11624_W13_2022076762_변경민

P1. qSort 구현하기

In [1]:

```
# quick sort

import random, time
listLength = 1000000

def getList(listLength):
    listA = list(range(0, listLength))
    random.shuffle(listA)
    return listA

def qSort(listA, start, end):
    if start < end:
        pvt, listA = partition(listA, start, end)
        qSort(listA, start, pvt-1)
        qSort(listA, pvt+1, end)
    return listA

def partition(listA, start, end):
    pvt = listA[end]
    pvtIdx = end

    while start < end:
        if listA[start] <= pvt:
            start = start + 1
        else:
            listA[start], listA[end-1] = listA[end-1], listA[start]
            end = end - 1
            end = end - 1

    listA[pvtIdx], listA[start] = listA[start], listA[pvtIdx]

    return start, listA

unsortedList = getList(listLength)
start = time.time()
sortedList = qSort(unsortedList, 0, len(unsortedList)-1)
print('정렬에 걸린 시간(s)', time.time() - start)
```

```
정렬에 걸린 시간(s) 4.475986003875732
```

P2. 빠른 정렬 수행해보기

n	1,000	10,000	40,000	80,000	160,000	320,000	640,000
선택정렬	0.044	3.765	64.205	252.151	x	x	x
삽입정렬	0.042	3.818	57.59	228.781	x	x	x
합병정렬	0.005	0.063	0.303	0.896	3.018	10.956	44.793
퀵정렬	0.002	0.027	0.124	0.279	0.569	1.311	2.662

In [14]:

```
# selection sort
import random
import time
def measureTime(n):
    listA = list(range(0,n))

    random.shuffle(listA)
#     print('정렬전:',listA)

    start = time.time()

    for i in range(len(listA)-1):
        min_idx = i
        for j in range(i+1, len(listA)):
            if listA[min_idx] > listA[j]:
                min_idx = j
        tmp = listA[i]
        listA[i] = listA[min_idx]
        listA[min_idx] = tmp
    print('|', round(time.time()-start,3), end = '')

measureTime(1000)
measureTime(10000)
measureTime(40000)
measureTime(80000)
```

```
| 0.044| 3.765| 64.205| 252.151
```

In [16]:

```
# insertion Sort
import random
import time

def measureTime(n):
    listA = list(range(0,n))
    random.shuffle(listA)

    start = time.time()

    for i in range(len(listA)-1):
        key = listA[i]
        j = i - 1
        while j >= 0 and listA[j] > key:
            listA[j+1] = listA[j]
            j = j - 1
        listA[j+1] = key

    print('|', round(time.time()-start,3), end = '')

measureTime(1000)
measureTime(10000)
measureTime(40000)
measureTime(80000)
```

```
| 0.042| 3.818| 57.59| 228.781
```

In [8]:

```
# mergeSort
import random
import time

def sortNList(n):
    l = list(range(n))
    random.shuffle(l)
    start = time.time()
```

```python
    sorted_l = mergeSort(l)
    print('|', round(time.time()-start,3), end = '')

def mergeSort(l):
    if len(l) <= 1:
        return l
    mid = len(l)//2
    left = mergeSort(l[:mid])
    right = mergeSort(l[mid:])
    return merge(left, right)

def merge(left, right):
    merged = list()
    while( len(left)> 0) or (len(right) > 0):
        if len(right) <= 0:
            return merged + left
        elif len(left) <= 0:
            return merged + right
        else:
            if left[0] <= right[0]:
                merged.append(left[0])
                del left[0]
            else:
                merged.append(right[0])
                del right[0]
    return merged

sortNList(1000)
sortNList(10000)
sortNList(40000)
sortNList(80000)
sortNList(160000)
sortNList(320000)
sortNList(640000)
```

```
| 0.005| 0.063| 0.303| 0.896| 3.018| 10.956| 44.793
```

In [15]:

```
# quickSort

import random, time

def getList(listLength):
    listA = list(range(0, listLength))
    random.shuffle(listA)
    return listA

def qSort(listA, start, end):
    if start < end:
        pvt, listA = partition(listA, start, end)
        qSort(listA, start, pvt-1)
        qSort(listA, pvt+1, end)
    return listA

def partition(listA, start, end):
    pvt = listA[end]
    pvtIdx = end

    while start < end:
        if listA[start] <= pvt:
            start = start + 1
        else:
            listA[start], listA[end-1] = listA[end-1], listA[start]
            end = end - 1

    listA[pvtIdx], listA[start] = listA[start], listA[pvtIdx]

    return start, listA

def sortNList(listLength):
    unsortedList = getList(listLength)
    start = time.time()
    sortedList = qSort(unsortedList, 0, len(unsortedList)-1)
    sortedList = qSort(unsortedList, 0, len(unsortedList)-1)
    print('|', round(time.time()-start,3), end = '')

sortNList(1000)
sortNList(10000)
sortNList(40000)
sortNList(80000)
sortNList(160000)
sortNList(320000)
sortNList(640000)
```

```
| 0.002| 0.027| 0.124| 0.279| 0.569| 1.311| 2.662
```

P3. 계수 정렬 (Counting sort)

In [55]:

```python
#countSort
import random

def getRandomList():
    l = list(range(100))
    return random.sample(l, 10)

def getMixedRandomList(l):
    random.shuffle(l)
    return l

unsortedList = getMixedRandomList(getRandomList()*100)
countList = [0]*100
sortedList = []

n = 0
while n < len(unsortedList):
    countList[unsortedList[n]] = countList[unsortedList[n]] + 1
    n = n + 1

n = 0
while n < len(unsortedList):
    while n < len(countList) and countList[n] > 0:
        sortedList.append(n)
        countList[n] = countList[n] - 1
    n = n + 1

print(sortedList)
```

```
[2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2
, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 29, 29,
29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29,
29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29,
29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29,
29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 32, 32, 32, 32, 32, 32,
32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32,
32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32,
32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32,
32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 48, 48, 48, 48, 48, 48, 48, 48, 48, 48,
48, 48, 48, 48, 48, 48, 48, 48, 48, 48, 48, 48, 48, 48, 48, 48, 48, 48, 48, 48, 48, 48, 48, 48, 48, 48,
48, 48, 48, 48, 48, 48, 48, 48, 48, 48, 48, 48, 48, 48, 48, 48, 48, 48, 48, 48, 48, 48, 48, 48, 48, 48,
48, 48, 48, 48, 48, 48, 48, 48, 48, 48, 48, 48, 48, 48, 48, 48, 48, 48, 48, 48, 48, 48, 48, 48, 48, 48,
48, 48, 48, 48, 48, 48, 48, 48, 48, 48, 48, 48, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50,
50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50,
50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50,
50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50,
50, 50, 50, 50, 50, 50, 50, 50, 51, 51, 51, 51, 51, 51, 51, 51, 51, 51, 51, 51, 51, 51, 51, 51, 51, 51,
51, 51, 51, 51, 51, 51, 51, 51, 51, 51, 51, 51, 51, 51, 51, 51, 51, 51, 51, 51, 51, 51, 51, 51, 51, 51,
51, 51, 51, 51, 51, 51, 51, 51, 51, 51, 51, 51, 51, 51, 51, 51, 51, 51, 51, 51, 51, 51, 51, 51, 51, 51,
51, 51, 51, 51, 51, 51, 51, 51, 51, 51, 51, 51, 51, 51, 51, 51, 51, 51, 51, 51, 51, 51, 51, 51, 51, 51,
51, 51, 51, 51, 52, 52, 52, 52, 52, 52, 52, 52, 52, 52, 52, 52, 52, 52, 52, 52, 52, 52, 52, 52, 52, 52,
52, 52, 52, 52, 52, 52, 52, 52, 52, 52, 52, 52, 52, 52, 52, 52, 52, 52, 52, 52, 52, 52, 52, 52, 52, 52,
52, 52, 52, 52, 52, 52, 52, 52, 52, 52, 52, 52, 52, 52, 52, 52, 52, 52, 52, 52, 52, 52, 52, 52, 52, 52,
52, 52, 52, 52, 52, 52, 52, 52, 52, 52, 52, 52, 52, 52, 52, 52, 52, 52, 52, 52, 52, 52, 52, 52, 52, 52,
70, 70, 70, 70, 70, 70, 70, 70, 70, 70, 70, 70, 70, 70, 70, 70, 70, 70, 70, 70, 70, 70, 70, 70, 70, 70,
70, 70, 70, 70, 70, 70, 70, 70, 70, 70, 70, 70, 70, 70, 70, 70, 70, 70, 70, 70, 70, 70, 70, 70, 70, 70,
70, 70, 70, 70, 70, 70, 70, 70, 70, 70, 70, 70, 70, 70, 70, 70, 70, 70, 70, 70, 70, 70, 70, 70, 70, 70,
70, 70, 70, 70, 70, 70, 70, 70, 70, 70, 70, 70, 70, 70, 70, 70, 70, 70, 70, 70, 70, 70, 80, 80, 80, 80,
80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80,
80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80,
80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80,
80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 80, 89, 89, 89, 89, 89, 89, 89, 89,
89, 89, 89, 89, 89, 89, 89, 89, 89, 89, 89, 89, 89, 89, 89, 89, 89, 89, 89, 89, 89, 89, 89, 89, 89, 89,
89, 89, 89, 89, 89, 89, 89, 89, 89, 89, 89, 89, 89, 89, 89, 89, 89, 89, 89, 89, 89, 89, 89, 89, 89, 89,
89, 89, 89, 89, 89, 89, 89, 89, 89, 89, 89, 89, 89, 89, 89, 89, 89, 89, 89, 89, 89, 89, 89, 89, 89, 89,
89, 89, 89, 89, 89, 89, 89, 89, 89, 89, 89, 89, 89, 89]
```