第三章算法分析

1. 编写程序实现归并排序算法 MergeSortL 和快速排序算法 QuickSort;

主要代码如下，完整代码见压缩包内 python 文件。

MergeSortL:

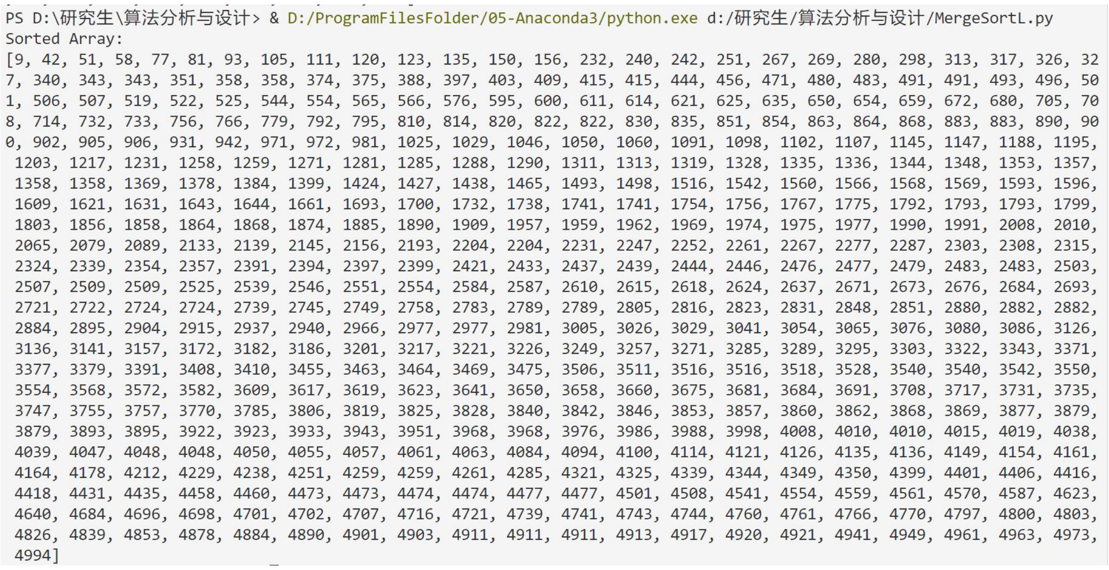
```
class ListNode: # 存储 index
    def __init__(self, val=0, next=None):
        self.val = val
        self.next = next

def insert_to_list(list_head, array, index):
    current_node = ListNode(val = index)
    # print("=========================" )
    # print(index)
    # print(array[index])
    # print_list(list_head)
    if (list_head == None):
        return current_node
    current_point = list_head
    last_node = None

    while(current_point != None):
        compared_index = current_point.val
        # print(array[compared_index])
        if (array[compared_index] > array[index]):
            if (last_node == None):
                current_node.next = list_head
                return current_node
            last_node.next = current_node
            current_node.next = current_point
            return list_head
        else:
            last_node = current_point
            current_point = current_point.next

    last_node.next = current_node
    return list_head

def insertion_sort(array, start, end):
    list_head = None
    for i in range(start, end + 1):
        list_head = insert_to_list(list_head, array, i)
        # print_list(list_head)
    return list_head
```

```
def merge_l(node_a, node_b, array):

    new_head = None
    current_node = None
    i = node_a
    j = node_b
    while i and j:
        i_num = array[i.val]
        j_num = array[j.val]

        if i_num <= j_num:
            # tmp_i_next = i.next
            if (new_head == None):
                new_head = i
            else :
                current_node.next = i
            current_node = i
            i = i.next
        else:
            if (new_head == None):
                new_head = j
            else :
                current_node.next = j
            current_node = j
            j = j.next
    if i:
        current_node.next = i
    if j:
        current_node.next = j

    return new_head

def merge_sort_l(array, start, end):
    if end - start + 1 < 16:
        return insertion_sort(array, start, end)
    else:
        mid = (start + end) >> 1
        node_a = merge_sort_l(array, start, mid)
        node_b = merge_sort_l(array, mid + 1, end)
        return merge_l(node_a, node_b, array)
```

QuickSort:

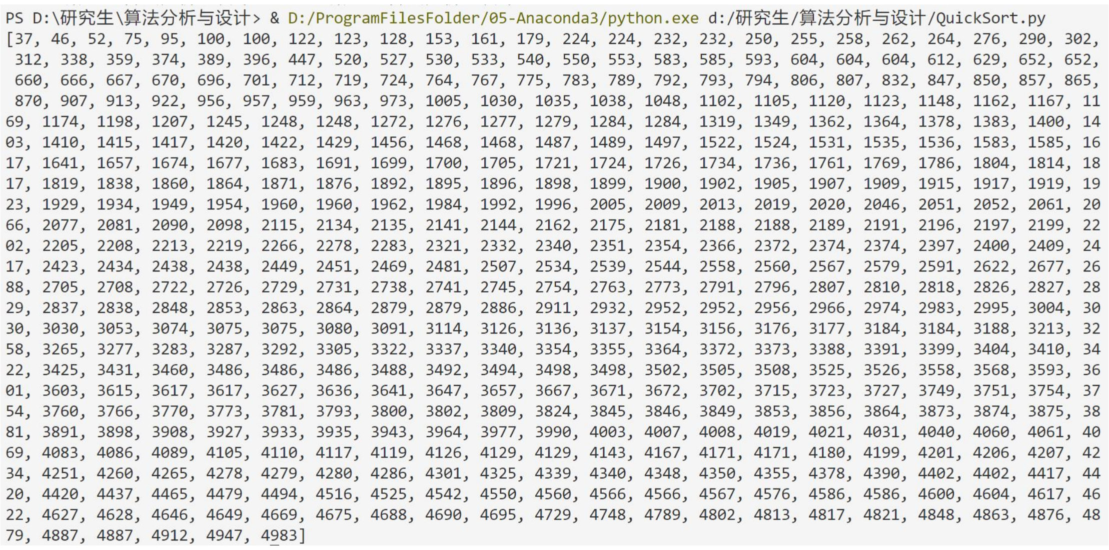
```
def quicksort(array, start, end):
    if start < end:
        pi = partition(array, start, end)
        quicksort(array, start, pi - 1)
        quicksort(array, pi + 1, end)

def partition(array, low, high):
    pivot = array[high]
    i = low - 1
    for j in range(low, high):
        if array[j] < pivot:
            i += 1
            array[i], array[j] = array[j], array[i]
    array[i + 1], array[high] = array[high], array[i + 1]
    return i + 1
```

MergeSortL 产生 500 个随机数（取值范围 1-5000）的排序结果：



```
PS D:\研究生\算法分析与设计> & D:/ProgramFilesFolder/05-Anaconda3/python.exe d:/研究生/算法分析与设计/MergeSortL.py
Sorted Array:
[9, 42, 51, 58, 77, 81, 93, 105, 111, 120, 123, 135, 150, 156, 232, 240, 242, 251, 267, 269, 280, 298, 313, 317, 326, 32
7, 340, 343, 343, 351, 358, 358, 374, 375, 388, 397, 403, 409, 415, 415, 444, 456, 471, 480, 483, 491, 491, 493, 496, 50
1, 506, 507, 519, 522, 525, 544, 554, 565, 566, 576, 595, 600, 611, 614, 621, 625, 635, 650, 654, 659, 672, 680, 705, 70
8, 714, 732, 733, 756, 766, 779, 792, 795, 810, 814, 820, 822, 822, 830, 835, 851, 854, 863, 864, 868, 883, 883, 890, 90
0, 902, 905, 906, 931, 942, 971, 972, 981, 1025, 1029, 1046, 1050, 1060, 1091, 1098, 1102, 1107, 1145, 1147, 1188, 1195,
 1203, 1217, 1231, 1258, 1259, 1271, 1281, 1285, 1288, 1290, 1311, 1313, 1319, 1328, 1335, 1336, 1344, 1348, 1353, 1357,
 1358, 1358, 1369, 1378, 1384, 1399, 1424, 1427, 1438, 1465, 1493, 1498, 1516, 1542, 1560, 1566, 1568, 1569, 1593, 1596,
 1609, 1621, 1631, 1643, 1644, 1661, 1693, 1700, 1732, 1738, 1741, 1741, 1754, 1756, 1767, 1775, 1792, 1793, 1793, 1799,
 1803, 1856, 1858, 1864, 1868, 1874, 1885, 1890, 1909, 1957, 1959, 1962, 1969, 1974, 1975, 1977, 1990, 1991, 2008, 2010,
 2065, 2079, 2089, 2133, 2139, 2145, 2156, 2193, 2204, 2204, 2231, 2247, 2252, 2261, 2267, 2277, 2287, 2303, 2308, 2315,
 2324, 2339, 2354, 2357, 2391, 2394, 2397, 2399, 2421, 2433, 2437, 2439, 2444, 2446, 2476, 2477, 2479, 2483, 2483, 2503,
 2507, 2509, 2509, 2525, 2539, 2546, 2551, 2554, 2584, 2587, 2610, 2615, 2618, 2624, 2637, 2671, 2673, 2676, 2684, 2693,
 2721, 2722, 2724, 2724, 2739, 2745, 2749, 2758, 2783, 2789, 2789, 2805, 2816, 2823, 2831, 2848, 2851, 2880, 2882, 2882,
 2884, 2895, 2904, 2915, 2937, 2940, 2966, 2977, 2977, 2981, 3005, 3026, 3029, 3041, 3054, 3065, 3076, 3080, 3086, 3126,
 3136, 3141, 3157, 3172, 3182, 3186, 3201, 3217, 3221, 3226, 3249, 3257, 3271, 3285, 3289, 3295, 3303, 3322, 3343, 3371,
 3377, 3379, 3391, 3408, 3410, 3455, 3463, 3464, 3469, 3475, 3506, 3511, 3516, 3516, 3518, 3528, 3540, 3540, 3542, 3550,
 3554, 3568, 3572, 3582, 3609, 3617, 3619, 3623, 3641, 3650, 3658, 3660, 3675, 3681, 3684, 3691, 3708, 3717, 3731, 3735,
 3747, 3755, 3757, 3770, 3785, 3806, 3819, 3825, 3828, 3840, 3842, 3846, 3853, 3857, 3860, 3862, 3868, 3869, 3877, 3879,
 3879, 3893, 3895, 3922, 3923, 3933, 3943, 3951, 3968, 3968, 3976, 3986, 3988, 3998, 4008, 4010, 4010, 4015, 4019, 4038,
 4039, 4047, 4048, 4048, 4050, 4055, 4057, 4061, 4063, 4084, 4094, 4100, 4114, 4121, 4126, 4135, 4136, 4149, 4154, 4161,
 4164, 4178, 4212, 4229, 4238, 4251, 4259, 4259, 4261, 4285, 4321, 4325, 4339, 4344, 4349, 4350, 4399, 4401, 4406, 4416,
 4418, 4431, 4435, 4458, 4460, 4473, 4473, 4474, 4474, 4477, 4477, 4501, 4508, 4541, 4554, 4559, 4561, 4570, 4587, 4623,
 4640, 4684, 4696, 4698, 4701, 4702, 4707, 4716, 4721, 4739, 4741, 4743, 4744, 4760, 4761, 4766, 4770, 4797, 4800, 4803,
 4826, 4839, 4853, 4878, 4884, 4890, 4901, 4903, 4911, 4911, 4911, 4913, 4917, 4920, 4921, 4941, 4949, 4961, 4963, 4973,
 4994]
```

QuickSort 产生 500 个随机数（取值范围 1-5000）的排序结果：



```
PS D:\研究生\算法分析与设计> & D:/ProgramFilesFolder/05-Anaconda3/python.exe d:/研究生/算法分析与设计/QuickSort.py
[37, 46, 52, 75, 95, 100, 100, 122, 123, 128, 153, 161, 179, 224, 224, 232, 232, 250, 255, 258, 262, 264, 276, 290, 302,
 312, 338, 359, 374, 389, 396, 447, 520, 527, 530, 533, 540, 550, 553, 583, 585, 593, 604, 604, 604, 612, 629, 652, 652,
 660, 666, 667, 670, 696, 701, 712, 719, 724, 764, 767, 775, 783, 789, 792, 793, 794, 806, 807, 832, 847, 850, 857, 865,
 870, 907, 913, 922, 956, 957, 959, 963, 973, 1005, 1030, 1035, 1038, 1048, 1102, 1105, 1120, 1123, 1148, 1162, 1167, 11
69, 1174, 1198, 1207, 1245, 1248, 1248, 1272, 1276, 1277, 1279, 1284, 1284, 1319, 1349, 1362, 1364, 1378, 1383, 1400, 14
03, 1410, 1415, 1417, 1420, 1422, 1429, 1456, 1468, 1468, 1487, 1489, 1497, 1522, 1524, 1531, 1535, 1536, 1583, 1585, 16
17, 1641, 1657, 1674, 1677, 1683, 1691, 1699, 1700, 1705, 1721, 1724, 1726, 1734, 1736, 1761, 1769, 1786, 1804, 1814, 18
17, 1819, 1838, 1860, 1864, 1871, 1876, 1892, 1895, 1896, 1898, 1899, 1900, 1902, 1905, 1907, 1909, 1915, 1917, 1919, 19
23, 1929, 1934, 1949, 1954, 1960, 1960, 1962, 1984, 1992, 1996, 2005, 2009, 2013, 2019, 2020, 2046, 2051, 2052, 2061, 20
66, 2077, 2081, 2090, 2098, 2115, 2134, 2135, 2141, 2144, 2162, 2175, 2181, 2188, 2188, 2189, 2191, 2196, 2197, 2199, 22
02, 2205, 2208, 2213, 2219, 2266, 2278, 2283, 2321, 2332, 2340, 2351, 2354, 2366, 2372, 2374, 2374, 2397, 2400, 2409, 24
17, 2423, 2434, 2438, 2438, 2449, 2451, 2469, 2481, 2507, 2534, 2539, 2544, 2558, 2560, 2567, 2579, 2591, 2622, 2677, 26
88, 2705, 2708, 2722, 2726, 2729, 2731, 2738, 2741, 2745, 2754, 2763, 2773, 2791, 2796, 2807, 2810, 2818, 2826, 2827, 28
29, 2837, 2838, 2848, 2853, 2863, 2864, 2879, 2879, 2886, 2911, 2932, 2952, 2952, 2956, 2966, 2974, 2983, 2995, 3004, 30
30, 3030, 3053, 3074, 3075, 3075, 3080, 3091, 3114, 3126, 3136, 3137, 3154, 3156, 3176, 3177, 3184, 3184, 3188, 3213, 32
58, 3265, 3277, 3283, 3287, 3292, 3305, 3322, 3337, 3340, 3354, 3355, 3364, 3372, 3373, 3388, 3391, 3399, 3404, 3410, 34
22, 3425, 3431, 3460, 3486, 3486, 3486, 3488, 3492, 3494, 3498, 3498, 3502, 3505, 3508, 3525, 3526, 3558, 3568, 3593, 36
01, 3603, 3615, 3617, 3617, 3627, 3636, 3641, 3647, 3657, 3667, 3671, 3672, 3702, 3715, 3723, 3727, 3749, 3751, 3754, 37
54, 3760, 3766, 3770, 3773, 3781, 3793, 3800, 3802, 3809, 3824, 3845, 3846, 3849, 3853, 3856, 3864, 3873, 3874, 3875, 38
81, 3891, 3898, 3908, 3927, 3933, 3935, 3943, 3964, 3977, 3990, 4003, 4007, 4008, 4019, 4021, 4031, 4040, 4060, 4061, 40
69, 4083, 4086, 4089, 4105, 4110, 4117, 4119, 4126, 4129, 4129, 4143, 4167, 4171, 4171, 4180, 4199, 4201, 4206, 4207, 42
34, 4251, 4260, 4265, 4278, 4279, 4280, 4286, 4301, 4325, 4339, 4340, 4348, 4350, 4355, 4378, 4390, 4402, 4402, 4417, 44
20, 4420, 4437, 4465, 4479, 4494, 4516, 4525, 4542, 4550, 4560, 4566, 4566, 4567, 4576, 4586, 4586, 4600, 4604, 4617, 46
22, 4627, 4628, 4646, 4649, 4669, 4675, 4688, 4690, 4695, 4729, 4748, 4789, 4802, 4813, 4817, 4821, 4848, 4863, 4876, 48
79, 4887, 4887, 4912, 4947, 4983]
```

2. 用长分别为 10000、30000、50000、80000、100000、200000 的 6 个数组(可用机器随机产生)的排列来统计这两种算法的时间复杂性；

执行时间的单位为：秒
随机数范围为 1-300000

	10000	30000	50000	80000	100000	200000
MergeSortL	0.01369	0.05854	0.10203	0.19005	0.26170	0.72831
QuickSort	0.01141	0.03777	0.06931	0.10264	0.15116	0.32312

可以发现 QuickSort 确实比 MergeSortL 快，那当然也会比没优化的 MergeSort 快
而且随着数据量的增大，这种差异越发明显

3．讨论归并排序算法 MergeSort 的空间复杂性。

归并排序由分解与合并两部分组成，如果用 S(n)表示排序 n 个数所用的总空间。
由分治：
分解的时候，MergeSort(low, mid) 和 MergeSort(mid+1, high)分别用 S(n/2)
归并的时候，Merge(low, mid, high) 空间复杂度为 O(n)，如果使用辅助数组，则约为 2n
则有：

$$S(n)=\max\{S(n / 2),O(n)\}$$

即：

$$S(n)<=S(n / 2)+O(n)$$

递归推导得：

$$S(n)<=S(n/2) + O(n)<=S(1)+O(n/2\text{^}k)+...+O(n/2)+O(n)$$

由推导进一步得：

$$S(n)<=S(1)+O(2n)=O(n)$$

又有存储数组长度为 n，则有 S(n)>=O(n)
综上得：

$$S(n)=O(n)$$

4．说明算法 PartSelect 的平均时间复杂性为$O(n)$。

提示：假定数组中的元素各不相同，且第一次划分时划分元素v是第i小元素的概率为$1/n$。因为 Partition 中的 case 语句所要求的时间都是$O(n)$，所以，存在常数c，使得算法 PartSelect 的平均时间复杂度$C_A^k(n)$可以表示为

$$C_A^k(n)\le cn+\frac{1}{n}\left(\sum_{1\le i<k}C_A^{k-i}(n-i)+\sum_{k<i\le n}C_A^k(i-1)\right)$$

令$R(n)=\max\limits_k(C_A^k(n))$，取$c\ge R(1)$，试证明$R(n)\le 4cn$。

证明如下：（下页）

令 $R(n) = \max(C_A^k(n))$. 设 $k=k_n$ 时取之. 则 $R(n)$ 有.

$$R(n) \leq cn + \frac{1}{n}\left(\sum_{1\leq i<k} C_A^{k-1}(n-i) + \sum_{k<i\leq n} C_A^{k}(i-1)\right)$$

取 $c \geq R(1)$ 当 $n=1$ 时. 取 $c\geq$年. 则 $R(1) \leq 4c$.

$n=2$ 时. $R(2) \leq 2c + \frac{1}{2}R(1) \leq 2c + 2c = 4c$ 成立.

假设结论对 $n=k-1$ 成立 则对 $n=k$ 有

$$R(k) \leq ck + \frac{1}{k}R(k-1) + \cdots + R(k-k_n+1) + R(k_n) + \cdots + R(k-1))$$

$$\leq ck + \frac{4c}{k}(k-1) + \cdots + (k-k_n+1) + k_n \cdots (k-1)$$

$$\leq ck + \frac{4c}{k}\left(\frac{(k_n-1)(2k-k_n)}{2} + \frac{(k-k_n)(k_n+k-1)}{2}\right)$$

$$\leq ck - \frac{4c}{k}\left(-\left(\frac{k+1}{2}\right)^2 - \frac{k^2-3k}{2}\right)$$

$$= ck + c\left(\frac{3k^2-4k+1}{k}\right)$$

$$\leq ck + c(3k-3)$$

$$\leq 4ck$$

归纳可得. $R(n) \leq 4cn$ 成立.

故 时间复杂度为 $O(n)$