

1.实现算法 mergeSort.py 如下:

```
#归并排序
import numpy as np
import math
import time
def mergeSort(arr):
    L=len(arr)
    if(L<2):#若
        return arr
    gap = math.floor(L/2)
    left = arr[0:gap]
    right = arr[gap:L]#分治
    return merge(mergeSort(left),mergeSort(right))
def merge(left,right):
    result=[]
    while(len(left)>0 and len(right)>0):
        if left[0]<=right[0]:
            result.append(left[0])
            del left[0]
        else:
            result.append(right[0])
            del right[0]
    while(len(left)):#剩下的已经保证是顺序了,依次插入
        result.append(left[0])
        del left[0]
    while(len(right)):
        result.append(right[0])
        del right[0]
    return result
# a=[1,3,5,7,2,4,6,8,3,4,5,61,123,4,2,72,4,9,223,4]
a=np.random.rand(200000).tolist()
tic=time.time()
b=mergeSort(a)
toc=time.time()
d=toc-tic
print(b)
print(d)
```

实现效果如下:

```
C:\WINDOWS\system32\cmd.exe - "D:\anaconda\condabin\conda.bat" activate py36
817575773538, 0.9990872677209522, 0.9990894210634882, 0.9990932482237408, 0.9991040890966983, 0.9991090109916745, 0.9991
090870463761, 0.9991132569012242, 0.9991157038577052, 0.9991226928600622, 0.9991400162464229, 0.9991422385580456, 0.9991
439015394283, 0.9991495327269589, 0.9991611870212194, 0.9991703329128664, 0.999173909216837, 0.9991787895194132, 0.9991
788975457762, 0.9991841875986879, 0.9991867158496253, 0.9991953790178988, 0.9991968986907559, 0.999198573015919, 0.99920
31990195991, 0.9992080966792405, 0.9992102509192695, 0.9992255456888443, 0.9992294757841954, 0.9992324263688614, 0.99923
31911389793, 0.9992336648195849, 0.9992340153553555, 0.9992379665520971, 0.9992391682314297, 0.9992399540413601, 0.99924
91299653143, 0.999258909644637, 0.999263346767253, 0.9992798993900455, 0.9992801006929481, 0.9992875495414877, 0.9992905
220233629, 0.9992913329990102, 0.9993013021599092, 0.9993077644682429, 0.9993173138869459, 0.9993181279369411, 0.9993225
387731653, 0.9993277043694216, 0.9993294528079203, 0.9993503611269865, 0.9993525083797407, 0.9993575522675339, 0.9993583
584115123, 0.9993590060928044, 0.9993925503507705, 0.9994097882495002, 0.9994235792889066, 0.9994269022547296, 0.9994289
301443366, 0.9994390610103396, 0.9994461271997221, 0.9994491310470426, 0.9994523603055038, 0.9994734981719782, 0.9994747
635523107, 0.9994760166792955, 0.9994780720060737, 0.9994886251429521, 0.9994930209842383, 0.9994972071335658, 0.9995004
526801475, 0.9995024653367259, 0.9995028384011952, 0.9995076330156296, 0.9995132058841301, 0.9995189753619482, 0.9995209
108356103, 0.9995288708815101, 0.9995295640114379, 0.9995296654446674, 0.9995325409029008, 0.9995375629671654, 0.9995408
672916425, 0.9995451249498372, 0.9995479514377587, 0.9995514460809586, 0.9995541974628872, 0.9995563850972773, 0.9995660
776998728, 0.9995702105836288, 0.9995836069279773, 0.9995839038983084, 0.9995877737480151, 0.999592900703295, 0.99959308
25460697, 0.9996147318395865, 0.9996194832774242, 0.9996213294815302, 0.9996262705831612, 0.9996388055161808, 0.99963960
12900756, 0.9996479577718379, 0.9996562459932853, 0.9996614262930744, 0.9996723265905653, 0.9996792370393651, 0.99969600
48505925, 0.9997024147081401, 0.9997086437089232, 0.9997094838296525, 0.999711400899547, 0.99971231154552, 0.99971413503
74377, 0.9997174229114433, 0.9997202242191593, 0.9997302361561163, 0.9997371076778252, 0.999742860524953, 0.999750696305
4884, 0.9997652012122618, 0.9997684213695607, 0.9997712072784232, 0.9997734139857101, 0.9997968817732176, 0.999798292035
0486, 0.9998020412113412, 0.9998119695106641, 0.999812766102482, 0.9998211694217721, 0.9998225786160988, 0.9998274047747
043, 0.9998326402808693, 0.9998361899526582, 0.9998401332949727, 0.9998445897239208, 0.9998581228110641, 0.9998613037785
355, 0.9998678448614314, 0.9998725887402085, 0.9998743769811211, 0.9998837996325124, 0.9998997756428272, 0.9999089698238
99, 0.9999253708687328, 0.9999357751282882, 0.9999411839908028, 0.9999438846362029, 0.9999541298250244, 0.99995425763826
68, 0.999961696557024, 0.9999619001156276, 0.9999657061457337, 0.9999698736220201, 0.9999855144408621, 0.999987907528350
5, 0.9999946422790081, 0.9999976330811026, 0.9999984814734683, 0.9999990025595296]
数组长度为: 200000 。耗费时间为: 6.58355975151062 s
(py36) D:\子钰\文案\个人\研究生\研一\计算机算法设计与分析\src>
```

编写 quickSort.py 如下:

```
#快速排序
import numpy as np
import time
def QS(arr,left,right):
    L=len(arr)
    if left<right:
        partitionIndex=partition(arr,left,right)#分区操作
        QS(arr,left,partitionIndex)
        QS(arr,partitionIndex+1,right)
    return arr
def partition(arr,left,right):
    pivot = left#基准值
    index=pivot+1
    for i in range(index,right+1):
        if arr[i]<arr[pivot]:
            swap(arr,index,i)
            index+=1
    swap(arr,index-1,pivot)#最后将基准值排在左边数组的最后, 因为已经可以确定
    该元素是最大的
    return index-1
def swap(arr,i,j):
    count=arr[i]
    arr[i]=arr[j]
    arr[j]=count
n=200000
a=np.random.rand(n).tolist()
tic=time.time()
left=0
right=len(a)-1
```

```

b=QS(a,left,right)
toc=time.time()
d=toc-tic
print(b)
print('数组长度为: ',n,'。耗费时间为: ',d,'s')

```

效果如下:

```

C:\WINDOWS\system32\cmd.exe - "D:\anaconda\condabin\conda.bat" activate py36
19816596232, 0.9992138427727402, 0.9992218310360668, 0.9992260385121099, 0.9992389655236606, 0.9992456306721605, 0.99924
69035130932, 0.999262294038346, 0.9992643964825391, 0.9992647161913409, 0.9992716719993756, 0.9992772773905633, 0.999284
9850883977, 0.9992903418286428, 0.9992920054122866, 0.9992989060081316, 0.9992999443085091, 0.9993218778391556, 0.999335
2084343274, 0.9993355854654309, 0.9993360824758258, 0.99934420828072, 0.9993444925404169, 0.9993455096680048, 0.99935257
81594913, 0.9993536120968493, 0.999361479348995, 0.9993753477782663, 0.999377570979802, 0.9993802029161766, 0.999382203
061094, 0.9993901824745898, 0.9993957977357496, 0.9994058623927078, 0.9994236264781245, 0.9994246406292695, 0.9994263512
083766, 0.9994278785096772, 0.9994300840378161, 0.9994303425971438, 0.9994306476781873, 0.9994354721634497, 0.9994528255
567546, 0.9994565068187456, 0.9994571861435708, 0.9994584856773269, 0.9994599339093155, 0.9994626016348827, 0.9994709823
163891, 0.9994828137999309, 0.9994834676042755, 0.9994937944450586, 0.9994962438833311, 0.9994962552404103, 0.9995022293
61552, 0.9995103055117591, 0.9995146222781333, 0.9995161994469163, 0.9995179104125171, 0.9995207727394688, 0.99953893180
49765, 0.9995392749062607, 0.9995410240687589, 0.9995423891646843, 0.9995455072300005, 0.9995523647458603, 0.99955548502
47511, 0.9995621013632043, 0.9995723773776032, 0.999575947391906, 0.9995779940458828, 0.9995870481195165, 0.999588536574
2075, 0.9995893128715586, 0.9995915806324994, 0.9995924417879015, 0.9995984203537135, 0.9995986808090523, 0.999602206602
974, 0.999605799537539, 0.9996097729689829, 0.999616617525111, 0.9996216168633955, 0.9996218327452108, 0.999626952788888
1, 0.9996356496406927, 0.9996398916666668, 0.9996510722565264, 0.9996662164324596, 0.999666510079288, 0.999667253534051
6, 0.9996865406695832, 0.9996878975895421, 0.9996892166842745, 0.9996894429828457, 0.9996916955283693, 0.999698621099737
7, 0.99970203169625, 0.9997165984625145, 0.999731653888636, 0.999740447110083, 0.9997477172063287, 0.999764628789762, 0.
9997680872375065, 0.9997702798872363, 0.9997705773161827, 0.9997826655008639, 0.9997844017278446, 0.9997930254538989, 0.
9997949139855274, 0.9997951169145765, 0.9997961109528217, 0.9998072871211906, 0.9998176000251832, 0.9998238357193078, 0.
9998317044177922, 0.9998491085409584, 0.999867294677941, 0.9998820339158915, 0.9998839824321121, 0.999889315531033, 0.99
98902303969223, 0.9998968721820732, 0.9999015733205274, 0.9999016749337362, 0.9999064500592437, 0.9999085014293999, 0.99
99129808760798, 0.9999168123810569, 0.9999189121129917, 0.9999200179527842, 0.9999258785597517, 0.9999265070208032, 0.99
99277420630949, 0.9999305638048624, 0.9999309501841463, 0.9999317083846953, 0.9999318600659886, 0.9999329973375544, 0.99
99336418015928, 0.9999352634834469, 0.9999377887684203, 0.999943430936011, 0.999946213926785, 0.9999507102707288, 0.9999
539889638621, 0.9999567408574115, 0.9999584281417657, 0.9999701937191084, 0.9999724086940349, 0.9999750131310812, 0.9999
779355310273, 0.9999799512946249, 0.9999906260567814, 0.999990890673181, 0.9999975838205656, 0.9999981308011757, 0.99999
9978135828]
数组长度为: 200000 。耗费时间为: 1.2514216899871826 s
(py36) D:\子钰\文案\个人\研究生\研一\计算机算法设计与分析\src>

```

2.分别用长为 10000、30000、80000、100000、200000 的数组来测试算法复杂度，可得下表：

quicksort:					
数据规模	10000	30000	80000	100000	200000
运行时间(s)	0.0864	0.2587	0.6778	0.9983	2.1318

mergeSort:					
数据规模	10000	30000	80000	100000	200000
运行时间(s)	0.1310	0.5125	2.2429	3.1958	11.0189

3.讨论归并排序算法 mergeSort 的空间复杂度

设 $S(n)$ 为 n 个长度数组排序所用空间。

则每次分组所需要的内存空间为 $S(\frac{n}{2})$ ，同时，合并两个组需要的内存空间为 $O(n) \approx 2n$

故 可 得 $S(n) \leq S(\frac{n}{2}) + O(n) \leq S(\frac{n}{4}) + O(n) + O(\frac{n}{2}) \leq \dots \leq S(1) + O(\frac{n}{2^k}) + \dots + O(n) \leq O(2n) = O(n)$

又因为数组长度为 n ，也为 $O(n)$ ，故空间复杂度为 $O(n)$

4.说明算法 PartSelect 的平均时间复杂性为 $O(n)$

证明：令 $C_A^k(n)$ 表示 n 个元素的数组 A 中寻找第 k 小个元素的平均时间复杂度，因为 partition(0,n-1)的时间复杂度是 $O(n)$ ，因此存在常数 c ，使得：

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

令 $R(n) = \max_k (C_A^k(n))$ ，设 $k = k_n$ 时满足。则 $R(n)$ 有：

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

取 $C \geq R(1)$, 当 $n = 1$ 时, 取 $c \geq \frac{C}{4}$, 则 $R(1) \leq 4c$

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

对于 $n = k - 1$, 假设 $R(k - 1) \leq 4c(k - 1)$ 成立, 则:

$$\begin{aligned} R(k) &\leq ck + \frac{1}{k} (R(k-1) + \dots + R(k-k_n+1) + R(k_n) \dots + R(k-1)) \\ &\leq ck + \frac{4c}{k} ((k-1) + \dots + (k-k_n+1) + k_n \dots (k-1)) \\ &\leq ck + \frac{4c}{k} \left(\frac{(k_n-1)(2k-k_n)}{2} + \frac{(k-k_n)(k_n+k-1)}{2} \right) \\ &= ck - \frac{4c}{k} \left(k_n^2 - (k+1)k_n - \frac{k^2-3k}{2} \right) \\ &\leq ck - \frac{4c}{k} \left(-\left(\frac{k+1}{2}\right)^2 - \frac{k^2-3k}{2} \right) \\ &= ck + c \left(\frac{3k^2-4k+1}{k} \right) \\ &\leq ck + c(3k-3) \\ &\leq 4ck \end{aligned}$$

故由数学归纳法, 对于任意 $n \in \mathcal{N}$, 均满足:

$$R(n) \leq 4cn$$

故其时间复杂度为 $O(n)$