LAZY PROP(counting primes btn L & R)

```
#include<bits/stdc++.h>
using namespace std;
#define ll long long
#define mx 1000005
ll a[10005],tree[40005],lazy[40005];
ll vis[mx];
void sieve()
{
   vis[1]=1;
   for(ll i=2;i<mx;i++)
   {
      if(vis[i]==0)
      {
         for(ll j=i*i;j<mx;j+=i)
            vis[j]=1;
      }
   }
}

void build(ll node,ll lo,ll hi)
{
   if(lo==hi)
   {
      tree[node]=!vis[a[lo]];
      return;
   }
   ll left=node*2,right=left+1;
   ll mid=(lo+hi)/2;
   build(left,lo,mid);
   build(right,mid+1,hi);

   tree[node]=tree[left]+tree[right];
}

void prop(ll node,ll lo,ll hi)
{
   if(lo==hi)
      return;

   ll mid=(lo+hi)/2;
   ll left=node*2,right=left+1;
   tree[left]=(mid-lo+1)*lazy[node];
   tree[right]=(hi-mid)*lazy[node];
   lazy[left]=lazy[right]=lazy[node];

   lazy[node]=-1;
}
```

```
void update(ll node,ll lo,ll hi,ll i,ll j,ll val)
{
   if(lazy[node]!=-1)
      prop(node,lo,hi);
   if(lo>j||hi<i)
      return;
   if(lo>=i&&hi<=j)
   {
      tree[node]=(hi-lo+1)*val;
      lazy[node]=val;
      return;
   }
   ll left=node*2,right=left+1;
   ll mid=(lo+hi)/2;
   update(left,lo,mid,i,j,val);
   update(right,mid+1,hi,i,j,val);
   tree[node]=tree[left]+tree[right];
}

ll query(ll node,ll lo,ll hi,ll i,ll j)
{
   if(lazy[node]!=-1)
      prop(node,lo,hi);
   if(lo>j||hi<i)
      return 0;
   if(lo>=i&&hi<=j)
      return tree[node];
   ll left=node*2,right=left+1;
   ll mid=(lo+hi)/2;
   ll p=query(left,lo,mid,i,j);
   ll q=query(right,mid+1,hi,i,j);
   return p+q;
}


int main()
{
   sieve();
   ll t,cs=0;
   cin >> t;
   while(t--)
   {
      memset(tree,0,sizeof(tree));
      memset(lazy,-1,sizeof(lazy));
      ll n,q;
      scanf("%lld%lld",&n,&q);
      for(ll i=1;i<=n;i++)
```

```
            scanf("%lld",&a[i]);
        build(1,1,n);

        printf("Case %lld:\n",++cs);
        for(ll i=1;i<=q;i++){
            ll x,u,v,val;
            scanf("%lld",&x);
            if(x==0)
            {
                scanf("%lld%lld%lld",&u,&v,&val);
                update(1,1,n,u,v,!vis[val]);
            }
            else
            {
                scanf("%lld%lld",&u,&v);
                ll ans=query(1,1,n,u,v);
                printf("%lld\n",ans);
            }
        }


    }
}
```

TRIE SAMPLE :

```
#include<bits/stdc++.h>
using namespace std;
int trie[100005][15];
int main()
{
   int t,n;
   cin >>t ;
   while(t--)
   {
      memset(trie,0,sizeof(trie));
      char ch[10005][12];
      int endp[100005]={},cnt[100005]={};

      int next=1;
      scanf("%d",&n);
      for(int i=0;i<n;i++){
         int now=0;
         scanf("%s",ch[i]);
         int sz=strlen(ch[i]);
      for(int j=0;ch[i][j]!='\0';j++)
      {
         int x=ch[i][j]-'0';
         int y=trie[now][x];
         if(y==0)
         {
            trie[now][x]=next++;
            now=trie[now][x];
            cnt[now]++;
            if(j==sz-1)
               endp[now]++;

         }
         else
         {
            now=trie[now][x];
            cnt[now]++;
            if(j==sz-1)
               endp[now]++;
         }

      }
      }

      int f=0;
      for(int i=0;i<n;i++)
      {
         int now=0;
```

```
        int sz=strlen(ch[i]);
        for(int j=0;ch[i][j]!='\0';j++)
        {
           int x=ch[i][j]-'0';
           int y=trie[now][x];
           if(cnt[y]==1){
              break;
           }
           else if(j==sz-1&&cnt[y]>1&&endp[y]>0)
           {
              f=1;
           }
           now=y;
        }
        if(f==1)
        {
           break;
        }
     }
   if(f==0)
      printf("YES\n");
   else
      printf("NO\n");
   }
}
```