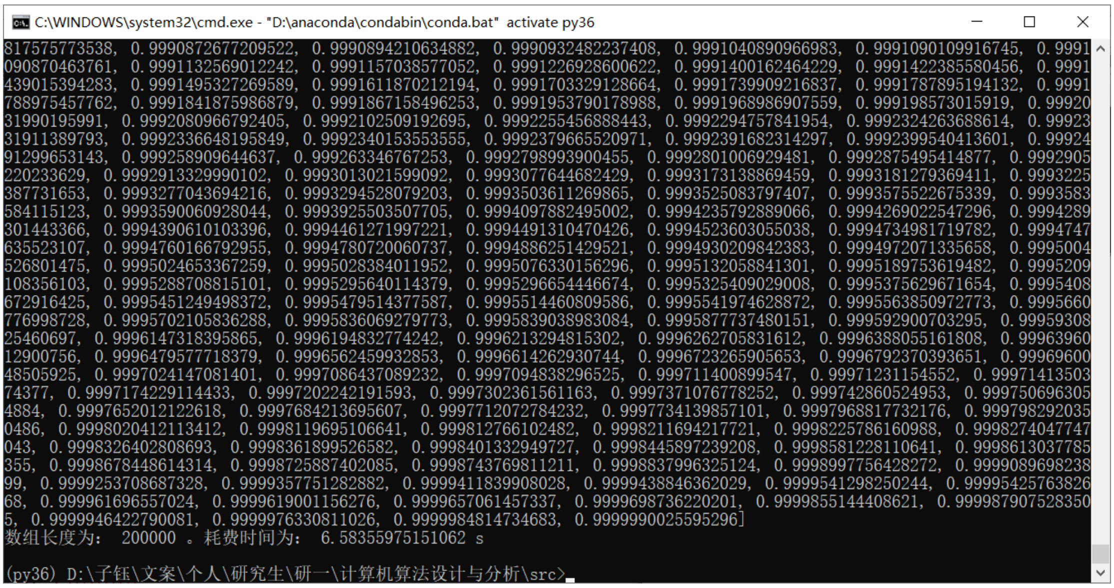
1.实现算法 mergeSort.py 如下：

```
#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：

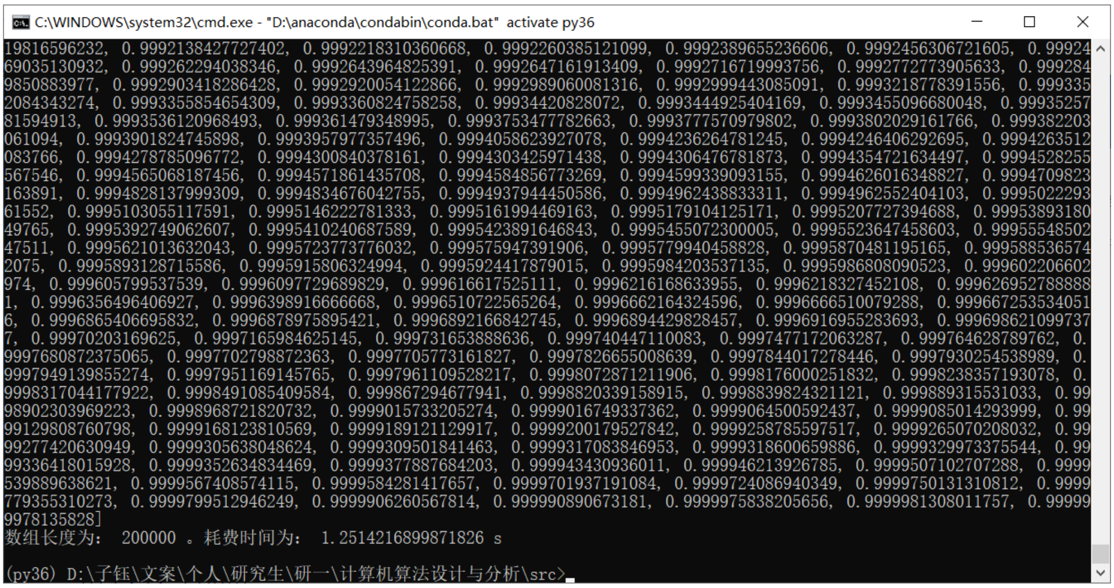


编写 quickSort.py 如下：

```
#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):
    privot = left#基准值
    index=privot+1
    for i in range(index,right+1):
        if arr[i]<arr[privot]:
            swap(arr,index,i)
            index+=1
    swap(arr,index-1,privot)#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了
    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
```

```python
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：



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$S\left(\frac{n}{2}\right)$，同时，合并两个组需要的内存空间为$O(n)\approx 2n$

故可得 $S(n)\le S\left(\frac{n}{2}\right)+O(n)\le S\left(\frac{n}{4}\right)+O(n)+O\left(\frac{n}{2}\right)\le\cdots\le S(1)+O\left(\frac{n}{2^k}\right)+\cdots+O(n)\le O(2n)=O(n)$

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：

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

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

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

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

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

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

$$\begin{aligned}
R(k) &\le \mathrm{c}\mathrm{k} + \frac{1}{\mathrm{k}}\big(R(k-1) + \cdots + R(k-k_n+1) + R(k_n) \ldots + R(k-1)\big) \\
&\le ck + \frac{4c}{k}\big((k-1) + \cdots + (k-k_n+1) + k_n \ldots (k-1)\big) \\
&\le ck + \frac{4c}{k}\left(\frac{(k_n-1)(2k-k_n)}{2} + \frac{(k-k_n)(k_n+k-1)}{2}\right) \\
&= ck - \frac{4c}{k}\left(k_n^2 - (k+1)k_n - \frac{k^2-3k}{2}\right) \\
&\le ck - \frac{4c}{k}\left(-\left(\frac{k+1}{2}\right)^2 - \frac{k^2-3k}{2}\right) \\
&= ck + c\left(\frac{3k^2-4k+1}{k}\right) \\
&\le ck + c(3k-3) \\
&\le 4ck
\end{aligned}$$

故由数学归纳法，对于任意$n \in \mathcal{N}$，均满足：

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

故其时间复杂度为$O(n)$