Q: Given N input strings & Q queries. for each query check if given query is prefix of any given input string.

Substring starts at index = 0. (complete string is also a Prefix.)

Note :- length of every string <= l.

Input strings (N)	Queries (Q)	Yes/No
anaconda	anaco	✓
dress	fry	×
eaten	roade	✓
friends	algor	✓
roades	sour	×
anaco	dress	✓
algorithms		
sound		

Idea :-

1) Insert all the words in a Trie.
2) for every query word, iterate over the trie from root & check query is Prefix or not.

TC : $O(N \times l + Q \times l)$

SC : $O(Nl)$

* Using Trie DS, searching prefix is most optimal.

⇒ Trie ≡ Prefix Tree.

Q: Given a binary matrix mat[N][M], find the no. of distinct rows.

mat[7][5]

	0	1	2	3	4	
0	1	0	0	1	0	✗
1	1	1	0	1	1	✗
2	0	1	0	1	0	✓
3	1	1	0	1	1	✓
4	1	1	0	0	1	✓
5	1	0	0	1	0	✓
6	0	0	1	1	0	✓

ans = 5

Every row, consider only it's one occurrence

Idea1 :-

For every row, compare it with the rows below it, If freq. == 0, count++.

TC: $O(N^2 \cdot M)$ SC: $O(1)$

$$\left.\begin{array}{lcl} 0 & \Rightarrow & N-1 \\ 1 & \Rightarrow & N-2 \\ 2 & \Rightarrow & N-3 \\ \vdots & & \vdots \\ & & 1 \end{array}\right\} \simeq O(N^2)$$

Idea 2 :-

Convert each row into String & insert into HashSet.

$$TC : \underbrace{N \times M}_{\substack{\uparrow \\ \text{Converting each} \\ \text{row into} \\ \text{String.}}} + \underbrace{N \times M}_{\substack{\uparrow \\ \text{insert } \textcircled{N} \text{ strings} \\ \text{of length M into} \\ \text{HashSet.}}}$$

$$SC : O(MN)$$

Idea 3 :-

	16	8	4	2	1	
	0	1	2	3	4	Decimal
0	1	0	0	1	0	18
1	1	1	0	1	1	27
2	0	1	0	1	0	10
3	1	1	0	1	1	27
4	1	1	0	0	1	25
5	1	0	0	1	0	18
6	0	0	1	1	0	6

⇒ 5

* For every row, convert it into decimal & insert into the HashSet.

$$TC : \underbrace{N \times M}_{\substack{\text{Convert every} \\ \text{row to a} \\ \text{decimal no.}}} + \underbrace{N}_{\substack{\text{insert N int's} \\ \text{into set.}}}$$

: $O(N \cdot M)$

SC : $O(N)$

↳ Hashset of N integers.

Issues

$M <= 31 \Rightarrow$ int

$M <= 63 \Rightarrow$ long

$M > 63$

$M = 200$?

* for larger values of M, this approach won't work.

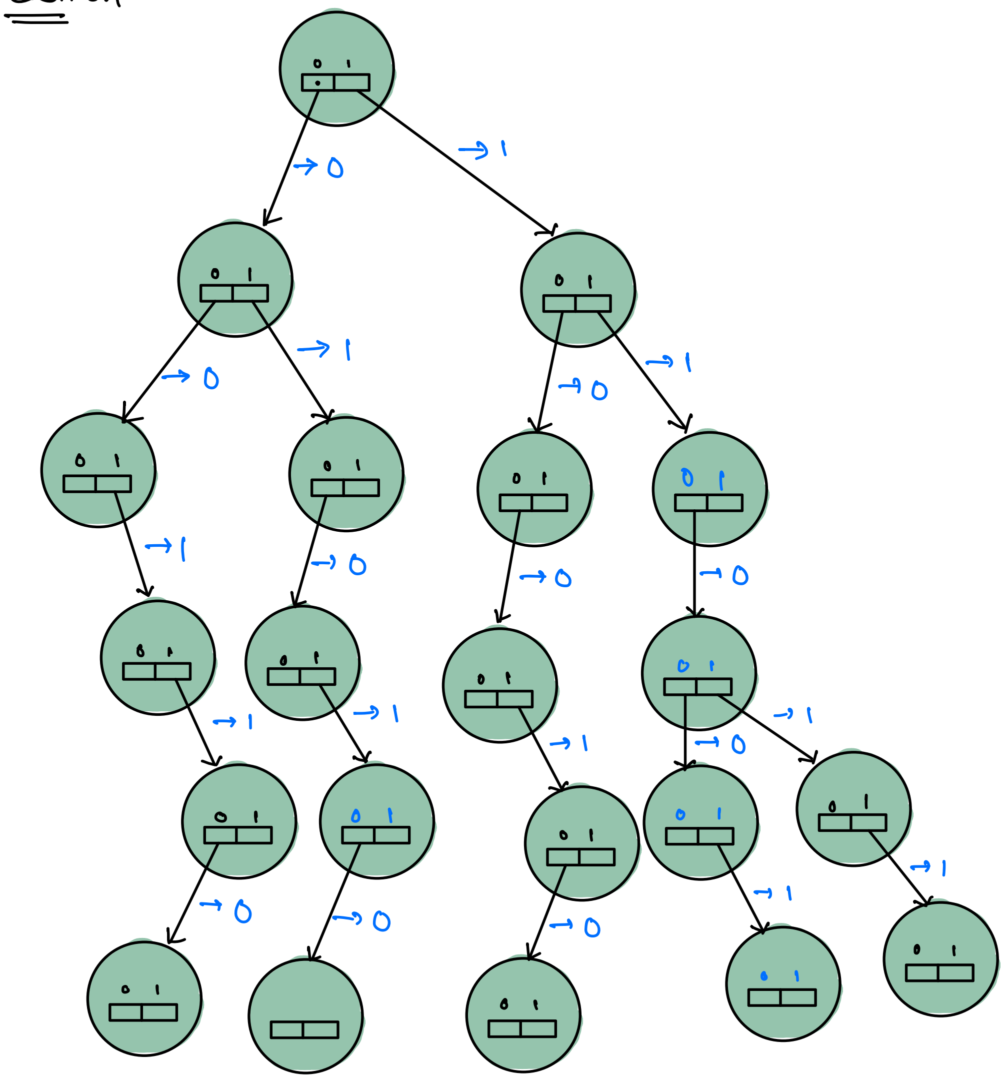
Idea 4 :-

* Insert every row in a Trie.

*
```
Class Node { // Binary Trie
    Node C[2];
    Node () {
        C[0] = Null;
        C[1] = Null;
    }
}
```

data $<^{0}_{1}$

Trie Creation



* While inserting a row in Trie, if we are not creating even a single new node, it means entire row was already present in the Trie.

Code :-

```
    int   uniqueRows ( int mat[][] , N, M) {
              Count = 0
              Node  root  =  new  Node();
              for( i= 0 ; i< N;  i++ ) {
                      // Insert  mat[i]
                      if ( insert (root, mat[i], M) )
                              Count++
              }
              return  count;
    }
```

will return True if we have created even a single node while inserting mat[i].

```
  bool   insert ( root, arr[] , M) {
         bool  flag  =  false;
         for( i= 0;  i< M ; i++ ){
                e = arr[i];
                if ( root.c[e] == Null ){
                        // create  a  new  Node
                        root.c[e]  =  new  Node();
                        flag = true
                        root  =  root.c[e];
                }
                else{
                        root  =  root.c[e];
                }
         }
         return  flag;
  }
```

TC : $O(N \times M)$

SC : $O(N \cdot M)$ [Worst Case]

Q. Given an Array of size N, find the pair with max XOR value.

⇒ i, j s.t arr[i] ^ arr[j] is MAX , i != j

N = 4

0	1	2	3
4	3	2	7

A[0] ^ A[1] = 7

A[0] ^ A[2] = 6

A[0] ^ A[3] = 3 ⇒ ans = 7

A[1] ^ A[2] = 1

A[1] ^ A[3] = 4

A[2] ^ A[3] = 5

Hint

A B

1 0 0 0 0 vs 0 1 1 1 1

A > B.

N = 9

0	1	2	3	4	5	6	7	8
22	61	38	27	21	34	42	37	43

		32	16	8	4	2	1
22	:	0	1	0	1	1	0
61	:	1	1	1	1	0	1
38	:	1	0	0	1	1	0
27	:	0	1	1	0	1	1
21	:	0	1	0	1	0	1
34	:	1	0	0	0	1	0
42	:	1	0	1	0	1	0
37	:	1	0	0	1	0	1
43	:	1	0	1	0	1	1

	5	4	3	2	1	0
A = 22 :	0	1	0	1	1	0
B = ___ :	1	0	1	0	1	1
A^B :	1	1	1	1	0	1

max value of A^B with A = 22.

⇒ Repeat the above process for every value of (A) in the Array & maintain max.

TC : O(N* (N* 31))

* Trie Approach

⇒ for all the no's, insert the same number of bits in the Trie.

⇒ find the max element in Array and find the no. of bits in this element.

```
Class Node {
      Node C[2];
      Node() {
          C[0] = Null;
          C[1] = Null;
      }
}

    int     maxXOR ( arr[], N) {
               int   me  =  max(arr); ⇒ O(N)
               int    b  =  maxSetBits(me);
               Node  root = new  Node();
               for( i= 0; i< N; i++) {
                       insert ( root, arr[i], b);
               }
                ans = 0;
                // fin element A[i] & get max XOR.
                for( i = 0; i< N; i++) {
                       ans = max(ans, query (root, A[i], b));
                 }
                return ans;
    }
```

```
void  insert ( root, ele , b) {
    for ( i= b; i>=0; i--) {
        // ith bit in ele.
        int e = checkBit (ele, i);
        if ( root.c[e] == Null) {
            root.c[e] = new Node();
            root = root.c[e];
        }
        else {
            root = root.c[e];
        }
    }
}

int  query ( root , ele , b) {
    ans = 0
    for (i = b; i>=0; i--) {
        // ith bit in ele.
        int e = checkBit (ele, i);
        // If e -> 0 => we need 1 ]  1-e
        // If e -> 1 => we need 0 ]
        if ( root.c[1-e] != Null ) {
            // We can set ith bit in ans.
            ans = ans+ (1<<i)
            root = root.c[1-e];
        }
        else{
            root = root.c[e];
        }
    }
    return ans;
}
```

TC :- $O(N \times B + N \times B)$

SC : $O(N \cdot B)$

——— * ———