

题目描述

给定两个数组a, b, 若 $a[i] == b[j]$ 则称 $[i, j]$ 为一个二元组^Q, 求在给定的两个数组中, 二元组的个数。

输入描述

第一行输入 m
第二行输入m个数, 表示第一个数组

第三行输入 n
第四行输入n个数, 表示第二个数组

输出描述

二元组个数。

用例

输入	4 1 2 3 4 1 1
输出	1
说明	二元组个数为 1个

输入	6 1 1 2 2 4 5 3 2 2 4
输出	5
说明	二元组个数为 5个。

题目解析

很简单的双重for,

```
1  /**
2   *
3   * @param {Array} arrM 第二行输入的数组
4   * @param {Number} m 第一行输入的数字m
5   * @param {Array} arrN 第四行输入的数组
6   * @param {Number} n 第二行输入的数字n
7   * @returns
8   */
9  function getResult(arrM, m, arrN, n) {
10     let count = 0;
11
12     for (let i = 0; i < m; i++) {
13         for (let j = 0; j < n; j++) {
14             if (arrM[i] === arrN[j]) {
15                 count++;
16             }
17         }
18     }
19
20     return count;
21 }
```

但是不知道数量级多少, 如果数量级比较大的话, 则 $O(n*m)$ 可能罩不住。

因此, 我们还需要考虑下大数量级的情况。

我的思路如下:

先找出m数组中, 在n数组中出现的数及个数, 在找出n数组中, 在m数组中出现的数及个数, 比如:

用例2中

题目解析

很简单双重for,

```
1  /**
2   *
3   * @param (Array) arrM 第二行输入的数组
4   * @param (Number) m 第一行输入的数字m
5   * @param (Array) arrN 第四行输入的数组
6   * @param (Number) n 第二行输入的数字n
7   * @returns
8   */
9  function getResult(arrM, m, arrN, n) {
10     let count = 0;
11
12     for (let i = 0; i < m; i++) {
13         for (let j = 0; j < n; j++) {
14             if (arrM[i] === arrN[j]) {
15                 count++;
16             }
17         }
18     }
19
20     return count;
21 }
```

但是不知道数量级多少，如果数量级比较大的话，则 $O(n*m)$ 可能罩不住。

因此，我们还需要考虑下大数量级的情况。

我的思路如下：

先找出m数组中，在n数组中出现的数及个数，在找出n数组中，在m数组中出现的数及个数，比如：

用例2中

countM = {2:2, 4:1}


countN = {2:2, 4:1}

然后将相同数的个数相乘，最后求和即为题解，比如 $2*2 + 1*1 = 5$

JavaScript算法源码

```
1  /* JavaScript Node ACM模式 控制台输入获取 */
2  const readline = require("readline");
3
4  const rl = readline.createInterface({
5      input: process.stdin,
6      output: process.stdout,
7  });
8
9  const lines = [];
10 rl.on("line", (line) => {
11     lines.push(line);
12
13     if (lines.length === 4) {
14         const m = lines[0] - 0;
15         const arrM = lines[1].split(" ").map(Number);
16
17         const n = lines[2] - 0;
18         const arrN = lines[3].split(" ").map(Number);
19
20         console.log(getResult(arrM, m, arrN, n));
21
22         lines.length = 0;
23     }
24 });
25
26 function getResult(arrM, m, arrN, n) {
27     const setM = new Set(arrM);
28     const setN = new Set(arrN);
29
30     const countM = {};
31     for (let m of arrM) {
32         if (setN.has(m)) countM[m] ? countM[m]++ : (countM[m] = 1);
33     }
34
35     const countN = {};
36     for (let n of arrN) {
37         if (setM.has(n)) countN[n] ? countN[n]++ : (countN[n] = 1);
38     }
39
40     let count = 0;
41     for (let k in countM) {
42         count += countM[k] * countN[k];
43     }
44
45     return count;
46 }
```

Java算法源码

 伏城之外 [已关注](#)

👍 0

💬 2

🔖 4

📁 4

[专栏目录](#)

[已订阅](#)

JavaScript算法源码

```
1  /* JavaScript Node ACM模式 控制台输入获取 */
2  const readline = require("readline");
3
4  const rl = readline.createInterface({
5    input: process.stdin,
6    output: process.stdout,
7  });
8
9  const lines = [];
10 rl.on("line", (line) => {
11   lines.push(line);
12
13   if (lines.length === 4) {
14     const m = lines[0] - 0;
15     const arrM = lines[1].split(" ").map(Number);
16
17     const n = lines[2] - 0;
18     const arrN = lines[3].split(" ").map(Number);
19
20     console.log(getResult(arrM, m, arrN, n));
21
22     lines.length = 0;
23   }
24 });
25
26 function getResult(arrM, m, arrN, n) {
27   const setM = new Set(arrM);
28   const setN = new Set(arrN);
29
30   const countM = {};
31   for (let m of arrM) {
32     if (setN.has(m)) countM[m] ? countM[m]++ : (countM[m] = 1);
33   }
34
35   const countN = {};
36   for (let n of arrN) {
37     if (setM.has(n)) countN[n] ? countN[n]++ : (countN[n] = 1);
38   }
39
40   let count = 0;
41   for (let k in countM) {
42     count += countM[k] * countN[k];
43   }
44
45   return count;
46 }
```

Java算法源码

```
1  import java.util.ArrayList;
2  import java.util.HashMap;
3  import java.util.HashSet;
4  import java.util.Scanner;
5
6  public class Main {
7    public static void main(String[] args) {
8      Scanner sc = new Scanner(System.in);
9
10     int m = sc.nextInt();
11     ArrayList<Integer> listM = new ArrayList<>();
12     for (int i = 0; i < m; i++) {
13       listM.add(sc.nextInt());
14     }
15
16     int n = sc.nextInt();
17     ArrayList<Integer> listN = new ArrayList<>();
18     for (int i = 0; i < n; i++) {
19       listN.add(sc.nextInt());
20     }
21
22     System.out.println(getResult(listM, listN));
23   }
24
25   public static int getResult(ArrayList<Integer> listM, ArrayList<Integer> listN) {
26     HashSet<Integer> setM = new HashSet<Integer>(listM);
27     HashSet<Integer> setN = new HashSet<Integer>(listN);
28
29     HashMap<Integer, Integer> countM = new HashMap<>();
30     for (Integer m : listM) {
31       if (setN.contains(m)) {
32         countM.put(m, countM.getOrDefault(m, 0) + 1);
33       }
34     }
35
36     HashMap<Integer, Integer> countN = new HashMap<>();
37     for (Integer n : listN) {
38       if (setM.contains(n)) {
39         countN.put(n, countN.getOrDefault(n, 0) + 1);
40       }
41     }
42   }
```

Java算法源码

```
1 import java.util.ArrayList;
2 import java.util.HashMap;
3 import java.util.HashSet;
4 import java.util.Scanner;
5
6 public class Main {
7     public static void main(String[] args) {
8         Scanner sc = new Scanner(System.in);
9
10        int m = sc.nextInt();
11        ArrayList<Integer> listM = new ArrayList<>();
12        for (int i = 0; i < m; i++) {
13            listM.add(sc.nextInt());
14        }
15
16        int n = sc.nextInt();
17        ArrayList<Integer> listN = new ArrayList<>();
18        for (int i = 0; i < n; i++) {
19            listN.add(sc.nextInt());
20        }
21
22        System.out.println(getResult(listM, listN));
23    }
24
25    public static int getResult(ArrayList<Integer> listM, ArrayList<Integer> listN) {
26        HashSet<Integer> setM = new HashSet<Integer>(listM);
27        HashSet<Integer> setN = new HashSet<Integer>(listN);
28
29        HashMap<Integer, Integer> countM = new HashMap<>();
30        for (Integer m : listM) {
31            if (setN.contains(m)) {
32                countM.put(m, countM.getOrDefault(m, 0) + 1);
33            }
34        }
35
36        HashMap<Integer, Integer> countN = new HashMap<>();
37        for (Integer n : listN) {
38            if (setM.contains(n)) {
39                countN.put(n, countN.getOrDefault(n, 0) + 1);
40            }
41        }
42
43        int count = 0;
44        for (Integer k : countM.keySet()) {
45            count += countM.get(k) * countN.get(k);
46        }
47
48        return count;
49    }
50 }
```

Python算法源码

```
1 # 输入读取
2 m = int(input())
3 arrM = list(map(int, input().split()))
4
5 n = int(input())
6 arrN = list(map(int, input().split()))
7
8
9 # 算法入口
10 def getResult(arrM, arrN):
11     setM = set(arrM)
12     setN = set(arrN)
13
14     countM = {}
15     for m in arrM:
16         if m in setN:
17             if countM.get(m) is None:
18                 countM[m] = 1
19             else:
20                 countM[m] += 1
21
22     countN = {}
23     for n in arrN:
24         if n in setM:
25             if countN.get(n) is None:
26                 countN[n] = 1
27             else:
28                 countN[n] += 1
29
30     count = 0
31     for k in countM.keys():
32         count += countM[k] * countN[k]
33
34     return count
35
36
37 # 算法调用
38 print(getResult(arrM, arrN))
```

Python算法源码

```
1 # 输入获取
2 m = int(input())
3 arrM = list(map(int, input().split()))
4
5 n = int(input())
6 arrN = list(map(int, input().split()))
7
8
9 # 算法入口
10 def getResult(arrM, arrN):
11     setM = set(arrM)
12     setN = set(arrN)
13
14     countM = {}
15     for m in arrM:
16         if m in setN:
17             if countM.get(m) is None:
18                 countM[m] = 1
19             else:
20                 countM[m] += 1
21
22     countN = {}
23     for n in arrN:
24         if n in setM:
25             if countN.get(n) is None:
26                 countN[n] = 1
27             else:
28                 countN[n] += 1
29
30     count = 0
31     for k in countM.keys():
32         count += countM[k] * countN[k]
33
34     return count
35
36
37 # 算法调用
38 print(getResult(arrM, arrN))
```

[复制](#)