

题目描述

在学校中，N个小朋友站成一队，第i个小朋友的身高为height[i]，
第i个小朋友可以看到的第一个比自己身高更高的小朋友j，那么j是i的好朋友(要求j > i)。
请重新生成一个列表，对应位置的输出是每个小朋友的好朋友位置，如果没有看到好朋友，请在该位置用0代替。
小朋友人数范围是 [0, 40000]。

输入描述

第一行输入N，N表示有N个小朋友
第二行输入N个小朋友的身高height[i]，都是整数

输出描述

输出N个小朋友的好朋友的位置

用例

输入	2 100 95
输出	0 0
说明	第一个小朋友身高100，站在队尾位置，向队首看，没有比他身高高的小朋友，所以输出第一个值为0。 第二个小朋友站在队首，前面也没有比他身高高的小朋友，所以输出第二个值为0。

输入	8 123 124 125 121 119 122 126 123
输出	1 2 6 5 5 6 0 0
说明	123的好朋友是1位置上的124 124的好朋友是2位置上的125 125的好朋友是6位置上的126 以此类推

题目解析

感觉很简单，直接双重for，逻辑如下

```
1  /* 记录前面出现的第一个比自己高的人的序号 */
2  function getHigherIndex(arr) {
3      let higherIndex = arr.slice().fill(0);
4      for (let i = 0; i < arr.length - 1; i++) {
5          for (let j = i + 1; j < arr.length; j++) {
6              if (arr[j] > arr[i]) {
7                  higherIndex[i] = j;
8                  break;
9              }
10         }
11     }
12     return higherIndex;
13 }
```

但是时间复杂度是 $O(n^2)$ 。那么是不是有优化的可能呢？

下面图示中，我们蓝色的是*i*指针，橙色的是*j*指针，我们可以发现*j*指针的扫描过程存在重复，比如：

- *i*=2时，*j*从121扫描到了126，将扫描过程中*j*指向的每一个数都和*i*指向的125进行了比较。
- *i*=3时，*j*又扫描了一遍119，122；
- *i*=4时，*j*又扫描了一遍122
- *i*=5时，*j*又扫描了一遍126

那么如何才能避免重复扫描呢？



此时我们可以利用栈结构

for循环遍历输入数组的每一个元素，并将遍历出来的 [元素值,索引位置] 尝试压入stack栈中。

如果栈顶没有元素，则直接压入。



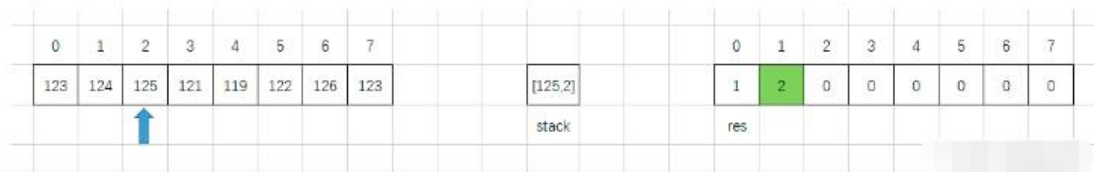
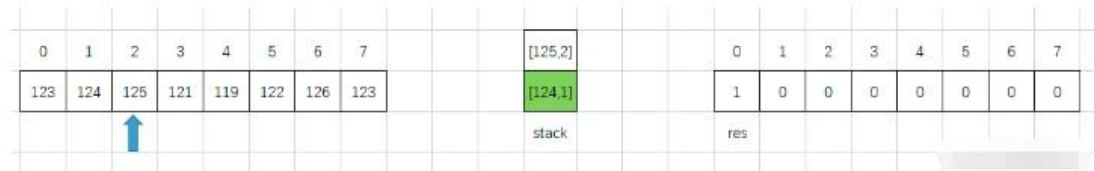
如果栈顶有元素，比如下面图示情况，则比较遍历出来的元素 124 和 stack 栈顶元素 123 的大小，如果遍历元素值较大，



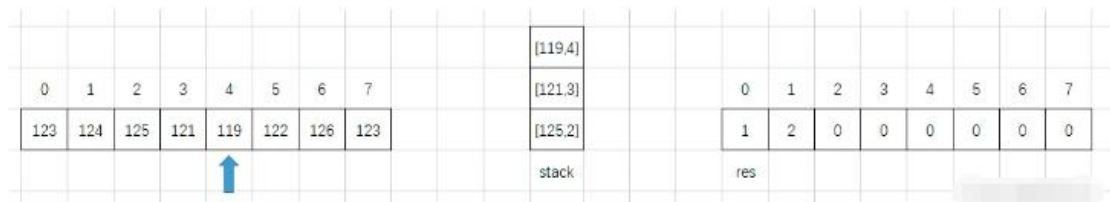
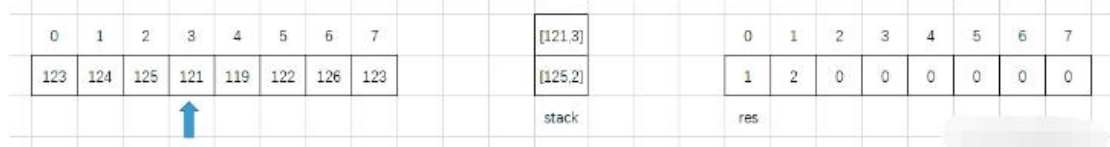
则将 124 的索引位置录入 res[123 的索引位置]，并将栈顶 123 弹栈后，压入 124



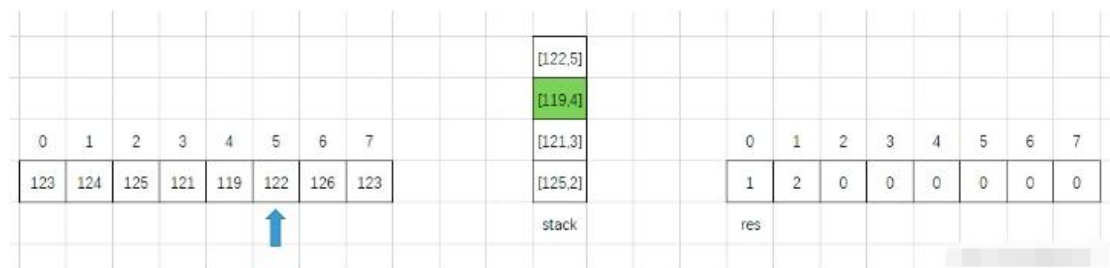
下面操作同理



如果遍历的元素值小于栈顶的元素值，则继续下次遍历，如下面例子 121 小于 125，则正常压入后，继续下次遍历



如果遍历的元素 122 比栈顶元素 119 大，则栈顶元素 119 弹栈



0	1	2	3	4	5	6	7
123	124	125	121	119	122	126	123

↑

[122,5]
[121,3]
[125,2]

stack

0	1	2	3	4	5	6	7
1	2	0	0	5	0	0	0

res

Diagram illustrating the state of the array and stack after the third iteration:

- Array: [123, 124, 125, 121, 119, 122, 126, 123]
- Stack: [122, 5], [121, 3], [125, 2]
- Result Array (res): [1, 2, 0, 0, 5, 0, 0, 0]

A blue arrow points to the element 122 at index 5 in the array.

0	1	2	3	4	5	6	7
123	124	125	121	119	122	126	123

↑

[126,6]
[122,5]
[125,2]

stack

0	1	2	3	4	5	6	7
1	2	0	5	5	0	0	0

res

[illegible]

	0	1	2	3	4	5	6	7											
	123	124	125	121	119	122	126	123											

[126,6]
[125,2]
stack

0	1	2	3	4	5	6	7
1	2	0	5	5	6	0	0
res							

0	1	2	3	4	5	6	7
123	124	125	121	119	122	126	123

↑

[126,6]

stack

0	1	2	3	4	5	6	7
1	2	6	5	5	6	0	0

res

	0	1	2	3	4	5	6	7											
	123	124	125	121	119	122	126	123											

↑

[123,7]																			
[126,6]																			
stack																			

	0	1	2	3	4	5	6	7											
	1	2	6	5	5	6	0	0											

res

Java算法源码

```
1 import java.util.LinkedList;
2 import java.util.Scanner;
3 import java.util.StringJoiner;
4
5 public class Main {
6     public static void main(String[] args) {
7         Scanner sc = new Scanner(System.in);
8
9         int n = sc.nextInt();
10
11         int[] arr = new int[n];
12         for (int i = 0; i < n; i++) arr[i] = sc.nextInt();
13
14         System.out.println(getResult(arr));
15     }
16
17     public static String getResult(int[] arr) {
18         LinkedList<int[]> stack = new LinkedList<>();
19
20         int len = arr.length;
21         int[] res = new int[len];
22
23         for (int i = 0; i < len; i++) {
24             int ele = arr[i];
25
26             while (true) {
27                 if (stack.size() == 0) {
28                     stack.add(new int[] {ele, i});
29                     break;
30                 }
31             }
```

```

32     int[] peek = stack.getLast();
33     int peekEle = peek[0];
34     int peekIndex = peek[1];
35
36     if (ele > peekEle) {
37         res[peekIndex] = i;
38         stack.removeLast();
39     } else {
40         stack.add(new int[] {ele, i});
41         break;
42     }
43 }
44 }
45
46 StringJoiner sj = new StringJoiner(" ");
47 for (int v : res) {
48     sj.add(v + "");
49 }
50
51 return sj.toString();
52 }
53 }

```

JS算法源码

```

1  /* JavaScript Node ACN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 === 2) {
14         let n = lines[0];
15         let arr = lines[1]
16             .split(" ")
17             .slice(0, n)
18             .map((ele) => parseInt(ele));
19
20         console.log(getHigherIndex(arr).join(" "));
21
22         lines.length = 0;
23     }
24 });
25

```

```
26  /* 记录第一眼看到比自己高的的人的序号 */
27  function getHigherIndex(arr) {
28      let stack = [];
29
30      let len = arr.length;
31      let res = new Array(len).fill(0);
32
33      for (let i = 0; i < len; i++) {
34          let ele = arr[i];
35          let index = i;
36
37          while (true) {
38              if (stack.length === 0) {
39                  stack.push([ele, index]);
40                  break;
41              }
42
43              let [peekEle, peekIndex] = stack[stack.length - 1];
44
45              if (ele > peekEle) {
46                  res[peekIndex] = index;
47                  stack.pop();
48              } else {
49                  stack.push([ele, index]);
50                  break;
51              }
52          }
53      }
54
55      return res;
56  }
```


Python算法源码

```
1  # 输入获取
2  n = int(input())
3  arr = list(map(int, input().split()))
4
5
6  # 算法入口
7  def getResult():
8      stack = []
9
10     res = [0]*(len(arr))
11
12     for i in range(len(arr)):
13         ele = arr[i]
14
15         while True:
16             if len(stack) == 0:
17                 stack.append([ele, i])
18                 break
19
20             peekEle, peekIndex = stack[-1]
21
22             if ele > peekEle:
23                 res[peekIndex] = i
24                 stack.pop()
25             else:
26                 stack.append([ele, i])
27                 break
28
29     return " ".join(map(str, res))
30
31
32 # 算法调用
33 print(getResult())
```