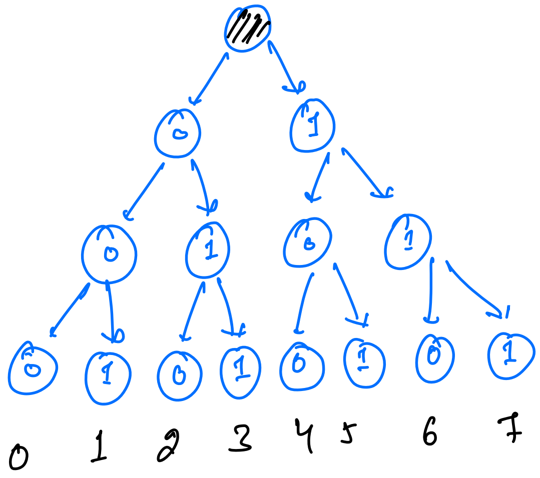
Trie → data structure.

↳ prefix tree.

Bit Representation

Tries on bit representation



0-7.

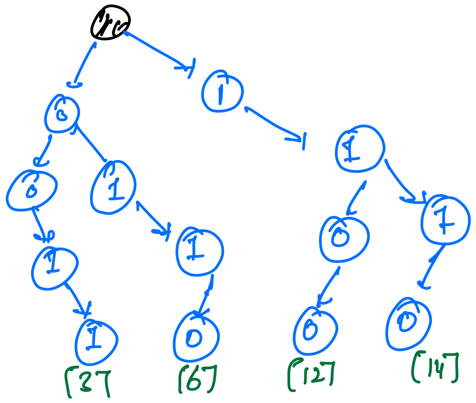
0 →	000
1 →	001
2 →	010
3 →	011
4 →	100
5 →	101
6 →	110
7 →	111

```
Class Node {
        int data;
        Node children[2];
}
```

children[0]   children[1]

[3, 6, 12, 14].

3 →	0011
6 →	0110
12 →	1100
14 →	1110



Q: Max value of XOR pair. [A[i] ^ A[j] is maximum]

A → [9 8 10 7]

xor = 15.

A	B	A^B.
0	0	0
1	1	0
0	1	1
1	0	1

idea-1 → Consider all the pairs. T.C → $O(N^2)$

$$\begin{array}{rccccccc} A \rightarrow & 1 & 0 & 1 & 1 & 1 & 0 & 1 \\ B \rightarrow & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ \hline & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{array}$$

$$\begin{array}{rccccccc} A \rightarrow & 1 & 0 & 1 & 1 & 1 & 0 & 1 \\ B \rightarrow & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ \hline & 0 & 1 & 1 & 1 & 1 & 1 & 1 \end{array}$$

$$\begin{array}{rccccccc} A \rightarrow & 1 & 0 & 1 & 1 & 1 & 0 & 1 \\ C \rightarrow & 0 & 0 & 1 & 1 & 1 & 0 & 1. \\ \hline & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}$$

$$\left[2^i > 2^0 + 2^1 + 2^2 + \text{——} \; 2^{i-1} \right]$$

MSB matters.

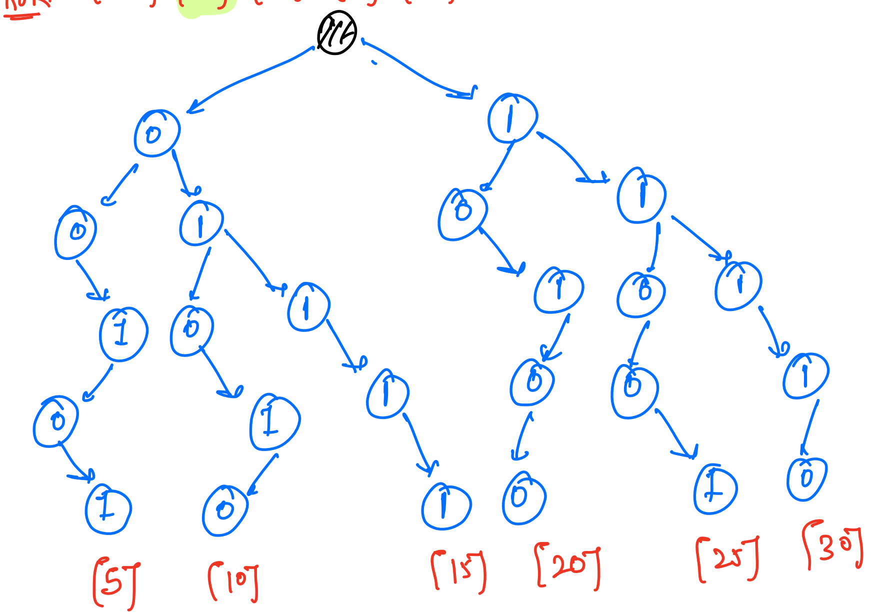
arr[] → [5, 10, 15, 20, 25, 30]

$$\begin{array}{cccccc} 5 & 10 & 15 & 20 & 25 & 30 \\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ 25 & 20 & 20 & 10 & 5 & 5 \end{array}$$

XoR [28] [30] [27] [30] [28] [27]

30 → 5 bits.

5 →	00101
10 →	01010
15 →	01111
20 →	10100
25 →	11001
30 →	11110



10 → 01010
demand → 10101

15 → 01111
demand → 10000

20 → 10100
demand → 01011

25 → 11001
demand → 00110

$$\begin{array}{rr} 10 \rightarrow & 01010 \\ 20 \rightarrow & 10100 \\ \hline & 11110 \\ \hline \end{array}$$

$$\begin{array}{rr} 15 \rightarrow & 01111 \\ 20 \rightarrow & 10100 \\ \hline & 11011 \\ \hline \end{array}$$

30 → 11110
demand → 00001

T.C → O(N * max no. of bits)

$$\left\{ \begin{array}{l} T.C \rightarrow O(N * \log_2 max) \\ S.C \rightarrow O(N * \log_2 max) \end{array} \right\}$$

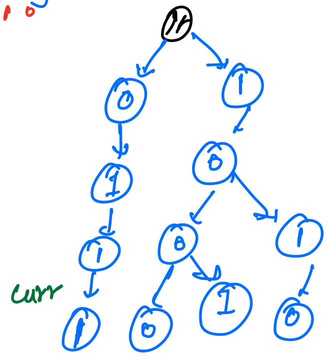
pseudo-code.

→ Build the trie (root) , x → log (max)
(no. of bits)

```
ans = 0
for( i=0 ; i < N ; i++){
    xor = 0 , curr = root
    for( j = x-1 ; j >= 0 ; j--){
        if ( jth bit of curr element is 1){
            if ( curr.children [0] != NULL){
                //set jth bit in xor-value.
                xor |= (1<<j);
                curr = curr. children [0]
            }
            else{
                curr = curr. children [1]
            }
        }
        else{
            if ( curr children [1] != NULL)
                // set the jth bit in xor-value
                xor |= (1<< j) ;
                curr = curr children[1];
            }
            else{
                curr = curr. children [0];
            }
        }.
    }
} ans = Math.max( ans , xor) ;
return ans;
```

[9, 8, 10, 7]

[1000]
3 2 1 0



ans = ~~0~~ ~~14~~ 15 , x = 4

xor = (~~0000~~)
(1111)
= 15.

Q: Maximum Subarray XOR.

arr → [1 4 3]

1	→	1
4	→	4
3	→	3
1,4	→	5
4,3	→	7
1,4,3	→	6

idea.1. Consider all the subarrays. T.C → $O(N^2)$

a	b	c	d	e	f	g	h

$a \wedge b \wedge c \wedge d \wedge e \wedge f$
$\wedge\ a \wedge b \wedge c$

$a+b+c+d+e+f$
$-(a+b+c)$

idea.2. → Convert given arr[ ] into prefix XOR [ ].

[pfxor[i] = xor of all elements from 0 to i]

$xor(i\text{-}j) = \boxed{pfxor[j] \wedge pfxor[i-1]}$

Basically, you are looking for a pair with max XOR value in the pfxor[ ].

{ individual - pfxor[i] } consider them as well.

Bundle Strings

N strings ≡ divide them into groups of K. [N is divisible by K]

[Score of a group = length of longest common prefix substring of that group]

→ find the total maximum score of all the groups.

eg. N=6, K=2

A
AB
ABBB
A
ABBB
ABBB
↓ ↓ ↓ ↓
$\frac{6}{2}$ $\frac{4}{2}$ $\frac{3}{2}$ $\frac{3}{2}$

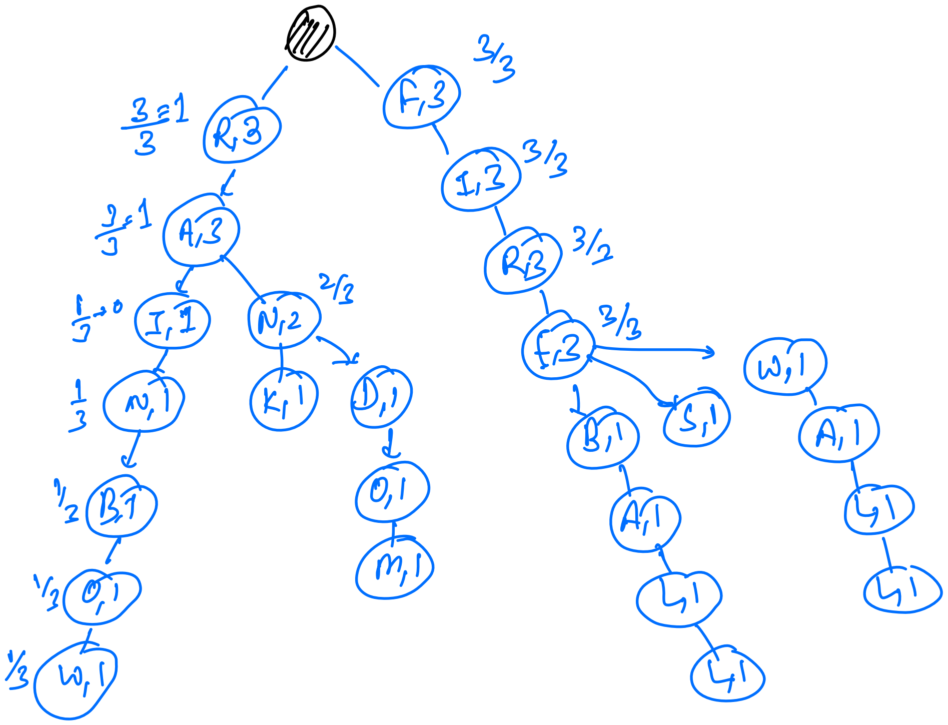
G1	G2	G3
A AB	ABBB A	ABBB ABBB
1 +	1 +	4 = 6.

A A	AB ABBB	ABBB ABBB
1 +	2 +	4 = 7.

N=6

RAINBOW ✓
FIREBALL ✓
RANK ✓
RANDOM ✓
FIREWALL ✓
FIRES ✓

K=3.



ans = 6.

```
Node {
    char data;
    Node children[26]
    int freq;
}
```

traversing trie.

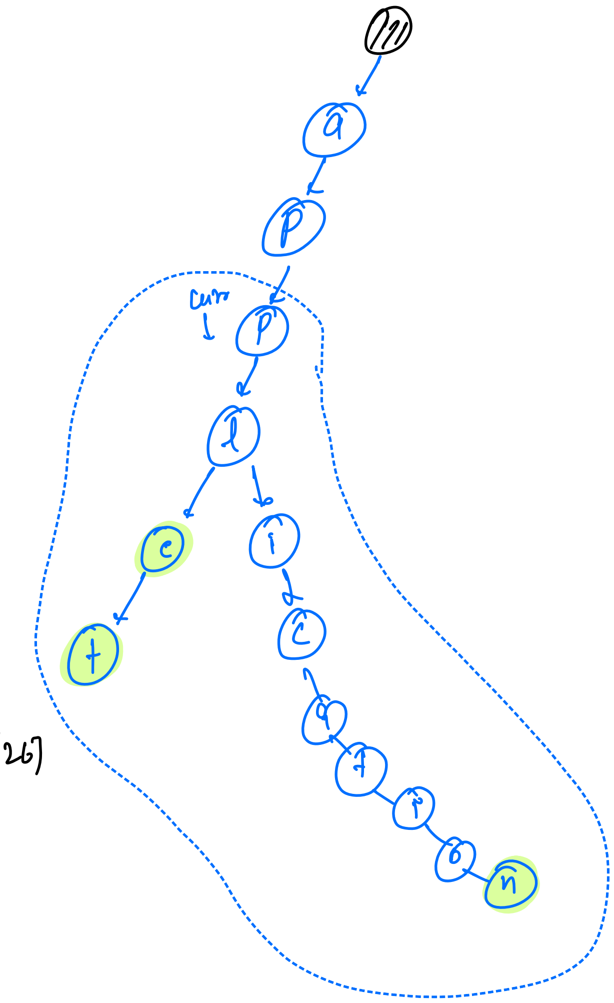
```
void traverse ( root, K) {

        ans += root.freq / K;
        for( i = 0 ; i < 26 ; i++) {
            if ( root.children [i] != NULL) {
                traverse( root.children [i] , K)
            }
        }
}
```

→ prefix - string
→ Bitwise representation of numbers

→ Trie.

apple
applet
application

app



```
Node {
  char data
  Node children[26]
}
```

```
void traverse (root) {
    →
      traverse (root.left);
      traverse (root.right);
}
```

```
void traverse ( root) {
    for(i=0; i<26 ; i++){
        if ( root.children[i] != NULL){
            traverse(root.children[i])
        }
}
```