1187. Make Array Strictly Increasing

Hint ⊙

Hard   👍 1.5K   👎 28   ☆   ⟳

🔒 Companies

Given two integer arrays `arr1` and `arr2`, return the minimum number of operations (possibly zero) needed to make `arr1` strictly increasing.

In one operation, you can choose two indices `0 <= i < arr1.length` and `0 <= j < arr2.length` and do the assignment `arr1[i] = arr2[j]`.

If there is no way to make `arr1` strictly increasing, return `-1`.

Example 1:

```
Input: arr1 = [1,5,3,6,7], arr2 = [1,3,2,4]
Output: 1
Explanation: Replace 5 with 2, then arr1 = [1, 2, 3, 6, 7].
```

Example 2:

```
Input: arr1 = [1,5,3,6,7], arr2 = [4,3,1]
Output: 2
Explanation: Replace 5 with 3 and then replace 3 with 4. arr1
= [1, 3, 4, 6, 7].
```

Example 3:

```
Input: arr1 = [1,5,3,6,7], arr2 = [1,6,3,3]
Output: -1
Explanation: You can't make arr1 strictly increasing.
```
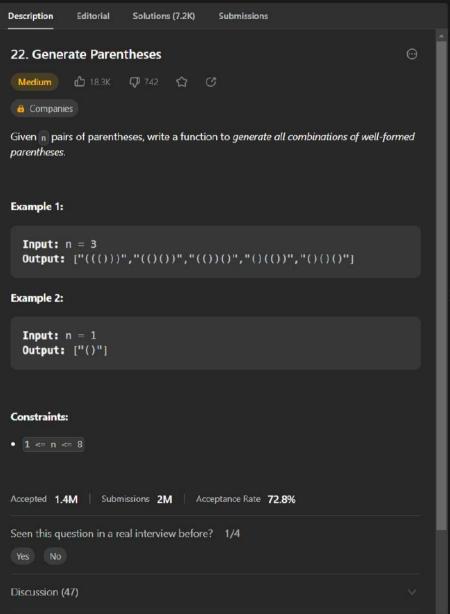
---

i Java ∨   • Auto   🔖 {} ⟲ ⌘ ⚙ ⤧

```java
class Solution {
    public int makeArrayIncreasing(int[] A, int[] B) { // A = arr1, B = arr2
        TreeSet<Integer> set = new TreeSet<>(Arrays.stream(B).boxed().toList());
        int[] dp = new int[B.length+1];
        dp[0]=-1;
        int INF = (int)2e9;
        for (int i = 0; i < A.length; i++){
            for (int j = B.length; j >= 0; j--){
                int a = A[i] > dp[j]? A[i] : INF; // option A - don't swap
                Integer b = set.higher(j==0?INF:dp[j-1]); // option B - swap
                dp[j]=Math.min(a, b==null?INF:b); // take the min of A and B
            }
        }
        for (int i = 0; i <= B.length; i++) if (dp[i] != INF){
            return i;
        }
        return -1;
```

Testcase   **Result**   🖫

Accepted   Runtime: 3 ms

• Case 1   • Case 2   • Case 3

Input

arr1 =
```
[1,5,3,6,7]
```

arr2 =
```
[1,3,2,4]
```

Output
```
1
```

Expected

Console ∨   🐞   Run   **Submit**

22. Generate Parentheses

Medium | 👍 18.3K | 👎 742 | ☆ | ↻

🔒 Companies

Given n pairs of parentheses, write a function to *generate all combinations of well-formed parentheses*.

Example 1:

```
Input: n = 3
Output: ["((()))","(()())","(())()","()(())","()()()"]
```

Example 2:

```
Input: n = 1
Output: ["()"]
```

Constraints:

- `1 <= n <= 8`

Accepted **1.4M** | Submissions **2M** | Acceptance Rate **72.8%**

Seen this question in a real interview before?   1/4

Yes | No

Discussion (47)   ⌄

---

i Java ⌄ | • Auto   🔖 {} ↻ ⌘ ⚙ ↙

```java
1   class Solution {
2       public List<String> generateParenthesis(int n) {
3           List<String> result = new ArrayList();
4           findAll("(",1,0,result,n);
5           return result;
6       }
7       void findAll(String current, int o, int c, List<String> result,int n){
8
9           if(current.length()==2*n){
10              result.add(current);
11              return;
12          }
13          if(o<n)findAll(current+"(",o+1,c,result,n);
14          if(c<o)findAll(current+")",o,c+1,result,n);
15      }
16  }
```

Testcase | **Result**   🖫

Accepted   Runtime: 0 ms

• Case 1 | • Case 2

Input

```
n =
3
```

Output

```
["((()))","(()())","(())()","()(())","()()()"]
```

Expected

```
["((()))","(()())","(())()","()(())","()()()"]
```

Console ⌄   🐞 Run Submit