67. Add Binary

EASY

Given two binary strings a and b, return *their sum as a binary string*.

A white background with black and white clouds

AI-generated content may be incorrect.

A white background with black and white clouds

AI-generated content may be incorrect.

**Constraints:**

* 1 <= a.length, b.length <= 104
* a and b consist only of '0' or '1' characters.
* Each string does not contain leading zeros except for the zero itself.

First Try: class Solution {

    private static final int zero = "0".codePointAt(0);

    public String addBinary(String a, String b) {

        int positionA = a.length()-1;

        int positionB = b.length()-1;

        int carry = 0; /\* holds the current carry value \*/

        StringBuffer result = new StringBuffer();

        while (positionA >= 0 || positionB >= 0) {

            int sum = carry;

            if (positionA >= 0) {

                sum += a.charAt(positionA) - zero;

                --positionA;

            }

            if (positionB >= 0) {

                sum += b.charAt(positionB) - zero;

                --positionB;

            }

            if (sum > 1) {

                carry = 1;

                sum -= 2;

                result.insert(0, sum);

            } else {

                carry = 0;

                result.insert(0, sum);

            }

        }

        if (carry != 0) {

            result.insert(0, carry);

        }

        return result.toString();

    }

}

A screenshot of a computer

AI-generated content may be incorrect.

I am not sure if the issue is using the StringBuffer. I will change to append everything and just reverse the results.

I decided to not use StringBuffer.insert

public class Solution {  
 private static final int zero = "0".codePointAt(0);  
  
 public String addBinary(String a, String b) {  
 int positionA = a.length()-1;  
 int positionB = b.length()-1;  
 int carry = 0; /\* holds the current carry value \*/

StringBuffer result = new StringBuffer();  
 while (positionA >= 0 || positionB >= 0) {  
 int sum = carry;  
 if (positionA >= 0) {  
 sum += a.charAt(positionA) - zero;  
 --positionA;  
 }  
 if (positionB >= 0) {  
 sum += b.charAt(positionB) - zero;  
 --positionB;  
 }  
 if (sum > 1) {  
 carry = 1;  
 sum -= 2;  
 result.append(sum);  
 } else {  
 carry = 0;  
 result.append(sum);  
 }  
 }  
 if (carry != 0) {  
 result.append(carry);  
 }  
 StringBuffer reverseStr = new StringBuffer();  
 for (int i = result.length()-1; i >= 0; i--) {  
 reverseStr.append(result.charAt(i));  
 }  
 return reverseStr.toString();  
  
 }

Result:

A screenshot of a phone

AI-generated content may be incorrect.

Next attempt is to avoid addition or anything by using two-dimensional arrays to hold the result + carry.

package LeetCode\_67**;**public class Solution3 {  
 private static int[][] *sumWithCarry0* = new int[**50**][**50**]**;** private final static int[][] *carryWithCarry0* = new int[**50**][**50**]**;** private static int[][] *sumWithCarry1* = new int[**50**][**50**]**;** private final static int[][] *carryWithCarry1* = new int[**50**][**50**]**;** public static void initializeArrays() {  
 *sumWithCarry0*[**48**][**48**] = **48;** *sumWithCarry0*[**48**][**49**] = **49;** *sumWithCarry0*[**49**][**48**] = **49;** *sumWithCarry0*[**49**][**49**] = **48;** *carryWithCarry0*[**48**][**48**] = **48;** *carryWithCarry0*[**48**][**49**] = **48;** *carryWithCarry0*[**49**][**48**] = **48;** *carryWithCarry0*[**49**][**49**] = **49;** *sumWithCarry1*[**48**][**48**] = **49;** *sumWithCarry1*[**48**][**49**] = **48;** *sumWithCarry1*[**49**][**48**] = **48;** *sumWithCarry1*[**49**][**49**] = **49;** *carryWithCarry1*[**48**][**48**] = **48;** *carryWithCarry1*[**48**][**49**] = **49;** *carryWithCarry1*[**49**][**48**] = **49;** *carryWithCarry1*[**49**][**49**] = **49;** }  
 public String addBinary(String *a***,** String *b*) {  
 int positionA = *a*.length()-**1;** int positionB = *b*.length()-**1;** char bitSum**;** char carry = '0'**;** /\* holds the current carry value \*/  
 StringBuffer *result* = new StringBuffer()**;** while (positionA >= **0** && positionB >= **0**) {  
 char *aChar* = *a*.charAt(positionA)**;** char *bChar* = *b*.charAt(positionB)**;** if (carry == '0') {  
 bitSum = (char) *sumWithCarry0*[*aChar*][*bChar*]**;** carry = (char) *carryWithCarry0*[*aChar*][*bChar*]**;** } else {  
 bitSum = (char) *sumWithCarry1*[*aChar*][*bChar*]**;** carry = (char) *carryWithCarry1*[*aChar*][*bChar*]**;** }  
 // Add bitSum to result  
 *result*.append(bitSum)**;** --positionA**;** --positionB**;** }  
 // We are done or need to process one string  
 if (positionA < **0** && positionB < **0**) {  
 // only need to process the carry  
 if (carry == '1') {  
 *result*.append("1")**;** }  
 } else if (positionA < **0**) {  
 // process b string where aChar is set to '0' or 48  
 while (positionB >= **0**) {  
 char *bChar* = *b*.charAt(positionB)**;** if (carry == '0') {  
 bitSum = (char) *sumWithCarry0*['0'][*bChar*]**;** carry = (char) *carryWithCarry0*['0'][*bChar*]**;** } else {  
 bitSum = (char) *sumWithCarry1*['0'][*bChar*]**;** carry = (char) *carryWithCarry1*['0'][*bChar*]**;** }  
 *result*.append(bitSum)**;** --positionB**;** }  
 // check carry here  
 if (carry == '1') {  
 *result*.append("1")**;** }  
 } else {  
 // process a string where bChar is set to '0' or 48  
 while (positionA >= **0**) {  
 char *aChar* = *a*.charAt(positionA)**;** if (carry == '0') {  
 bitSum = (char) *sumWithCarry0*[*aChar*]['0']**;** carry = (char) *carryWithCarry0*[*aChar*]['0']**;** } else {  
 bitSum = (char) *sumWithCarry1*[*aChar*]['0']**;** carry = (char) *carryWithCarry1*[*aChar*]['0']**;** }  
 *result*.append(bitSum)**;** --positionA**;** }  
 if (carry == '1') {  
 *result*.append("1")**;** }  
  
 }  
  
 // Reverse the results  
 StringBuffer *reverseStr* = new StringBuffer()**;** for (int i = *result*.length()-**1;** i >= **0;** i--) {  
 *reverseStr*.append(*result*.charAt(i))**;** }  
 return *reverseStr*.toString()**;** }  
 public static void main(String[] *args*) {  
 String *a* = "1010"**;** String *b* = "1011"**;** Solution *solution* = new Solution()**;** // "10101"  
 System.*out*.println("expectedResult: " + *solution*.addBinary(*a***,** *b*))**;** }  
}

