+x2+x3+x4...+xk<=n;
1204. xi>=1;
1205. solution :
1206. x1+x2+x3+x4...+xk + m <=n;
1207. number of way=summation of (indx m=0 to n-k) (n-m-1)C(k-
1208. 1);
1209. //Degree of Time
1210.
1211. ans=abs((11*m -60*h)/2);
1212. //Template
1213. #include<bits/stdc++.h>
1214. using namespace std;
1215. #define TT ios::sync_with_stdio(false);
1216. cin.tie(0);cout.tie(0)
1217. #define ll long long int
1218. #define ull unsigned long long int
1219. #define vi vector<int>
1220. #define vc vector<char>
1221. #define vs vector<string>

```

https://pastebin.com/print/gK48UqAF

40/42

```
1220. #define      vll          vector<long long int>
1221. #define      vp            vector< pair<int,int>
>
1222. #define      pb            push_back
1223. #define      pob           pop_back
1224. #define      pll           pair<long long int,
long long int>
1225. #define      F            first
1226. #define      S            second
1227. #define      sc(x)         scanf("%lld",&x)
1228. #define      sci(x)         scanf("%d",&x)
1229. #define      sc2(x,y)       scanf("%lld
%lld",&x,&y)
1230. #define      pf            printf
1231. #define      min3(a,b,c)    min(a,b<c?b:c)
1232. #define      max3(a,b,c)    max(a,b>c?b:c)
1233. #define      all(v)         v.begin(), v.end()
1234. #define      rall(v)         v.rbegin(), v.rend()
1235. ///===== CONSTANT =====///
1236. #define mx18  1000000000000000000
1237. #define mx9   1000000007
1238. #define mx8   100000007
1239. #define mx7   10000006
1240. #define mx6   1000056
1241. #define mx5   200005
1242. #define mx4   10005
1243. #define inf   1<<30
1244. #define eps   1e-9
1245. #define mod    mx9
1246. ll dx[] = {1,-1,0,0};
1247. ll dy[] = {0,0,1,-1};
```

```
1248. long double PI = acosl(-1);
1249. ///===== Debugging
=====///
1250. #define      debug(x)      cerr << #x << " = " << x << endl
1251. #define      debug2(x, y)   cerr << #x << ": "
<< x << " " << #y << ": " << y << endl;
1252. #define      debug3(x, y, z) cerr << #x << ": "
<< x << " " << #y << ": " << y << " " << #z << ": " << z
<< endl;
1253. #define      debug4(a, b, c, d) cerr << #a << ": "
<< a << " " << #b << ": " << b << " " << #c << ": " << c
<< " | " << #d << ": " << d << endl;
1254. ///=====Bitmask=====///
1255. //ordered_set st;
1256. //int Set(int N,int pos){return N=N | (1<<pos);}
1257. //int reset(int N,int pos){return N= N & ~(1<<pos);}
1258. //bool check(int N,int pos){return (bool)(N & (1<<pos));}
```