**算法设计与分析第三章作业**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **姓名** | **龙衍程** | **班级** | **计科八班** | **学号** | 220110804 |
| **第1题** |  | | | | |
| **第2题** |  | | | | |
| **第3题** |  | | | | |
| **第4题** |  | | | | |
| **第5题** |  | | | | |
| **总分** |  | | | | |
| **备注** | 作业提交截止时间：2023-10-14日24:00，超过提交截至时间的作业视为无效。作业提交邮箱：[272181768@QQ.com。](mailto:hitcsalgo2020@163.com。作业的首页要注明：学生姓名，班级，学号)作业文件名命名方式： 第x章-x班-姓名-学号（例， 第1章-1班-张三-2018054000.doc）； 邮件主题为：第x章作业, x班，姓名，学号（例， 第1章作业，1班，张三，2018054000）。缺少这些信息的作业将被酌情扣分。 | | | | |

1. 将数组进行归并排序，并通过递归树方法计算归并排序时间复杂度。**（该题考察归并排序以及时间复杂度，15分）**

* ***归并排序：***

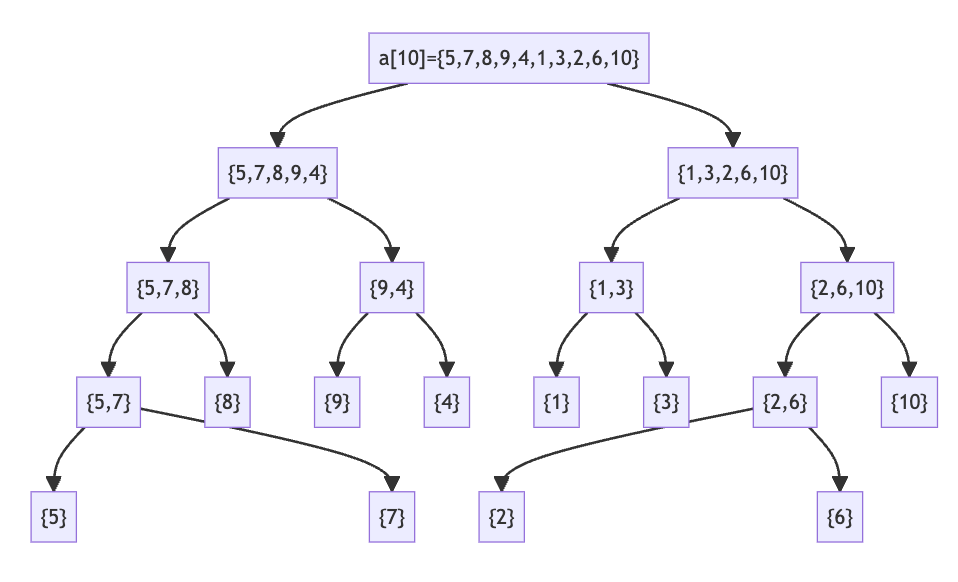
*给定数组 a[10]={5,7,8,9,4,1,3,2,6,10}：*

* 1. *分治：*
     1. *将数组分为两部分：{5,7,8,9,4} 和 {1,3,2,6,10}。*
     2. *将每部分再次分为两部分：{5,7,8} 和 {9,4}；{1,3,2} 和 {6,10}*
     3. *继续分割：{5}、{7,8}、{9}、{4}；{1}、{3,2}、{6}、{10}*
     4. *继续分割：{7}、{8}、{3}、{2}*
  2. *归并：*
     1. *{7,8} 归并为 {7,8}；{3,2} 归并为 {2,3}*
     2. *{5,7,8} 归并为 {5,7,8}；{9,4} 归并为 {4,9}；*

*{1,2,3} 归并为 {1,2,3}；{6,10} 归并为 {6,10}*

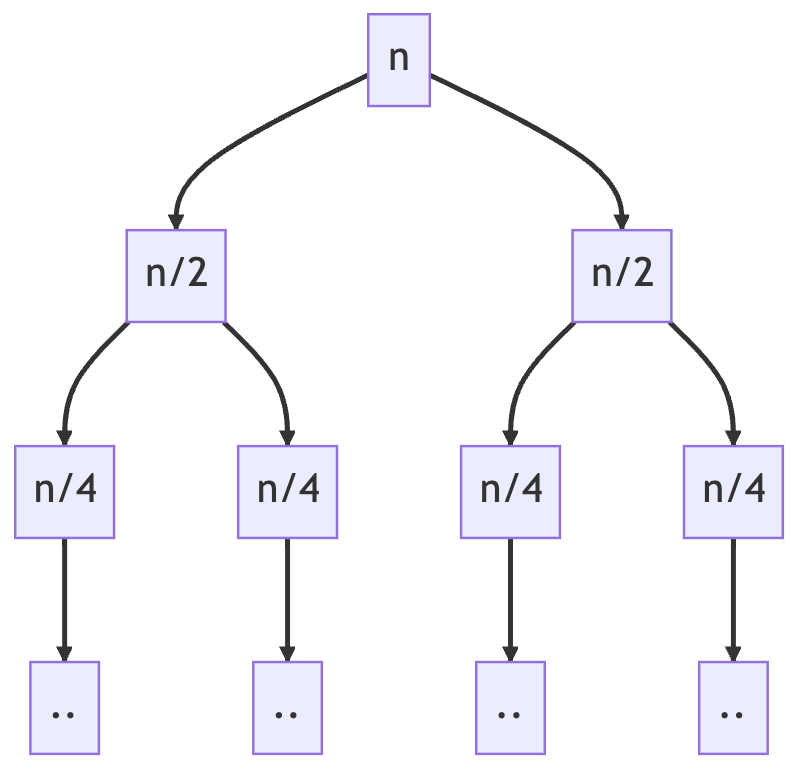
* + 1. *{5,7,8,4,9} 归并为 {4,5,7,8,9}；{1,2,3,6,10} 归并为 {1,2,3,6,10}*
    2. *{4,5,7,8,9,1,2,3,6,10} 归并为 {1,2,3,4,5,6,7,8,9,10}*

*具体归并步骤如下图：*

**

* ***时间复杂度计算：***

*归并排序的递归树如下图所示：*

**

*每层归并操作的时间复杂度为线性，共有层，总时间复杂度为*

1. 存储给定一个个元素有序的（升序）整型数组和一个目标值，写一个时间复杂度为的伪代码搜索中的**（需附完整Python源代码）**，如果目标值存在返回下标，否则返回。证明代码时间复杂度为。**（该题考察分治法以及时间复杂度，15分）**

**示例 1: 示例 2:**

**输入: nums = [-1,0,3,5,9,12], target = 9 输入: nums = [-1,0,3,5,9,12], target = 2**

**输出: 4 输出: -1**

**解释: 9 出现在 nums 中并且下标为 4 解释: 2 不存在 nums 中因此返回 -1**

***思想：****采用二分法进行二分搜索*

|  |  |
| --- | --- |
| ***伪代码*** | ***python代码*** |
| *int function bi\_search(nums, target)*  *{*  *left = 0*  *right = length - 1*  *while left <= right:*  *mid = (left + right) / 2*  *if nums[mid] == target:*  *return mid*  *else if nums[mid] < target:*  *left = mid + 1*  *else:*  *right = mid - 1*  *return -1*  *}* | *def bi\_search(nums, target):*  *left, right = 0, len(nums) - 1*  *while left <= right:*  *mid = (left + right) // 2*  *if nums[mid] == target:*  *return mid*  *elif nums[mid] < target:*  *left = mid + 1*  *else:*  *right = mid - 1*  *return -1* |

*每一步搜索时间复杂度为O（1），中将搜索范围减半，总时间复杂度为O（logn）*

1. 给定由个互不相同的数组成的集合以及正整数，试设计一个时间复杂度的算法，并写出伪代码，找出中最接近的中位数的个数。**（该题考察分治策略，20分）**

***思想：***

*以课程中的MoM快速选择算法为基础，找出中位数，并对各数与中位数差值进行快速选择*

***伪代码：***

*function closest\_to\_median(s, k)*

*{*

*//找到s的中位数*

*median = MOMSelect(s, len(s) // 2)*

*//计算每个元素与中位数的差值*

*for(auto i : s):*

*diffs.append(abs(i - median))*

*//使用MOM算法找到第k小的差值*

*kth\_smallest\_diff = MOMSelect(diffs, k)*

*//选择差值最小的k个元素*

*for(auto i : s):*

*if(abs(i - median) <= kth\_smallest\_diff):*

*result.append(i)*

*return result*

*}*

*int function MOMSelect(s, i)*

*{*

*//返回s中第i小的元素*

*for(int j=1;i<=n/5;j++)*

*{*

*InsertSort(A[(j-1)\*5+1:(j-1)\*5+5]);*

*Swap(B[j],A[(j-1)\*5+3]);*

*}*

*x=MOMSelect(B[1:n/5],n/10);*

*k=partition(A[1:n],x);*

*if(k = i)return x;*

*else if(k > i)return MOMSelect(A[1:k-1],i);*

*else return MOMSelect(A[k+1:n],i-k);*

*}*

1. 输入两个大小分别为和的正序（从小到大）数组和。设计算法，并编写时间复杂度为的伪代码**（需附完整Python源代码）**，找出并返回这两个正序数组的中位数。**（该题考察分治法以及时间复杂度，20分）**

**示例 1： 示例2：**

**输入：nums1=[1,3], nums2=[2] 输入：nums1=[1,2], nums2=[3,4]**

**输出：2.00000 输出：2.50000**

**解释：合并数组=[1,2,3]，中位数2 解释：合并数组=[1,2,3,4]，中位数 (2+3)/2=2.5**

***思想：****用二分对较长数组进行搜索，不断确定ij位置，最终得到中位数*

|  |  |
| --- | --- |
| *伪代码* | *python代码* |
| *findMedian(nums1,nums2) {*  *if (nums1.size() > nums2.size())*  *swap(nums1, nums2);*  *int m = nums1.size();*  *int n = nums2.size();*  *int imin = 0, imax = m, half\_len = (m + n + 1) / 2;*  *max\_of\_left, min\_of\_right=0,0;*  *while (imin <= imax) {*  *int i = (imin + imax) / 2;*  *int j = half\_len - i;*  *if (i < m && nums2[j-1] > nums1[i])*  *imin = i + 1;*  *else if (i > 0 && nums1[i-1] > nums2[j])*  *imax = i - 1;*  *else {*  *if (i == 0) {*  *max\_of\_left = nums2[j-1];*  *} else if (j == 0) {*  *max\_of\_left = nums1[i-1];*  *} else {*  *max\_of\_left = max(nums1[i-1], nums2[j-1]);*  *}*  *if ((m + n) % 2 == 1)*  *return max\_of\_left;*  *if (i == m) {*  *min\_of\_right = nums2[j];*  *} else if (j == n) {*  *min\_of\_right = nums1[i];*  *} else {*  *min\_of\_right = min(nums1[i], nums2[j]);*  *}*  *return (max\_of\_left + min\_of\_right) / 2.0;*  *}*  *}*  *return 0;* | *def findMedian(nums1, nums2):*  *if len(nums1) > len(nums2):*  *nums1, nums2 = nums2, nums1*  *m, n = len(nums1), len(nums2)*  *imin, imax, half\_len = 0, m, (m + n + 1) // 2*  *while imin <= imax:*  *i = (imin + imax) // 2*  *j = half\_len - i*  *if i < m and nums2[j-1] > nums1[i]:*  *imin = i + 1*  *elif i > 0 and nums1[i-1] > nums2[j]:*  *imax = i - 1*  *else:*  *if i == 0: max\_of\_left = nums2[j-1]*  *elif j == 0: max\_of\_left = nums1[i-1]*  *else: max\_of\_left = max(nums1[i-1], nums2[j-1])*  *if (m + n) % 2 == 1:*  *return max\_of\_left*  *if i == m: min\_of\_right = nums2[j]*  *elif j == n: min\_of\_right = nums1[i]*  *else: min\_of\_right = min(nums1[i], nums2[j])*  *return (max\_of\_left + min\_of\_right) / 2.0* |

1. 输入一个整数数组和一个整数，请输出其中出现频率前高的元素，设计算法并写出伪代码**（需附完整Python源代码）**。**（该题考察分治法中快速排序划分的思想，30分）**

**示例 1: 示例 2:**

**输入: nums = [1,1,1,2,2,3], k = 2 输入: nums = [1], k = 1**

**输出: [1,2] 输出: [1]**

***思想：***

*使用哈希表统计每个元素的频率并作为二元组填入列表，对频率进行快速选择排序*

***伪代码：***

*function topKFreq(nums, k) {*

*// 用哈希表统计元素的频率*

*map<int, int> freq\_Map;*

*for (int num : nums) {*

*if (freq\_Map.find(num) == freq\_Map.end()) {*

*freq\_Map[num] = 1;*

*} else {*

*freq\_Map[num]++;*

*}*

*}*

*vector<pair<int, int>> pairs;*

*for (auto& kv : freq\_Map) {*

*pairs.push\_back(kv);*

*}*

*// 快排划分*

*function partition(int left, int right, int pivot\_index) {*

*int pivot\_freq = pairs[pivot\_index].second;*

*swap(pairs[pivot\_index], pairs[right]);*

*int store\_index = left;*

*for (int i = left; i < right; i++) {*

*if (pairs[i].second > pivot\_freq) {*

*swap(pairs[store\_index], pairs[i]);*

*store\_index++;*

*}*

*}*

*swap(pairs[store\_index], pairs[right]);*

*return store\_index;*

*}*

*// 快排选择*

*function quickSelect(int left, int right, int k\_smallest) {*

*if (left == right) {*

*return;*

*}*

*int pivot\_index = random(left, right); //*

*pivot\_index = partition(left, right, pivot\_index);*

*if (k\_smallest == pivot\_index) {*

*return;*

*} else if (k\_smallest < pivot\_index) {*

*quickSelect(left, pivot\_index - 1, k\_smallest);*

*} else {*

*quickSelect(pivot\_index + 1, right, k\_smallest);*

*}*

*}*

*quickSelect(0, pairs.size() - 1, k - 1);*

*vector<int> result;*

*for (int i = 0; i < k; i++) {*

*result.push\_back(pairs[i].first);*

*}*

*return result;*

*}*

***python代码:***

*def topKFreq(nums, k):*

*# 哈希表统计元素的频率*

*frequencyMap = {}*

*for num in nums:*

*if num not in frequencyMap:*

*frequencyMap[num] = 1*

*else:*

*frequencyMap[num] += 1*

*pairs = list(frequencyMap.items())*

*# 快排划分*

*def partition(left, right, pivot\_index):*

*pivot\_frequency = pairs[pivot\_index][1]*

*pairs[pivot\_index], pairs[right] = pairs[right], pairs[pivot\_index]*

*store\_index = left*

*for i in range(left, right):*

*if pairs[i][1] > pivot\_frequency:*

*pairs[store\_index], pairs[i] = pairs[i], pairs[store\_index]*

*store\_index += 1*

*pairs[right], pairs[store\_index] = pairs[store\_index], pairs[right]*

*return store\_index*

*#快排选择*

*def quickSelect(left, right, k\_smallest):*

*if left == right:*

*return*

*pivot\_index = random.randint(left, right)*

*pivot\_index = partition(left, right, pivot\_index)*

*if k\_smallest == pivot\_index:*

*return*

*elif k\_smallest < pivot\_index:*

*quickSelect(left, pivot\_index - 1, k\_smallest)*

*else:*

*quickSelect(pivot\_index + 1, right, k\_smallest)*

*quickSelect(0, len(pairs) - 1, k - 1)*

*return [item[0] for item in pairs[:k]]*