LIS

הסדרה העולה הארוכה ביותר

הבעיה: נמצאו אלגוריתם המחשב את הסדרה העולה הארוכה ביותר במערך הקלט ?

קלט: `{ 107 , 101 , 0 , 3 , 2 , 100 , 1 }` = arr

פלט: 5 (האורך המקסימלי של הסדרה העולה).

הסדרה : `{ 107 , 101 , 3 , 2 , 1 }`.

פתרון:

נמצאת סדרת המספרים :

1. שביר מערך באותו האורך [] (בו-נימב' - MAT) ו-[] (חב נימב' - D) .
2. נכניס את האיבר הראשון של המערך שקלטנו ל-2 המערכים החדשים .
3. נוסיף אינדקס - **סוף** ל-0 (כב' שבעת בכל רגע מה אורך המערך) .
4. נעבור על כל שאר האיברים ונשאל:
   א. מה מיקום איבר ה-I - ע"י חיפוש בינארי.
   ב. נכניס לאלכסון את איבר ה-I (ב-MAT).
   ג. נעבור על כל המספרים מ-0 עד לאינדקס -
   - נעתיק אותם משורה נמעלה לשורה החדשה (שורה אחת למטה).
   - נעתיק את איבר' האלכסון למערך D.
   ד. נכניס למקום האינדקס במערך D את האיבר במקום ה-I.
   ה. אם אינדקס גדול מהסוף אז תוסיף לסוף אחד.
5. שביר מערך בגודל סוף ונעתיק את השורה התחתונה ביותר שלו המשולש (במערך הבו-נימב') וזה יהיה סדרת המספרים העולה שחיפשנו.

סיבוכיות- סדרה :

$$O(n^2) = O(n * n) = O((n + \log n) * n)$$

<u>אלגוריתם שלב סברת המספרים (שלב בינארי - אלגו הסברה):</u>

1. שביר מערך באותו השלב [] (חצי מימד - D) .
2. נכניס את האיבר הראשון של המערך שקיטן ממערך החדש .
3. נוסיף אינדקס - סוף ל-0 (כי שבעת בכל רגע מה שלב המערך) .
4. נעבור עד כל שאר האיברים ונשאל:
   א. מה מיקום איבר ה-I - ע"י חיפוש בינארי.
   ב. אם האינדקס קטן או שווה לסוף
      אז D במקום האינדקס מקבל את איבר ה-I.

   אחרת

   - עוסיף לסוף אחד.
   - D במקום הסוף מקבל את איבר ה-I.
5. נחזיר (סוף + 1).

<u>סיבוכיות- שלב בינארי:</u>

O(n * log n)

<u>החיפוש הבינארי</u> - מקבל 3 פרמטרים:

1. מערך D.
2. עד איזה אינדקס לבדוק במערך (סוף).
3. ערך - מקום ה-I.

סיבוכיות: O(log n) .

קוד:

```
public class LongestIncreasingSequence {
        public static int binarySearchBetween(int []arr, int end, int value){
                // O( log n )
                // חיפוש בינארי
                int low = 0, high = end;
                if ( value < arr[0] ) // התחלה
                        return 0;
                if ( value > arr[end] ) // סוף
                        return end+1;
                while ( low <= high ){
                        int middle = (low + high)/2;
                        if ( low == high )
                                        return low;
                        else {
                                if ( arr[middle] == value )
                                        return middle;

                                if ( value < arr[middle] )
                                        high = middle;
                                else
                                        low = middle+1;
                        }
                }
                return -1;
        }
        public static int LISLength(int [] arr){
                // O( n * log n )
                // מערך הרצפה אלא את הרחבה מחיר הפונקציה
                int D[] = new int[arr.length];
                D[0] = arr[0];
                int end = 0;
                for ( int i=1; i<arr.length; i++ ){
                        int index = binarySearchBetween(D, end, arr[i]);
                        if ( index <= end )
                                D[index] = arr[i];
                        else{
                                end++;
                                D[end] = arr[i];
                        }
                }
                return end+1;
        }
```

```
public static int[] LIS(int [] arr){
        // O( n * n )
        // הפונקציה מחזירה את הסדרה הארוכה ביותר
        int MAT [][] = new int[arr.length][arr.length];
        int D[] = new int[arr.length];
        MAT[0][0] = arr[0];
        D[0] = arr[0];
        int end = 0;
        for ( int i=1; i<arr.length; i++ ){
                int index = binarySearchBetween(D, end, arr[i]);
                MAT[index][index] = arr[i];
                for( int j=0; j<index; j++ ){
                        MAT[index][j]=MAT[index-1][j];
                        D[j]=MAT[j][j];
                }
                D[index] = arr[i];
                if ( index > end )
                        end++;
                printIntMatrix(MAT);
                System.out.println();
        }
        int ans[] = new int[end+1];
        for( int j=0; j<=end; j++ )
                ans[j]=MAT[end][j];
        return ans;
}


public static void main(String[] args) {
        //int [] arr = randomIntArrayOfDiffNumbers(10);
        int [] arr ={ 1 , 100 , 2 , 3 , 0 , 101 , 107 };
        printIntegerArray(arr);
        System.out.println("m = " + LISLength(arr));
        int [] d = LIS(arr);
        printIntegerArray(d);
}
```

```
    // ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    // ~~~~~~~~~~~~~ פונקציות עזר - לא צריך לדעת למבחן ~~~~~~~~~~~~~~~~


    public static int [] randomIntArrayOfDiffNumbers(int size){
        int []arr = new int[size];
        for(int i=0; i<arr.length;){
            int randNumber = (int)(Math.random()*size)+1;
            if (!contains(arr,i-1,randNumber)){
                arr[i] = randNumber;
                i++;
            }
        }
        return arr;
    }

    public static boolean contains(int [] arr, int end, int value){
        boolean ans = false;
        for(int i=0; !ans &&i<=end; i++){
            if (arr[i]==value)
                ans = true;
        }
        return ans;
    }

    public static void printIntegerArray(int [] arr){
        for(int t=0; t<arr.length; t++){
            System.out.print(arr[t]+"\t");
        }
        System.out.println();
    }

    public static void printIntMatrix(int [][] mat){
        for(int i=0; i<mat.length; i++){
            for(int j=0; j<mat[0].length; j++){
                System.out.print(mat[i][j]+"\t");
            }
            System.out.println();
        }
    }
}
```