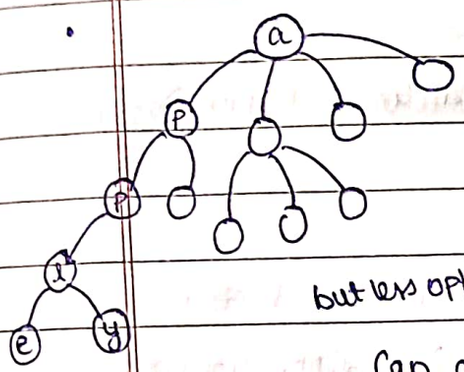
Trie

- Just like AVL trees it uses the self balancing approach with $O(\log_2 n)$ complexities. Here it's slightly more optimised $O(\log_3 n)$ by making more than two childrens (n children) so it reduces traversing through $\log_2 n$.

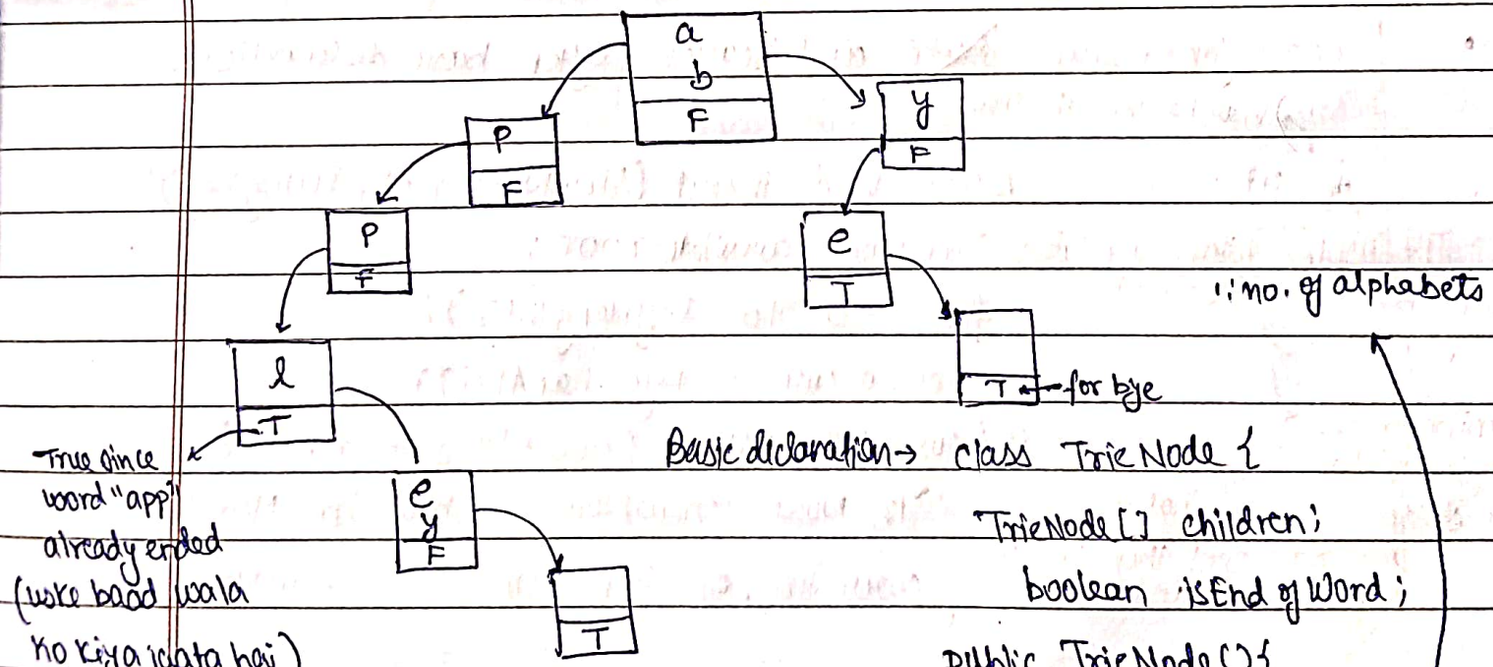


- It is also called as a prefix Tree.
  ⇒ apple, apply → where there's prefix are same so when they're put in nodes they have same ancestors.

but less optimised ⇒ check this graph.

Can also do as Hashmap<character, TrieNode>

- Not contains just 2 childs, it has array of childrens.
- isEndofWord for checking kaha pr ek prefix khatam hua hai.
- Working :- "app", "apple", "bye", "by", "apply".



Basic declaration →

```
class TrieNode {
    TrieNode[] children;
    boolean isEndofWord;
    public TrieNode(){
    this.children = new TrieNode[26];
    this.isEndofWord = false;
    }
}
```

↳ initialise by false

↑ no. of alphabets

- Search in TrieNode :-

  eg. "app" → from last figure → check a is present ✓ ~~with~~

  Exist ← check next is True ✓ ← check p is present ✓ ← check p is present ✓
  (EndofWord)

  e.g. "ap" → check a ✓ → check p ✓ → check for next EndofWord is true X → Not exist.

- Why not Hashset is used, when strings (~~dictory~~ dictionary) can be added in set also ?
  → Can't optimise like this
  → Same word with prefixes is connected here but in Hashset different memory is stored for diff. word.
  → Finding element (word) of dictionary is easy here, not in Hashset usme randomly bhot hai elements stored.
  → Can't find words with same prefixes in Hashset.
  → Hence, Hashset is not better for search.

- code for insert, ~~delete~~ and search after basic declaration.
  ↗ visualise through Tree

```
insert →  static void insert (TrieNode root, String key){
          TrieNode currNode = root;
          for (i = 0 to key.length()){
            char curr = key.charAt(i);
            if (currNode.children[curr-'a'] == null){
              TrieNode newNode = new TrieNode();
              currNode.children[curr-'a'] = newNode;
            }
            currNode = currNode.children[curr-'a'];
          }
          currNode.isEndOfWord = true;
        }
```

'a' = 97 (ASCII) → Start currNode from root
if char = 'b'
∴ 'b' - 'a' = 1 ← determines index ie children char where stored
if that char is not present then create a new Node and place in index.
make children as curr. and proceed ← (else)
mark last as True. ←

Search → static boolean search (TrieNode root, string key){

same as insert ✓ {
TrieNode currNode = root;
for(i=0 to key.length()){
char curr = key.charAt(i);

If does not exist ∴ false → if(currNode.children[curr-'a']==null){
return false;

proceed further to check → currNode = currNode.children[curr-'a'];
}
if(currNode.isEndOfWord==true){ → return currNode.isEndOfWord;
return true;
}

if after finishing a word → next one is true then only it exist otherwise uska sirf prefix exist karta tha.

* check ques. videos from Apna College ~~Go~~ Video.

Q. Maximum XOR of Two nos. in array.
{8, 5, 4, 9, 1} = a[]

Sol: 1. Brute force : Take two pointers i & j and run i from 0 to a.length and j from i+1 to a.length and do a[i]^a[j].
→ $O(N^2)$ Not optimized.

2. Trie (O(n)) : • generally in ques. it can range 32 bits, for our understanding take 4 bit.
• If all bits are 1111 → maximum.

• Take any no. eg. 4 (0100) now to make it (1111) we've to find such no. from array which is close to (1011).
• Priority of closeness of no. is from MSB. eg. here MSB=1 ∴ the nos. with MSB=1 are 8, 9 move to next digit and get ans.
• Just jiss we did for 4, check for all nos. ki uska complementary kon sa sabse hai.

decimal

- ∴ Total bits in an no. = $(\log_2 N)+1$

  ∴ we are traversing through array and bits

  T.C. = $O(N \times (\log_2 N + 1))$

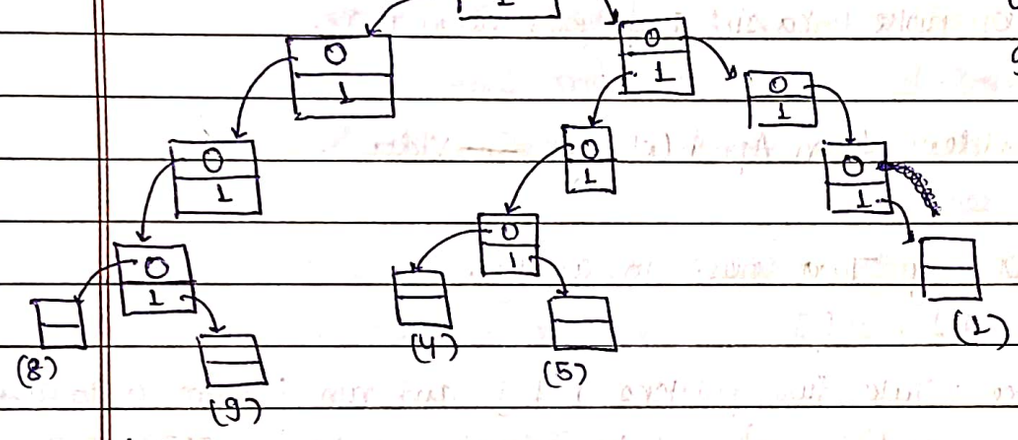
  max value = 32

  ∴ $O(N)$ ✓

- How in Trie → • insert a no. from array (bit form) in Trie.
  - then find a no. which is reverse in bit for that no.
  - No need for isEndOfWord (required only for) ∵ we're not searching here.

Trie Formation ⇒

8 = 1000
5 = 0101
4 = 0100
9 = 1001
1 = 0001



int large = 0;

take first no. → 8 (1000) → check for a bit as (0111) for (1111).

∴ we got 0101 (5) ∴ XOR with 8 will make = 1101 = (13 = large).

if 13 > 0 ✓

take second no. → 5 (0101) → check for (1010)

∴ we got 1000 ∴ XOR make = 1101 = ✗ → mtlb already nikal liye isse.

take a no. → 1 (0001) → check for (1110)

∴ we got 1000 → 9 < large ∴ ignore.

Note: To check a nos. n^{th} bit is 0 or 1.

⇒ uss no. ko right shift kar do jitne bits hai utne time then 'and' with 1 ⇒ $(n >> i) \& 1$.

- For setting the i^{th} bit → $1 << i$. (∵ it left shifts 1 i^{th} ~~items~~ times.)
  Does not work the case for no. ~~*~~ except 1 → jiss tm set nhi kar rahe hoge bas shift kar rahe hoge.

e.g. $1 << 4$ = 0000 0001 → 0001 000 . ∴ 4th bit set also ✓ (all else zero)

$4 << 4$ = 0000 0100 → 0100 0000 ∴ Not bit set.

code → (easy hai bs lamba hai)

```
static class TrieNode {
    TrieNode zero, one;
}
static void insert(TrieNode root, int n){
    TrieNode cur = root;
    for(i=31; i>=0; i--){
        int bit = (n>>i) & 1;
        if(bit==0){
            if(cur.zero==null){
                TrieNode newNode = new TrieNode();
                cur.zero = newNode;}
            else { cur = cur.zero;}
        }
        else{ if(cur.one==null){
                TrieNode newNode = new TrieNode();
                cur.one = newNode;
            }
            else → cur = cur.one;
        }
```

checks a bit is '0' or '1' →

```
static int findMaxXor(TrieNode root, int n){
    TrieNode cur = root;
    int ans = 0;
    for(int i=31; i>=0; i--){
        int bit = (n>>i) & 1;
        if(bit==1){
```

Ajanta
Page No.
Date

when we got 0 at 31st bit ✓

↑ to make ith bit 1 and store in final ans jisse compare hoga (after XOR)

```
if(curr.zero != null){
    ans += (1<<i);  → Diff. bit then ans. ki ith bit = 1
    curr = curr.zero;  → proceed
}
else{
    curr = curr.one;
}
else{
    if(curr.one != null){
        i(write on own)
    }
}
return ans;
```

∵ maximum it 111111111.... me milega na samjho!!!

```
public static int maxXor(int a[], int n){
    TrieNode root = new TrieNode();
    int ans=0;
    for(i=0 to n) {   → a.length
        insert(root, a[i]);
    }
    for(i=1 to n){   (why this 'i' check)
        ans = max(ans, findMaxXor(root, a[i]));
    }
    return ans;
}
```

Ajanta
Page No.
Date 12-07-25

Bit Manipulation and Maths

* Basics
- Check notes group in whatsapp to why we use bit manipulation.
- faster
- How to find binary no. for -ve integer.

2's complement = binary -ve inverse.

e.g. negative inverse of 5 :→ 2's complement = ① invert all bits ② Add one

$101 \rightarrow 010 \rightarrow \boxed{011} = -5$

acting as

e.g. $12 + (-5) = 1100 \;\&\; 0111$ ∴⇒

$00\ldots0000\ 1100$
$+\ 11\ldots1111\ 1011$
$00\ldots0000\ 0111 \rightarrow \boxed{7}$ Ans.

how this came?

⇒ $5 = 0000\ldots0101 \xrightarrow{2's} 1111\ldots1011$ (aisa dikhega actual mein)

- Hence, there is no subtraction here in computer, (only add) ∴ to subtract, add it with its 2's complement

- Bitwise operators → & , | , ^ , ~ , >> , <<
  AND, OR, XOR, $\bar{A}$, Right shift, Left shift

e.g. $a/2 \Rightarrow a>>1$ (same baat)
↓ better & faster

Right shift ↓ ÷2 ; Left shift ↓ ×2

e.g. $2*a \Rightarrow a<<1$
better use in loops

- Dekho! generally *, %, / are used for decimal nos. So computer converts this into binary then perform operation ∴ overhead introduced ∴ becomes slower.
- To check odd/even ∴ ~~divide~~ AND that no. by 1 if O/P = 1 (odd), 0 (even)
  $n \& 1 \rightarrow$ O/P = 1 (odd) else even → `if (n & 1 == 0) { //even }`

Ajanta
Page No.
Date

• Swap two numbers → 3 times XOR [simple]

e.g. a=5, b=7

a = a^b = 2
b = a^b = 5
a = a^b = 7

101 ^ 111 = 010 == 'a' updated
010 ^ 111 = 101
101 ^ 010 = 111

after doing a^b three times and storing in a, b, a respectively you can swap.

* Bit Masking {saara case samjho, ratna has to leave this chapter}
↳ altering the properties of a no. by performing operations.

• Find i^{th} bit

n → 1 0 0 1 1 0 1 0 1
mask → 0 0 0 1 0 0 0 0 0 (1<<5)
(left shift, 5 times) one
0 0 0 1 0 0 0 0 0 → ∵ AND
↳ non-zero ∴ "1" was at 5th bit

Now, n & mask → non-zero no. ∴ "1"
↳ else → 0.

(mask = 1<<i)

• Set i^{th} bit of any no.

n | mask ; mask = 1<<i
OR

Now, to set that bit just → n = n | mask

• Clear i^{th} bit

socho rather just formula ratna!!!

n = 1 0 0 1 (1) 0 1 0 0 1
How to clear that bit only? 1 1 1 1 0 1 1 1 1
1 0 0 1 0 0 1 0 0 1 ∵ AND

[saara 1 kardo aur uspe 0 rakho fir AND ↙]
mask

mask → left shift 1 'i' times then complement

∴ ⇒ $\sim(1<<i) = mask$

Hence, $n = n \,\&\, mask$.

*Q. Find no. of bits to change to convert a to b.
(Integer form of ~~bit~~ Edit Distance)

Sol^n. eg. a → 1 0 1 1 0
b → 1 1 0 1 1 → a^b = 0 1 1 0 1
3 bits

Approach → • XOR a and b ✓

*• Jitna baar 1 aaya utna operations.

Q. How many set bits in a number?

First of all to find no. of bits in a no. = $[\log_2 n] + 1$

e.g. $n = 8$ ∴ 3+1 = 4

$n = 26$ ∴ $\log_2 26$ → check range → 4 to 5 → floor = 4+1 = 5 Ans.

Ily. to check how many digits in a no. = $[\log_{10} n] + 1$

e.g. 13 → $[\log_{10} n] + 1$ = 1+1 = 2 ✓

Method:1 → e.g. a = 13 ⇒ 1 1 0 1 ans = 1 (if LSB = 1 ∴ +1)

right shift → 0 1 1 0 ans = 1+0

" → 0 0 1 1 ans = 1+0+1

till zero → 0 0 0 1 ans = 1+0+1+1 = 3 Ans.

→ 0 0 0 0 //stop

Traversing all bits i.e. $\log_2 n + 1$ bits ∴ $O(\log_2 n)$ complexity.

~~★★★~~ → v.imp. in CP

Method:2 → $n \,\&\, (n-1)$ ⇒ clears the least significant set bit.

e.g. a = 13 ∴ 1101 → $n \,\&\, (n-1)$ → 1100

⇒ 1100 → " → 1000

⇒ 1000 → " → 0000 Ans. count = 3 ✓ in just 3 operations only.

T.C. = O(set bits)

Dry run (one step) → n & (n-1)

e.g. n = 13 → n = 1101 ∴ n-1 = 1100

∴ 1101 & 1100 = 1100 ✓

Now, n = 12 ∴ n-1 = 11 ⇒ 1011

↳ Mat bolna tmko 2's complement iske liye karna padega. LOL!

∴ 1100 & 1011

= 1000

Now, n = 8 ; n-1 = 7

1000 & 0111 = 0000 ✓

* XOR and Imp. Bit manipulation Ques.

Note :- 1. $n \wedge n = 0$

2. $0 \wedge n = n$

(basic/easy ques)

Q.1. Find the only non-repeating element in an array where every element repeats twice.

Sol: e.g. a = [5, 4, 1, 4, 3, 5, 1] . Ans. = 3. (baaki saara two times repeat)

1. Brute force → Nested loop → $O(N^2)$ ✗

2. Hashset → elements ko set mei add karte jao, if any no. comes again then usse set se bahar nikal do. Jo bacha uso answer. → Good but space complexity ↑
T.C. O(N) O(N)

3. T.C. = O(N), S.C. = O(1)

```
int ans = 0th index = 5
for( i=1 to n){
ans = ans ^ a[i]
}
```

DRY RUN

i=1

ans = 5 ^ 4

i=2 → ans = 5 ^ 4 ^ 1

i=3 → ans = 5 ^ 4 ^ 1 ^ 4

i=4 → ans = 5 ^ 1 ^ 3

i=5 → " = 5 ^ 1 ^ 3 ^ 5

i=6 → " = 1 ^ 3 ^ 1

end → ans = 3 ✓

Ajanta
Page No.
Date

***
good ques Q.2 Sol^ Same as Q.1 but find two such non-repeating element. (rather one)

a = [5, 4, 1, 4, 3, 5, 1, 2] .

* → Socho dimag. se ruk ke ki XOR ke help se kaise solve karoge.

Ans. → When we proceed same as earlier ques. ans in the end → ans = 3 ^ 2

Now How to get 3 and 2 ?

⇒ 0 1 1
^ 0 1 0
ans = 0 0 1 → if LSB of ans is "1" ∴ Dono element ka LSB diff. hai ∵ XOR hua tha

∴ check for Kiska LSB 1 hai ya 0 (1 : odd no., 0 : even no.)

5, 1, 3, 5, 1 | 4, 4, 2

till here both ans will be separated, now XOR ans with ~~both~~ any one set

~~you'll get answers~~ (1) (3^2)^5^1^3^5^1 = (2)

~~(2) (3^2)^4^4^2~~

Now XOR with ans. ⇒ 2 ^ (3^2) = (3) → ans.

(isko temp me store kar lena for future use.)

Agar LSB is '0' then check for that jiska bit set hai .

Q.3. Find the only ..... (here el repeats thrice).

e.g. a = [2, 2, 1, 5, 1, 1, 2] ans. 5

T.C. O(N), SC (O(1)) → since, 32 bit array is constant size.

Sol^ make a count array of 32 size → (then all are initialized with zero)

→ then add all binary value places on indices

→ Jab count array ban jaye then those indices jiske ander ka value agar 3 ke multiple me hai to use zero le lo baaki ko 1 ~~(except zeros) 2 till size of bit~~

→ ans wo hoga jo binary value bana.

		...	1	3	4
			1	0	1

1 0 1 = 5 ✓

010 → 2
010 → 2
001 → 1
101 → 5
001 → 1
001 → 1
010 → 1

134

→ Get binary to decimal by multiply by power of 2

Ajanta
Page No.
Date 13/07/25

* Mathematics

* Number theory:-

1. Factorial of a no. ✓

2. Trailing zeros in factorial

eg. $12! = 1 \times 2 \times 3 \times 4 \times 5 \times 6 \times 7 \times 8 \times 9 \times \underbrace{10}_{(2\times5)} \times 11 \times 12$

- get the factorial of a no. → loop this = $n \times (n-1)!$

```
static int fact(int n){
    int ans = 1;
    for(i = 2 to = n){ ans = ans * i; }
    return ans;
```

- Find no. of 2 and 5 pairs.

in 12! → we have 2 pairs.

How to check ⇒ $\left[\frac{n}{5}\right] = 2$ ✓ (Jitna 5 hai utna, no need to check for 2 → ye hamesha rehta hai)

if no. 'n' is > 25 (5×5) ∴ ⇒ $\left[\frac{n}{5}\right] + \left[\frac{n}{25}\right]$

∵ 2 always but get extra 5.

if no. 'n' is > 125 ∴ $\left[\frac{n}{5}\right] + \left[\frac{n}{25}\right] + \left[\frac{n}{125}\right]$ Ily, proceed for 625, etc.

∴ no. of Trailing zeros = $\left[\frac{n}{5}\right] + \left[\frac{n}{25}\right] + \left[\frac{n}{125}\right] + \left[\frac{n}{625}\right] + \dots$ Ans.

code ⇒ Given 'n' →

```
int res = 0;
for(i = 5; i <= n; i = i * 5){
    res = res + n / i;
}
return res;
```

3. Palindrome No. → reverse that no. (by % with 10, append with rem, proceed by n /= 10).

4. Prime No. → $O(N^{1/2})$ is only useful (No brute force count method i.e. $O(N)$).

n%i == 0 ∴ count++;
if count = 2 → Prime
≠2 → Not Prime.

* to check a no. i.e. prime you've to check till its $\sqrt{N}$ value *

- 0, 1 → false, 2 → true, proceed with 3, increment i with 2 since, 2 ke multiple wale nos. ⁊ check karne ka koi faida nhi ∵ they are never primes.

code →
```
if (n <= 1) { return false; }
if (n == 2) { return true; }
for (i = 3 ; i*i <= n ; i+=2){        // √n can be written as i*i <= n
    if (n % i == 0) { return false; }
}
return true;
```

5. Sieve of Eraustosthenes

↳ To find all prime nos. before 'n'.

e.g. n = 12 → 2, 3, 5, 7, 11.

Approach:-

- Make a boolean array of 'n' size and initialise all with true.
- Mark all multiples of 2 then 3 as false. Why 2 & 3 →. all nos. starting from 2 to $[\sqrt{n}] \rightarrow [\sqrt{12}] = [3.78..] = 3$. leaving their first multiple (i.e. itself)
- All True's are prime.

*	2	3	4	5	6	7	8	9	10	11	12
[] :	T	T	T	T	T	T	T	T	T	T	T
2→	T	T	T	F	T	F	T	F	T	F	F
3→ (after 2)	T	T	T	F	T	F	F	F	F	T	F

code →
```
static boolean[] SOE(int n) {                    // ∵ checking till last one also
    boolean isPrime[] = new boolean[n+1];
```
mark all as true → `Arrays.fill(isPrime, true);`

Not by default false ∴ hum saabko true kr diye h. ↔
```
    isPrime[0] = false;      // ∵ index start from 0.
    isPrime[1] = false;
    for (i = 2 to i*i <= n ) {        // same condition of prime no.
```

Ajanta
Page No.
Date

for multiplying of nos.

i=3

ie. 2×i → 2×3=6=j
then j=j+i=6+3=9
∵ 9<12 → make 9 false

```
for( j= 2*i ; j<=n ; j+=i){
    isPrime[j] = false;
}
return isPrime;
```

6. GCD (greatest common divisor = HCF)

$gcd(a,b) = gcb(b, a\%b)$ → for condition that $a\%b \neq 0$

★★★ Learn (memorise direct) ★★★

if $a\%b == 0$ ∴ ans. b.

code →

```
static int gcd (int a, int b){
    if (a%.b==0) return b;
    return gcd(b, a%.b);
}
```

⑦ Leaving Modulo arithmatics for now → if solved 200 leetcode Ques. and got that it was imp. then only.