**最长连续公共子串**

public static int largest(String A, String B) {

int n = A.length();

int m = B.length();

int num = 0;

int[][] dp = new int[n + 1][m + 1]; // ///不赋值时的初始值为0

for (int i = 1; i <= n; i++) {

char c1 = A.charAt(i - 1);

for (int j = 1; j <= m; j++) {

char c2 = B.charAt(j - 1);

if (c1 == c2) {

dp[i][j] = dp[i - 1][j - 1] + 1;

} else {

dp[i][j] = 0;

}

num = Math.max(num, dp[i][j]);

}

}

return num;

}

**最长公共子串**

public static int findLCS(String A, int n, String B, int m) {

int[][] dp = new int[n + 1][m + 1];

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

dp[i][0] = 0;

}

for (int i = 0; i < dp[0].length; i++) {

dp[0][i] = 0;

}

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

for (int j = 1; j < dp[0].length; j++) {

if (A.charAt(i - 1) == B.charAt(j - 1)) {

dp[i][j] = dp[i - 1][j - 1] + 1;

} else {

dp[i][j] = Math.max(dp[i][j - 1], dp[i - 1][j]);

}

}

}

return dp[n][m];

}

**字符串转数字**

public static int stringToNum(String string){

int num=0;

int i=0;

boolean possitive=true;

if(string.length()==0){

return 0;

}

if(string.charAt(0)=='-'){

i=1;

possitive=false;

}

for(;i<string.length();i++){

int t=0;

t=string.charAt(i)-'0';

num=num\*10+t;

}

if (possitive==false) {

num\*=-1;

}

return num;

}

**找出字符串中出现的第一个不重复出现的字符串**

public class Solution {

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

List<Character> list = new LinkedList<>();

//Insert one char from stringstream

public void Insert(char ch)

{

if(map.containsKey(ch)){

map.put(ch,map.get(ch)+1);

}else{

map.put(ch,1);

}

list.add(ch);

}

public char FirstAppearingOnce()

{

char c='#';

for(int i=0;i<list.size();i++){

if(map.get(list.get(i))==1){

c=list.get(i);

break;

}

}

return c;

}

}

dd

gg

tt