1. Largest Palindrome Substring
2. Alternate Subarray.
3. Valid Sudoku.

Q.1 Given a string, find the length of largest palindromic substring.

Amazon
GS | MS
Directi
...

S: a b a c a b ⇒ 5

S: a b c d b ⇒ 1

Quiz S: a b a e a b f ⇒ 5

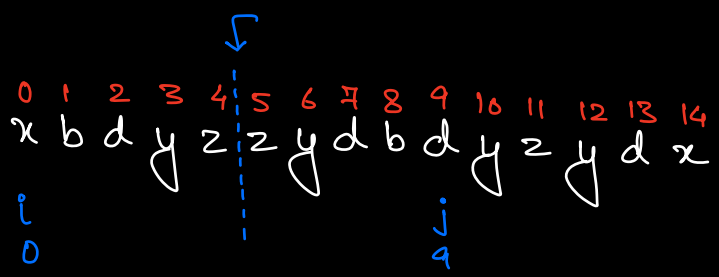
Brute Force

→ Iterate over all the substrings ⇒ $O(N^2)$

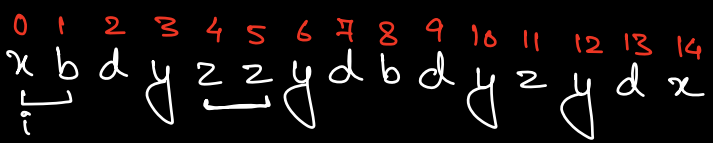
→ Check if a substring is palindrome or not. ↳ $O(N)$

TC: $O(N^3)$
SC: $O(1)$

Quiz




```
int lengthPalindrome(S, Ci, Cj)
    i = Ci, j = Cj;
    while( i >= 0 && j < N) {
        if(S[i] == S[j])
            i--, j++;
        else
            break;
    }
    return j-i-1;
}
```



i=0 → fun(S, 0, 0) ⇒ 1-(-1)-1 = 1 ✓

→ fun(S, 0, 1) ⇒ 1-0-1 = 0
i j

```
ans = 1;
for (i = 0; i < N; i++) {
    // Odd length palindrome
    ans = max(ans, lengthPalindrome(s, i, i));
    // Even length palindrome
    ans = max(ans, lengthPalindrome(s, i, i+1));
}
return ans;
```

TC : $O(N^2)$

SC : $O(1)$

TC : $O(N^2)$ SC : $O(1)$ ⟶ TC : $O(N^2)$ SC : $O(N^2)$ ⟶ TC : $O(N)$

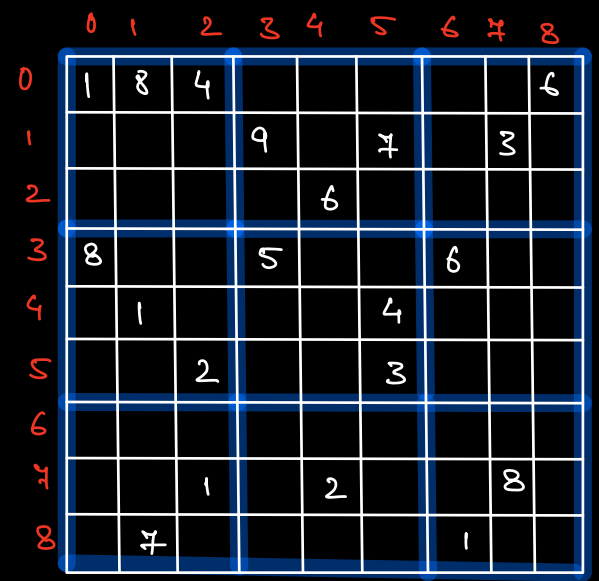
DP Soln

Manacher's Algo

Q. Valid Sudoku

Given a partially filled sudoku, check if it's valid or invalid.

9×9



	0	1	2	3	4	5	6	7	8
0	1	8	4						6
1				9		7		3	
2					6				
3	8			5			6		
4		1				4			
5			2			3			
6									
7			1		2			8	
8		7					1		

Rules :

1) No repetition in any row.
2) No repetition in any col.
3) No repetition in any 3×3 box

```
// Rows
for ( i = 0; i < 9 ; i++ ){
    HashSet<int> set;
    for ( j = 0; j < 9 ; j++ ){
        if ( set.contains ( mat[i][j] ) )
            return false;
        else
            set.add (mat[i][j]);
    }
}
// Cols
for ( i = 0; i < 9 ; i++ ){
    HashSet<int> set;
    for ( j = 0; j < 9 ; j++ ){
        if ( set.contains ( mat[j][i] ) )
            return false;
        else
            set.add (mat[j][i]);
    }
}
// 3x3 Boxes
→ (0,0)  (0,3)  (0,6)
→ (3,0)  (3,3)  (3,6)
→ (6,0)  (6,3)  (6,6)
```

```
for( i=0 ; i<9 ; i+=3){
    for( j=0 ; j<9 ; j+=3){
        // i,j => start index of 3x3 Box.
        HashSet<int> set;
        for( k=i ; k<i+3 ; k++){
            for( l=j ; l<j+3 ; l++){
                if( set.contains( mat[k][l] ))
                    return false;
                else
                    set.add(mat[k][l]);
            }
        }
    }
}
```

# of iterations = 3 x (9x9)

= 3 x 81

= 243. ≈ O(1) TC

N=9

# of iterations = $3 \times N^2$

TC : $O(N^2)$

Q. Alternate Subarrays.

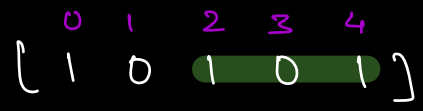
Given an int array A of size N comprising of only 0's & 1's & an integer B. Find all indices in array A that can act as a centre of a 2*B+1 length 0-1 alternate subarray.

A: [1 0 0 1 0 1 0 1 0 0 0 1 0]

{1, 0, 1, 0 ----} ✓ {0} ✓ {1 0 1 0 1} ✓

{0, 1, 0, 1, ---} ✓ {1} ✓

Ex

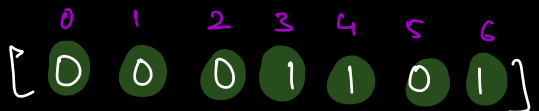


B = 1

2*1+1 = 3

ans: [1, 2, 3]

Ex



B = 0

(1)

ans:- [0 1 2 3 4 5 6]

```
bool isAlternate(arr[], s, e){
    for(i=s+1; i<=e; i++){
        if(A[i] == A[i-1])
            return false;
    }
    return true;
}

k = 2B+1
s = 0
e = k-1
while(e < N){
    // s,e
    // Check if subarray [s,e] is alternate.
    if(isAlternate(arr, s,e)){
        // Add mid index in ans array.
        ans.add(s+B);
    }
    s++
    e++
}
```

TC : $O(N^2)$
SC : $O(1)$

* 10^8 iterations / sec

Time limit : 1 sec

$N = 10^6 \Rightarrow O(N^2) \lesssim 10^{12} > 1\text{sec}$

$\Rightarrow$ TLE.

$N = 10^3 \Rightarrow O(N^2) \Rightarrow 10^6 < 1\text{sec}$

$\Rightarrow$ ✓

$N = 20 \Rightarrow O(2^N)$ ✓

$\lesssim 2^{20}$

$2^{10} \cdot 2^{10} \lesssim 10^6 < 1\text{sec}$

$2^{10} \rightarrow 1024 \lesssim 10^3$

$2^{10} \lesssim 10^3$