

Find Anagram Mappings

Given two lists A and B , and B is an anagram of A . B is an anagram of A means B is made by randomizing the order of the elements in A .

We want to find an *index mapping* P , from A to B . A mapping $P[i] = j$ means the i th element in A appears in B at index j .

These lists A and B may contain duplicates. If there are multiple answers, output any of them.

For example, given

$A = [12, 28, 46, 32, 50]$

$B = [50, 12, 32, 46, 28]$

We should return

$[1, 4, 3, 2, 0]$

as $P[0] = 1$ because the 0th element of A appears at $B[1]$, and $P[1] = 4$ because the 1st element of A appears at $B[4]$, and so on.

Note:

1. A, B have equal lengths in range $[1, 100]$.
2. $A[i], B[i]$ are integers in range $[0, 10^5]$.

Solution 1

Other solution does not handle duplicates.

Like [12,12] and [12,12]. (This test case missing?) Description says duplicates are possible

So we should use ArrayList in the HashMap.

To clarify my confusion again.

if A[10,20,10], B[10,10,20], here multiple answer is [1,2,0] or [0,2,1].

I don't think we should do like [1,2,1] or [0,2,0]. But some people think is okay. But i think of this "B is formed by randomizing the A", which means all indices has to be used?

Let me know what you think?

```
public int[] anagramMappings(int[] A, int[] B) {
    int[] result = new int [A.length];
    Map<Integer, List<Integer>> map = new HashMap<>();
    for(int i = 0; i < B.length; i++) {
        map.computeIfAbsent(B[i], k -> new ArrayList<>()).add(i);
    }
    for(int i = 0; i < A.length; i++) {
        result[i] = map.get(A[i]).remove(map.get(A[i]).size()-1);
    }
    return result;
}
```

written by [wavy](#) original link [here](#)

Solution 2

O(N) solution using hashmap:

```
def anagramMappings(self, A, B):  
    d = {}  
    for i, b in enumerate(B):  
        if b not in d:  
            d[b] = []  
        d[b].append(i)  
    return [d[a].pop() for a in A]
```

2 lines accepted solution, but it can't return all index in case of duplicates.

```
def anagramMappings(self, A, B):  
    d = {b:i for i,b in enumerate(B)}  
    return [d[a] for a in A]
```

O(N²) solution in 1 line:

```
def anagramMappings(self, A, B):  
    return [B.index(a) for a in A]
```

written by [lee215](#) original link [here](#)

Solution 3

```
class Solution {  
    public int[] anagramMappings(int[] A, int[] B) {  
        int l = A.length;  
        int[] res = new int[l];  
        HashMap<Integer,Integer> map = new HashMap<>();  
        for(int i=0;i<l;i++){  
            map.put(B[i],i);  
        }  
        for(int i=0;i<l;i++){  
            res[i] = map.get(A[i]);  
        }  
        return res;  
    }  
}
```

written by [ashish53v](#) original link [here](#)

From [Leetcode](#).