**Valid Parentheses**

public boolean isStringBalancedUsingContains(String s)

{

while(s.contains("{}")||s.contains("[]")||s.contains("()"))

{

s=s.replace("{}","").

replace("[]","").

replace("()","");

}

return s.equals("");

}

public boolean isValid(String s) {

Stack<Character> st = new Stack();

for(char ch: s.toCharArray()){

if(ch=='{'||ch=='['||ch=='('){

st.push(ch);

}

else{

if(st.isEmpty()) return false;

char p = st.pop();

if(ch == ')' && p == '('||ch==']' && p =='[' || ch=='}'&& p=='{') continue;

else return false;

}

}

return st.isEmpty();

}

**Longest Common Prefix** : i/p – flower, flow,flaw o/p - fl

public String longestCommonPrefix(String[] strs) {

if(strs.length==0) return "";

String result = strs[0];

for(int i=1;i<strs.length;i++){

while(strs[i].indexOf(result)!=0){

result = result.substring(0,result.length()-1);

}

}

return result;

}

**Remove All Adjacent Duplicates In String i/p – abbaca o/p ca i/p azxxzy o/p ay**

public String removeDuplicates(String s) {

Stack<Character> st = new Stack<>();

for (int i=0;i<s.length();i++){

if(st.isEmpty()){

st.push(s.charAt(i));

continue;

}

if(st.peek()==s.charAt(i)){

st.pop();

}

else{

st.push(s.charAt(i));

}

}

StringBuilder sb = new StringBuilder();

while(!st.isEmpty()){

sb.append(st.pop());

}

sb.reverse();

return sb.toString();

}

***Subdomain Visit Count*** :

Input: cpdomains = ["9001 discuss.leetcode.com"] Output: ["9001 leetcode.com","9001 discuss.leetcode.com","9001 com"]

public List<String> subdomainVisits(String[] cpdomains) {

Map<String ,Integer> map = new HashMap<>();

List<String> result = new ArrayList<>();

for (int i=0;i<cpdomains.length;i++){

String[] stringData = cpdomains[i].split(" ");

String[] str = stringData[1].split("\\.");

String subDomains = "";

for(int j = str.length-1;j>=0;j--){

subDomains = str[j]+ subDomains;

if(!map.containsKey(subDomains))

map.put(subDomains , Integer.parseInt(stringData[0]));

else

map.put(subDomains , map.get(subDomains)+Integer.parseInt(stringData[0]));

subDomains = "."+ subDomains;

}

}

for(Map.Entry<String,Integer> hm :map.entrySet()){

result.add(hm.getValue() + " "+ hm.getKey());

}

return result;

}

#### [Longest Substring Without Repeating Characters](https://leetcode.com/problems/longest-substring-without-repeating-characters/)

public int lengthOfLongestSubstring(String s) {

int i =0; //slow pointer

int j =0; //fast pointer

int result =0;

Set<Character> set = new HashSet<>();

while (j<s.length()) {

char ch = s.charAt(j);

if(!set.contains(ch)){

set.add(ch);

j++;

}

else{

set.remove(s.charAt(i));

i++;

}

result = Math.max(set.size(),result);

}

return result;

}

#### [Valid Anagram](https://leetcode.com/problems/valid-anagram/)

public boolean isAnagram(String s, String t) {

if(s.length()!=t.length()){

return false;

}

int fmap[] = new int[26];

for(int i=0;i<s.length();i++){

fmap[s.charAt(i)-'a']++;

fmap[t.charAt(i)-'a']--;

}

for(int x:fmap){

if(x!=0){

return false;

}

}

return true;

}

#### [Longest Palindromic Substring](https://leetcode.com/problems/longest-palindromic-substring/)

static boolean isPalindrome(String s){

StringBuilder sb = new StringBuilder(s);

String s1= sb.reverse().toString();

if(s.equals(s1)){

return true;

}

return false;

}

public String longestPalindrome(String s) {

if(s==null || s.length()<2)

{

return s;

}

String var ="";

String maxStr ="";

for(int i=0;i<s.length();i++)

{

for(int j=i+1;j<=s.length();j++)

{

var = s.substring(i,j);

if(isPalindrome(var)) {

if(var.length()>maxStr.length()){

maxStr=var;

}

}

}

}

if(s.length()>0 && maxStr.length()==0){

maxStr = Character.toString(s.charAt(0)) ;

}

return maxStr;

#### }

#### [Unique Email Addresses](https://leetcode.com/problems/unique-email-addresses/)

#### public int numUniqueEmails(String[] emails) {

#### Set<String> set = new HashSet<>();

#### for (int i=0;i<emails.length;i++){

#### String s[]= emails[i].split("@");

#### String second = "@"+s[1];

#### String first = s[0].split("\\+")[0]; // remove anyting after +

#### first=first.replace(".",""); // replace . with space before adding to set

#### // String final = first+second;

#### set.add(first+second);

#### }

#### return set.size();

#### }

#### *Sum numbers in a String :* i/p as34dfsdf12dfsdf45 o/p 34+12+45 = 91

#### static int findSum(String s) {

#### // holds sum of all numbers present in the string

#### int sum=0;

#### String temp="0";

#### for(int i=0;i<s.length();i++){

#### char ch = s.charAt(i);

#### if(Character.isDigit(ch))

#### {

#### temp = temp+ ch;

#### }

#### else{

#### sum = sum + Integer.parseInt(temp);

#### temp ="0";

#### }}

#### return sum + Integer.parseInt(temp);

#### }

***Reverse String word by word without using any inbuilt functions***

i/p This is my first round in Rupeek o/p sihT si ym tsrif dnuor ni keepuR

public String reverseWordByWord(String str){

int strLength = str.length()-1;

String reverse ="",temp ="";

for(int i=0;i<=strLength;i++)

{

temp= temp + str.charAt(i);

if((str.charAt(i)==' ')||(i==strLength) ){

for(int j=temp.length()-1;j>=0;j--)

{

reverse = reverse + temp.charAt(j);

if((j==0) && (i!=strLength))

reverse= reverse + " ";

}

temp ="";

}

}

return reverse;

}

***Reverse String keeping spaces intact***

i/p My name is Shailesh o/p hs elia hS siemanyM

static String reverseWithSpacesIntact(String S)

{

int start = 0;

int end = S.length() -1;

char[] str = S.toCharArray();

while (start<end){

if (str[start] ==' '){

start++;

continue ;

}

else if (str[end]==' '){

end --;

continue;

}

else{

char temp =str[start] ;

str[start] = str[end];

str[end] = temp;

start ++;

end --;

}

}

return String.valueOf(str);

}

***Count number of Alphabets and Integer in a String***

i/p "My birth comes on 9 August 1992" o/p digit = 5 , Character =20

for(int i=0;i<s.length();i++)

{

if(s.charAt(i)>=48 && s.charAt(i)<=57)

{

System.out.print("Digit is "+s.charAt(i));

digitCount++;

}

else if( (s.charAt(i)>=65 && s.charAt(i)<=90) || (s.charAt(i)>=97 && s.charAt(i)<=122)){

System.out.print("String is "+s.charAt(i));

stringCount++;

}

}

System.out.println("No of digits is "+digitCount);

for(int i=0;i<s.length();i++)

{

if(Character.isDigit(s.charAt(i)))

{

countdigit++;

}

if(Character.isAlphabetic(s.charAt(i)))

{

stringCount++;

}

}

System.out.println("No of digits is "+countdigit);

System.out.println("No of string is "+stringCount);

***String word with Frequency***

i/p aabbbcccddd o/p a2b3c3d2

String str1 = "aaaabbcbaccc";

StringBuilder sb = new StringBuilder();

// a2b3c3a2

int n = str1.length() - 1;

int count = 1;

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

if (str1.charAt(i + 1) == str1.charAt(i)) {

count++;

}

if (str1.charAt(i + 1) != str1.charAt(i)) {

sb.append(str1.charAt(i) + "" + count);

count = 1;

}

}

System.out.println(sb);

}

***Group Anagrams***

Input: strs = {"cat","ate","nat","bat"} Output: {{"bat"},{"nat","tan"},{"ate","eat","tea"}}

public List<List<String>> groupAnagrams(String[] strs) {

HashMap<String,List<String>> map=new HashMap<>();

for(int i=0;i<strs.length;i++){

String s1=strs[i];

char[] arr=s1.toCharArray();

Arrays.sort(arr);

String str=new String(arr);

if(map.containsKey(str)){

map.get(str).add(s1);

}else{

map.put(str,new ArrayList<>());

map.get(str).add(s1);

}

}

return new ArrayList<>(map.values());

}



