

LintCode领扣题解 (/problem) / 整数排序 · Sort Integers

## 整数排序 · Sort Integers

中文

LintCode 版权所有 (/problem/?tags=lintcode-copyright)

排序 (/problem/?tags=sort)

### 描述

给一组整数，按照升序排序。使用选择排序，冒泡排序，插入排序或者任何  $O(n^2)$  的排序算法。

### 样例

样例 1:

输入: [3, 2, 1, 4, 5]

输出: [1, 2, 3, 4, 5]

样例解释:

返回排序后的数组。

样例 2:

输入: [1, 1, 2, 1, 1]

输出: [1, 1, 1, 1, 2]

样例解释:

返回排好序的数组。

在线评测地址: <https://www.lintcode.com/problem/sort-integers/> (<https://www.lintcode.com/problem/sort-integers/>)

收起题目描述 ^

语言类型 [ALL \(15\)](#) [python \(7\)](#) [java \(5\)](#) [cpp \(2\)](#) [javascript \(1\)](#)[上传题解](#)

令狐冲  
更新于 6/9/2020, 7:03:48 AM

几种  $O(n^2)$  时间复杂度的排序算法。

```
/**
 * 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - 面试/Resume / Project 2020版
 * - Design类课程包括：系统设计 System Design，面向对象设计 OOD
 * - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
 * - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
 */
//选择排序
public class Solution {
    /*
     * @param A: an integer array
     * @return:
     */
    public void sortIntegers(int[] A) {
        // write your code here
    }
}
```

```
        for (int i = 0; i < A.length; i++) {
            int minIdx = i;
            for (int j = i; j < A.length; j++) {
                if (A[j] < A[minIdx]) {
                    minIdx = j;
                }
            }
            int tmp = A[i];
            A[i] = A[minIdx];
            A[minIdx] = tmp;
        }
    }
}

//选择排序2
public class Solution {
    /*
     * @param A: an integer array
     * @return:
     */
    public void sortIntegers(int[] A) {
        // write your code here
        for (int i = 0; i < A.length; i++) {
            for (int j = i + 1; j < A.length; j++) {
                if (A[i] > A[j]) {
                    int tmp = A[i];
                    A[i] = A[j];
                    A[j] = tmp;
                }
            }
        }
    }
}

//插入排序
public class Solution {
    /*
     * @param A: an integer array
     * @return:
     */
    public void sortIntegers(int[] A) {
        // write your code here
        for (int i = 0; i < A.length; i++) {
            int newVal = A[i];
            int j = i - 1;

            while (j >= 0 && A[j] > newVal) {
                A[j + 1] = A[j];
                j--;
            }
            A[j + 1] = newVal;
        }
    }
}

//冒泡排序
public class Solution {
    /*
     * @param A: an integer array
     * @return:
     */
    public void sortIntegers(int[] A) {
        // write your code here
        while (true) {
            boolean exchange = false;
            for (int i = 0; i < A.length - 1; i++) {
                if (A[i] > A[i + 1]) {
```

```
        int tmp = A[i];
        A[i] = A[i + 1];
        A[i + 1] = tmp;
        exchange = true;
    }
}
if (!exchange) {
    break;
}
}
```

👍 获赞 3

💬 添加评论

# 你的口袋题库

## 2000+算法真题、国内外名企题库免费开放



九章算法APP



令狐冲

更新于 7/6/2020, 2:24:15 PM

补充了各个算法的实现

```

/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm_source=code
 */
class Solution:
    # @param {int[]} A an integer array
    # @return nothing
    def sortIntegers(self, A):
        # Write your code here
        A.sort()

# 选择排序
class Solution:
    # @param {int[]} A an integer array
    # @return nothing
    def sortIntegers(self, A):
        # Write your code here
        for i in range(0, len(A)):
            t = i
            for j in range(i + 1, len(A)):
                if A[j] < A[t]:
                    t = j
            A[i], A[t] = A[t], A[i]

# 插入排序
class Solution:
    # @param {int[]} A an integer array
    # @return nothing
    def sortIntegers(self, A):
        # Write your code here
        for i in range(1, len(A) + 1):
            for j in range(i - 1, 0, -1):
                if A[j - 1] > A[j]:
                    A[j - 1], A[j] = A[j], A[j - 1]
            else:
                break

# 冒泡排序
class Solution:
    # @param {int[]} A an integer array
    # @return nothing
    def sortIntegers(self, A):
        # Write your code here
        for i in range(0, len(A) - 1):
            for j in range(0, len(A) - 1):
                if A[j] > A[j + 1]:
                    A[j], A[j + 1] = A[j + 1], A[j]

```

👍 获赞 0

💬 8 条评论



令狐冲

更新于 6/9/2020, 7:04:30 AM

这里直接调用了algorithm中的排序库函数, 默认是升序排序。

```

/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。

```

```

* - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ /
Resume / Project 2020版
* - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
* - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
* - 更多详情请见官方网站: http://www.jiuzhang.com/?utm\_source=code
*/
class Solution {
public:
    /**
     * @param A an integer array
     * @return void
     */
    void sortIntegers(vector<int>& A) {
        // Write your code here
        sort(A.begin(), A.end());
    }
};

/*
*      选择排序
*/
class Solution {
public:
    /**
     * @param A an integer array
     * @return void
     */
    void sortIntegers(vector<int>& A) {
        // Write your code here
        for(int i = 0; i < A.size(); i++) {
            int t = i;
            for(int j = i + 1; j < A.size(); j++) {
                if(A[j] < A[t]) t = j;
            }
            swap(A[i], A[t]);
        }
    }
};

/*
*      插入排序
*/
class Solution {
public:
    /**
     * @param A an integer array
     * @return void
     */
    void sortIntegers(vector<int>& A) {
        // Write your code here
        for(int i = 1; i <= A.size(); i++) {
            for(int j = i - 1; j > 0; j--) {
                if(A[j-1] > A[j]) {
                    swap(A[j-1], A[j]);
                }
                else
                    break;
            }
        }
    }
};

/*
*      冒泡排序
*/
class Solution {
public:
    /**

```

```

    * @param A an integer array
    * @return void
    */
void sortIntegers(vector<int>& A) {
    // Write your code here
    for(int i = 0; i < A.size() - 1; i++) {
        for(int j = 0; j < A.size() - 1; j++) {
            if(A[j] > A[j + 1]) swap(A[j], A[j + 1]);
        }
    }
}
};

```

👍 获赞 0    💬 添加评论



九章用户GJ3OSM

更新于 12/16/2020, 2:34:21 AM

十大排序算法实现 冒泡排序、选择排序、插入排序、希尔排序、归并排序、快速排序、堆排序、计数排序、桶排序、基数排序（适用于正数排序）

```

/**
 * 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括：系统设计 System Design，面向对象设计 OOD
 * - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
 * - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
 */
class Solution:
    """
    @param A: an integer array
    @return: nothing
    """
    def sortIntegers(self, A):
        # write your code here
        if A == None or len(A) == 0:
            return
        A.sort()

# Bubble Sort
class Solution:
    def sortIntegers(self, A):
        if A == None or len(A) == 0:
            return

        for i in range(len(A) - 1):
            for j in range(len(A) - 1 - i):
                if A[j] > A[j + 1]:
                    A[j], A[j + 1] = A[j + 1], A[j]

# Selection Sort
class Solution:
    def sortIntegers(self, A):
        if A == None or len(A) == 0:
            return
        for i in range(len(A)):
            tmp = i
            for j in range(i + 1, len(A)):
                if A[j] < A[tmp]:
                    tmp = j
            A[i], A[tmp] = A[tmp], A[i]

# Insertion Sort

```

```
class Solution:
    def sortIntegers(self, A):
        if A == None or len(A) == 0:
            return
        for i in range(1, len(A)):
            for j in range(i - 1, -1, -1):
                if A[i] < A[j]:
                    A[i], A[j] = A[j], A[i]
                    i = j
            else:
                break

# Shell Sort
class Solution:
    def sortIntegers(self, A):
        if A == None or len(A) == 0:
            return
        dk = len(A) // 2
        while dk >= 1:
            self.InsertSort(A, dk)
            dk = dk // 2

    def InsertSort(self, A, dk):
        for i in range(dk):
            for j in range(i, len(A), dk):
                for k in range(j - 1, -1, -dk):
                    if A[j] < A[k]:
                        A[j], A[k] = A[k], A[j]
                        j = k
                else:
                    break

# Merge Sort
class Solution:
    def sortIntegers(self, A):
        if A == None or len(A) == 0:
            return
        tmp = [0] * len(A)
        self.mergeSort(A, 0, len(A) - 1, tmp)

    def mergeSort(self, A, start, end, tmp):
        if start >= end:
            return

        self.mergeSort(A, start, (start + end) // 2, tmp)
        self.mergeSort(A, (start + end) // 2 + 1, end, tmp)

        self.merge(A, start, end, tmp)

    def merge(self, A, start, end, tmp):
        mid = (start + end) // 2
        leftIndex = start
        rightIndex = mid + 1
        index = leftIndex

        while leftIndex <= mid and rightIndex <= end:
            if A[leftIndex] < A[rightIndex]:
                tmp[index] = A[leftIndex]
                leftIndex += 1
            else:
                tmp[index] = A[rightIndex]
                rightIndex += 1
            index += 1

        while leftIndex <= mid:
            tmp[index] = A[leftIndex]
            index += 1
```

```
        leftIndex += 1
    while rightIndex <= end:
        tmp[index] = A[rightIndex]
        index += 1
        rightIndex += 1

    for index in range(start, end + 1):
        A[index] = tmp[index]

# Quick Sort
class Solution:
    def sortIntegers(self, A):
        if A == None or len(A) == 0:
            return
        self.quickSort(A, 0, len(A) - 1)

    def quickSort(self, A, start, end):
        if start >= end:
            return

        left, right = start, end
        pivot = A[(start+end)//2]

        while left <= right:
            while left <= right and A[left] < pivot:
                left += 1
            while left <= right and A[right] > pivot:
                right -= 1

            if left <= right:
                A[left], A[right] = A[right], A[left]
                left += 1
                right -= 1

        self.quickSort(A, start, right)
        self.quickSort(A, left, end)

# Heap Sort
class Solution:
    def sortIntegers(self, A):
        if A == None or len(A) == 0:
            return
        A.insert(0, 0)

        lenA = len(A) - 1
        dk = lenA // 2
        for i in range(dk):
            self.heapAdjust(A, dk - i, lenA)

        for i in range(lenA - 1):
            A[1], A[lenA - i] = A[lenA - i], A[1]
            self.heapAdjust(A, 1, lenA - i - 1)

        A.remove(0)

    def heapAdjust(self, A, start, end):
        tmp = A[start]

        i = start
        j = 2 * i

        while j <= end:
            if j < end and A[j] < A[j + 1]:
                j += 1
            if tmp < A[j]:
                A[i] = A[j]
                i = j
                j = 2 * i
```



```
        else:
            break

    A[i] = tmp

# Counting Sort
class Solution:
    def sortIntegers(self, A):
        if A == None or len(A) == 0:
            return

        minA = min(A)
        maxA = max(A)

        L = [0] * (maxA - minA + 1)

        for i in range(len(A)):
            L[A[i] - minA] += 1

        index = 0
        for i in range(len(L)):
            while L[i] > 0:
                A[index] = minA + i
                index += 1
                L[i] -= 1

# Bucket Sort
class Solution:
    def sortIntegers(self, A):
        if A == None or len(A) == 0:
            return

        n = 100
        new_list = [[] for _ in range(n)]
        minA = min(A)

        for data in A:
            index = (data - minA) // 100
            new_list[index].append(data)

        for i in range(n):
            new_list[i].sort()

        index = 0
        for i in range(n):
            for j in range(len(new_list[i])):
                A[index] = new_list[i][j]
                index += 1

# Radix Sort
# Suitable for positive sorting
class Solution:
    def sortIntegers(self, A):
        if A == None or len(A) == 0:
            return

        maxA = max(A)

        d = 0
        while maxA // 10 > 0:
            d += 1
            maxA = maxA // 10
        d += 1

        for i in range(d):
            s = [[] for _ in range(10)]
            for data in A:
                s[data // (10 ** i) % 10].append(data)
```

```
index = 0
for j in s:
    for k in j:
        A[index] = k
        index += 1
```

👍 获赞 4    💬 添加评论



**Gwendolyn**

更新于 6/9/2020, 7:03:47 AM

Quick Sort 的标准写法 python 版。单独写了partition的函数。增强可读性。pivot用序列中的第一个。第15行。每一次sort后, 要把pivot放回到中间。第27行。最坏的极端情况是 $O(n^2)$ 。i.e.单调递减序列。

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm_source=code
 */
def sortIntegers(self, A):
    # write your code here
    self.quickSort(A, 0, len(A)-1)

    def quickSort(self, A, start, end):
        if start >= end:
            return

        splitpoint = self.partition(A, start, end)
        self.quickSort(A, start, splitpoint-1)
        self.quickSort(A, splitpoint+1, end)

    def partition(self, A, start, end):

        pivot = A[start]
        left, right = start+1, end

        while left <= right:
            while left <= right and A[left] <= pivot:
                left +=1
            while left <= right and A[right] >= pivot:
                right -=1

            if left <= right:
                A[left], A[right] = A[right], A[left]

        A[start], A[right] = A[right], A[start]

        return right
```

👍 获赞 4    💬 添加评论



**Jason Chen**

更新于 6/9/2020, 7:03:47 AM

九章官方的插入排序应该是pass不了: 我自己写了新的, 希望能帮助到理解

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ /
Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm\_source=code
 */
def sortIntegers(A):
    for i in range(1, len(A)):
        for j in range(i - 1, -1, -1):
            if A[i] < A[j]:
                A[i], A[j] = A[j], A[i]
                i = j
            else:
                break
    return A
```

👍 获赞 3

💬 添加评论



**Chichi**

更新于 6/9/2020, 7:03:50 AM

快速排序, 时间复杂度为 $O(n \log n)$ , 空间复杂度为 $O(\log n)$  在指针移动的时候, 没有检查 $left \leq right$ , 代码也可以运行 不知道为什么在while循环中重复检查 $left \leq right$

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm_source=code
 */
public class Solution {
    /**
     * @param A: an integer array
     * @return: nothing
     */
    public void sortIntegers(int[] A) {
        // write your code here
        if (A == null || A.length == 0) {
            return;
        }

        quickSort(A, 0, A.length - 1);
    }

    private void quickSort(int[] A, int start, int end) {
        if (start >= end) {
            return;
        }

        int left = start, right = end;
        int pivot = A[(left + right) / 2];
        while (left <= right) {
            while (A[left] < pivot) {
                left++;
            }

            while(A[right] > pivot) {
                right--;
            }

            if(left <= right) {
                int tmp = A[left];
                A[left] = A[right];
                A[right] = tmp;
                left++;
                right--;
            }
        }

        quickSort(A, start, right);
        quickSort(A, left, end);
    }
}
```

👍 获赞 2

💬 2 条评论



九章令狐冲

更新于 11/3/2020, 3:49:39 AM

归并排序

java

python

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm_source=code
 */
public class Solution {
    public void sortIntegers(int[] A) {
        if (A == null || A.length == 0) {
            return;
        }
        int[] temp = new int[A.length];
        mergeSort(A, 0, A.length - 1, temp);
    }

    private void mergeSort(int[] A, int start, int end, int[] temp) {
        if (start >= end) {
            return;
        }
        mergeSort(A, start, (start + end) / 2, temp);
        mergeSort(A, (start + end) / 2 + 1, end, temp);
        merge(A, start, end, temp);
    }

    private void merge(int[] A, int start, int end, int[] temp) {
        int middle = (start + end) / 2;
        int leftIndex = start;
        int rightIndex = middle + 1;
        int index = start;
        while (leftIndex <= middle && rightIndex <= end) {
            if (A[leftIndex] < A[rightIndex]) {
                temp[index++] = A[leftIndex++];
            } else {
                temp[index++] = A[rightIndex++];
            }
        }
        while (leftIndex <= middle) {
            temp[index++] = A[leftIndex++];
        }
        while (rightIndex <= end) {
            temp[index++] = A[rightIndex++];
        }
        for (int i = start; i <= end; i++) {
            A[i] = temp[i];
        }
    }
}
```

👍 获赞 0

💬 添加评论



九章令狐冲

更新于 11/3/2020, 3:47:31 AM

快速排序

java

python

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ /
Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm_source=code
 */
public class Solution {
    /**
     * @param A: an integer array
     * @return: nothing
     */
    public void sortIntegers(int[] A) {
        if (A == null || A.length == 0) {
            return;
        }
        quickSort(A, 0, A.length - 1);
    }

    private void quickSort(int[] A, int start, int end) {
        if (start >= end) {
            return;
        }

        int left = start, right = end;
        int pivot = A[(start + end) / 2];
        while (left <= right) {
            while (left <= right && A[left] < pivot) {
                left++;
            }
            while (left <= right && A[right] > pivot) {
                right--;
            }
            if (left <= right) {
                int temp = A[left];
                A[left] = A[right];
                A[right] = temp;
                left++;
                right--;
            }
        }
        quickSort(A, start, right);
        quickSort(A, left, end);
    }
}
```

👍 获赞 0    💬 添加评论



你是疯儿我是傻

更新于 6/21/2020, 8:34:55 PM

堆排序非递归写法:

java

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm_source=code
 */
private void shiftDown(int[] A, int i, int lastIndex) {
    int cur = i;
    while(cur <= lastIndex) {
        int child = 2 * cur + 1;
        if(child + 1 <= lastIndex && A[child + 1] > A[child]) {
            child = child + 1; //compare left & right child, take the greater one
        }

        if(child > lastIndex || A[child] <= A[cur]) break;

        swap(A, cur, child);
        cur = child;
    }
}

public void heapify(int[] A) {
    for (int i = (A.length - 1) / 2; i >= 0; i--) {
        shiftDown(A, i, A.length - 1);
    }
}

void sortIntegers(int[] A) {
    heapify(A);
    for (int i = A.length - 1; i > 0; i--) {
        int tmp = A[0];
        A[0] = A[i];
        A[i] = tmp;
        shiftDown(A, 0, i - 1);
    }
}
```

👍 获赞 0

💬 添加评论



jingyi.liu

更新于 6/9/2020, 7:04:24 AM

使用了递归的思路, 但是在pycharm可以运行出来, 在测试的时候反而有错, 请指正

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm_source=code
 */
class Solution:
    """
    @param A: an integer array
    @return: nothing
    """
    # write your code here
    def sortIntegers(self,A):
        smallest = A[0]
        if len(A) < 2:
            return A
        for i in range(1, len(A) - 1):
            if A[i] < smallest:
                smallest = A[i]
        A.remove(smallest)
        return [smallest] + self.sortIntegers(A)
```

👍 获赞 0    💬 添加评论



**Nidhogg-D-Joking**

更新于 6/9/2020, 7:04:18 AM

点击这里看文档 ([https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/Global\\_Objects/Array/sort](https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/Global_Objects/Array/sort))JavaScript现有的排序方法!

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm_source=code
 */
const sortIntegers = function (A) {
    A.sort(function(a, b) {
        return a - b;
    });
}
```

👍 获赞 0    💬 添加评论



**九章用户X8R5MA**

更新于 6/9/2020, 7:04:14 AM

## 选择排序



```
/**
 * 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括：系统设计 System Design，面向对象设计 OOD
 * - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
 * - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
 */
思想：扫描一次数组，我们就能得到最小值，把这个最小值取出，再扫描剩下的元素，就能得到第二小的值，以此类推。

设数组是A[0...n-1]

扫描A[0...n-1]，选出最小的，与A[0]交换，A[0]已经有序；
扫描A[1...n-1]，选出最小的，与A[1]交换，A[0...1]已经有序；
扫描A[2...n-1]，选出最小的，与A[2]交换，A[0...2]已经有序；
...
扫描A[n-2...n-1]，选出最小的，与A[n-2]交换，A[0...n-2]已经有序；
整个数组已经有序

时间复杂度：O(n^2)
为数不多的优点：原地排序
实际场景下几乎不使用
```

## 冒泡排序

```
/**
 * 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括：系统设计 System Design，面向对象设计 OOD
 * - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
 * - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
 */
思想：我有强迫症，看到这里有一对数字不是有序的，我就交换它们！又发现了一对，再交换...直到没有不是有序的数对为止！

设数组是A[0...n-1]

考察A[0]和A[1]，A[1]和A[2]...A[n-2]和A[n-1]，如果不是有序就交换，这一轮考察结束后，最大的已经被交换（冒泡）到A[n-1]；
考察A[0]和A[1]，A[1]和A[2]...A[n-3]和A[n-2]，如果不是有序就交换，这一轮考察结束后，第2大的已经被交换（冒泡）到A[n-2]；
...
考察A[0]和A[1]，如果不是有序就交换，这一轮考察结束后，第n-1大的已经被交换（冒泡）到A[1]；
整个数组已经有序
Notes：一共考察了n-1轮，每一轮考察的范围都会减少1，总比较次数为n*(n-1)/2；最多做n*(n-1)/2次交换（交换次数太多了）。

时间复杂度：O(n^2)
也属于排序的入门级算法
```

## 插入排序

```
/**
 * 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ /
Resume / Project 2020版
 * - Design类课程包括：系统设计 System Design，面向对象设计 OOD
 * - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
 * - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
 */
插排的原理就类似于抓扑克牌，每次摸一张牌，然后把它插入之前已经有序的牌中。
```

插入排序的具体做法如下：

- 从第2个元素开始（第1个元素已经有序），  
在已经有序的序列中从右往左走，  
如果当前位置元素大于待插入元素，就将其往右挪一位，  
直到走到数组头部或者走到第一个小于等于待插入元素的位置，将待插入元素挪到到该位置之后；
- 取出下一个元素（未排序的序列），重复上面的步骤，直到整个序列处理完。

插排的性能总体而言是比较好的，尤其在数组近似有序的时候，因为每个元素平均移动的次数将比较少。  
虽然最坏时间复杂度是 $O(n^2)$ （数组完全逆序），但是最佳时间复杂度是 $O(n)$ （数组完全有序）。  
在STL中，sort内部的算法中在排序的后期使用了插排。

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm\_source=code
 */
class Solution {
public:
    /**
     * @param A an integer array
     * @return void
     */
    void sortIntegers(vector<int>& A) {
        //SelectionSort(A);
        //BubbleSort(A);
        InsertionSort(A);
    }

    template<typename T>
    void SelectionSort(vector<T>& A) {
        int n = A.size();
        // 扫描n-1次
        for (int i = 0; i < n - 1; ++i) {
            int pos_min = i; // 最小值的位置
            for (int j = i; j < n; ++j) {
                if (A[j] < A[pos_min]) {
                    pos_min = j;
                }
            }
            std::swap(A[i], A[pos_min]);
        }
    }

    template<typename T>
    void BubbleSort(vector<T>& A) {
        int n = A.size();
        // 考察n-1轮
        for (int i = 0; i < n - 1; ++i) {
            for (int j = 0; j < n-1-i; ++j) {
                if (A[j] > A[j+1]) {
                    std::swap(A[j], A[j+1]);
                }
            }
        }
    }

    template<typename T>
    void InsertionSort(vector<T>& A) {
        int n = A.size();
        for (int i = 1; i < n; ++i) {
            T temp = std::move(A[i]);
            int j;
            for (j = i; j > 0 && temp < A[j-1]; --j) {
                A[j] = std::move(A[j-1]);
            }
            A[j] = std::move(temp);
        }
    }
};
```

👍 获赞 0    💬 添加评论

进阶课程

视频+互动	直播+互动	直播+互动	互动课
<div>九章算法班 2021 版</div> <div>8周时间精通 57 个核心高频考点，9 招击破 FLAG、BATJ 算法面试。22....</div>	<div>系统架构设计 System Design 2021 版</div> <div>成为百万架构师必上。30 课时带你快速掌握18大系统架构设计知识点与面...</div>	<div>九章算法面试高频题冲刺班</div> <div>每期更新 15% 题目，考前押题，一举拿下FLAG &amp; BATJ Offer</div>	<div>面向对象设计 OOD</div> <div>应届生及亚马逊面试必考，IT求职必备基础</div>