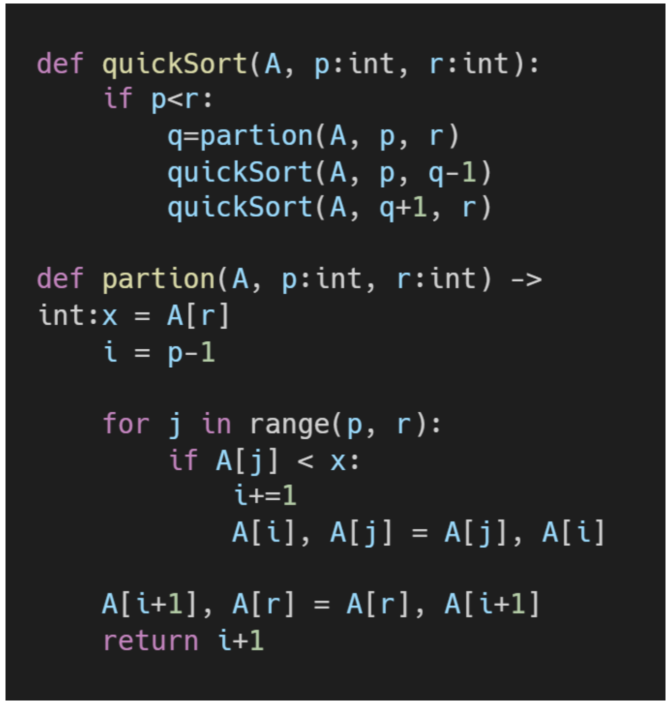
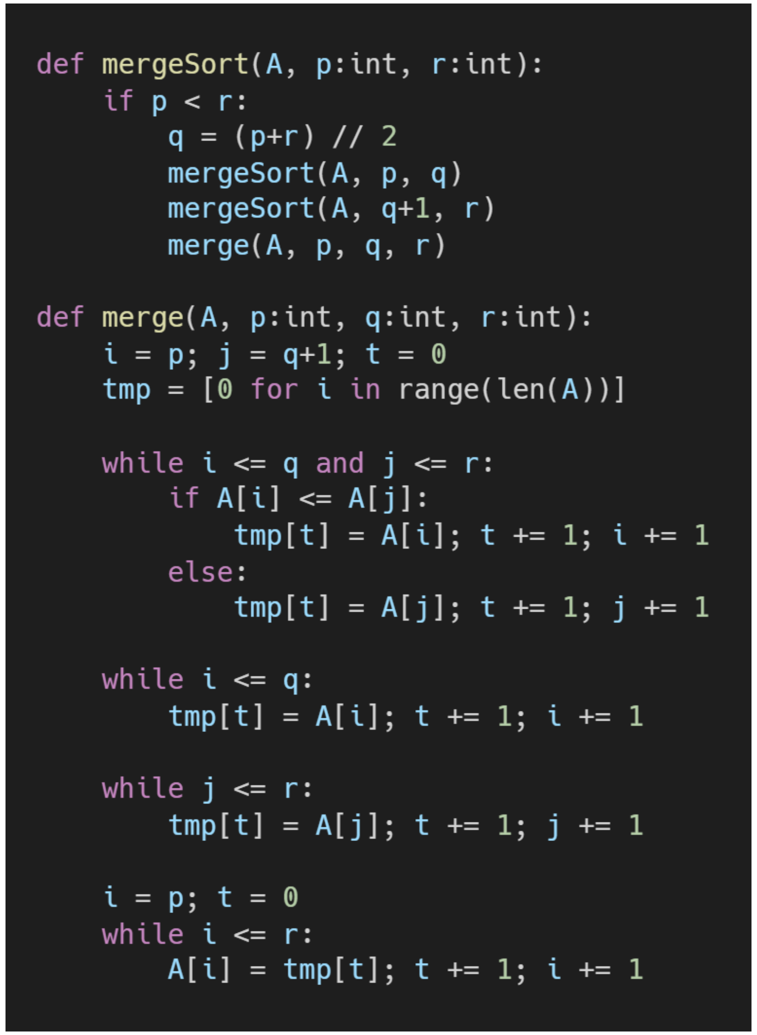


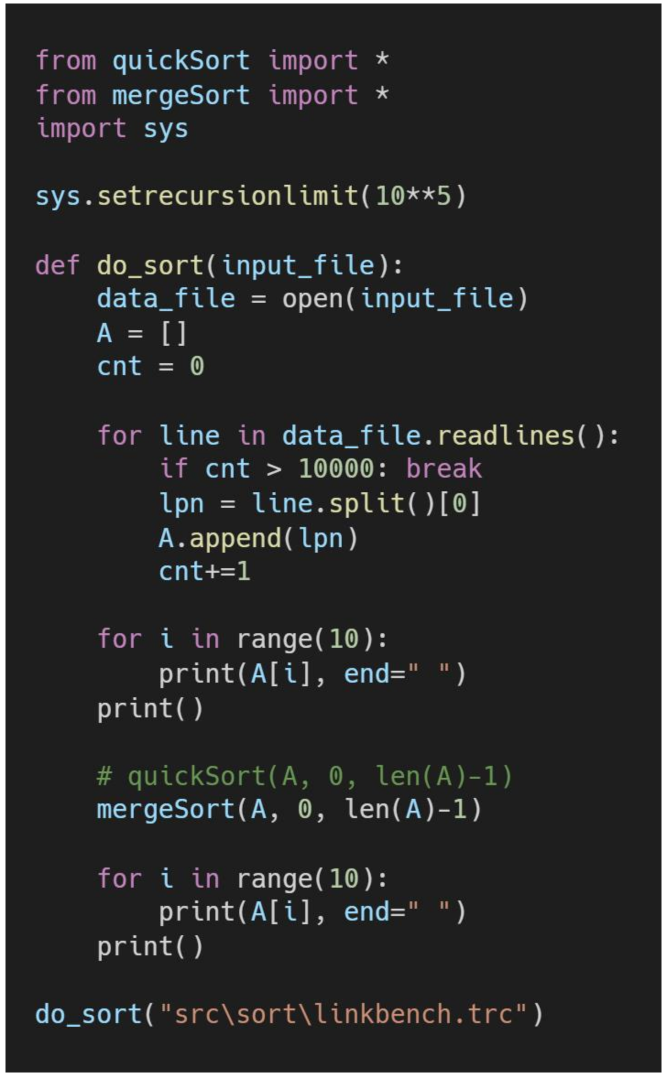
```python
def quickSort(A, p:int, r:int):
    if p<r:
        q=partion(A, p, r)
        quickSort(A, p, q-1)
        quickSort(A, q+1, r)

def partion(A, p:int, r:int) ->
int:x = A[r]
    i = p-1

    for j in range(p, r):
        if A[j] < x:
            i+=1
            A[i], A[j] = A[j], A[i]

    A[i+1], A[r] = A[r], A[i+1]
    return i+1
```

▲ quickSort.py



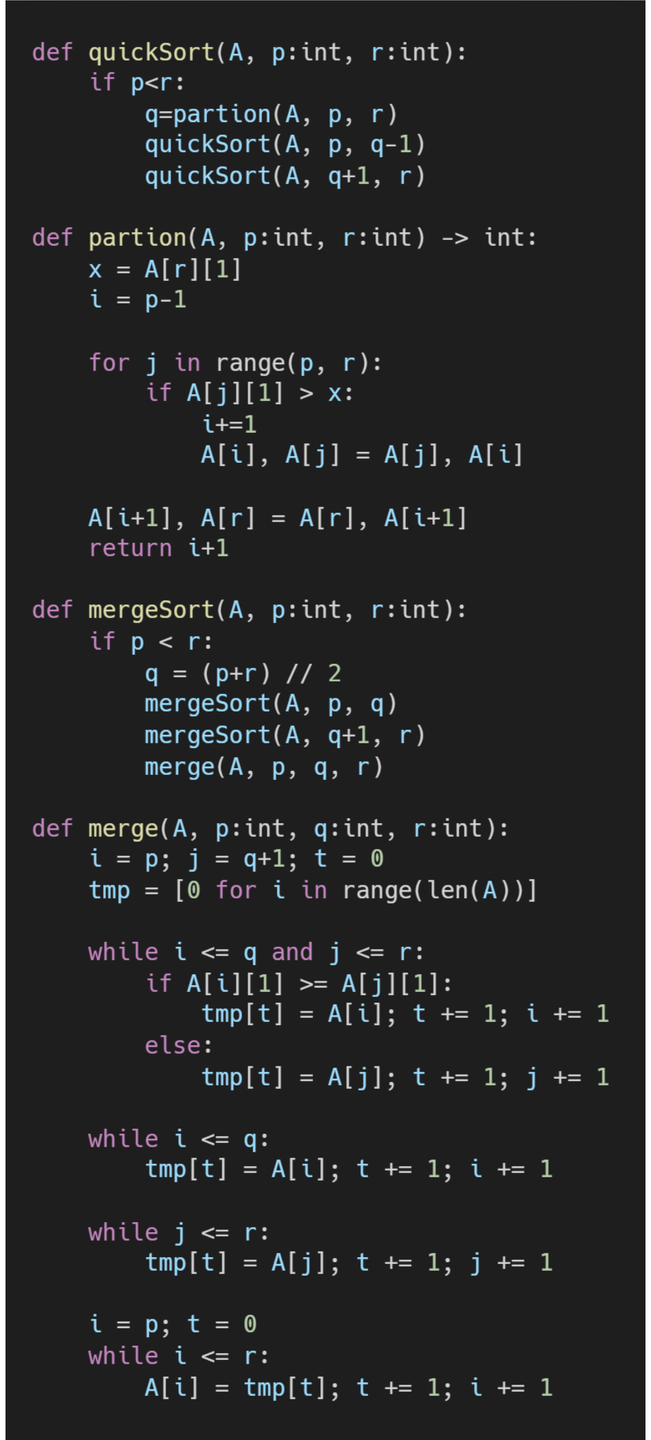
```python
def mergeSort(A, p:int, r:int):
    if p < r:
        q = (p+r) // 2
        mergeSort(A, p, q)
        mergeSort(A, q+1, r)
        merge(A, p, q, r)

def merge(A, p:int, q:int, r:int):
    i = p; j = q+1; t = 0
    tmp = [0 for i in range(len(A))]

    while i <= q and j <= r:
        if A[i] <= A[j]:
            tmp[t] = A[i]; t += 1; i += 1
        else:
            tmp[t] = A[j]; t += 1; j += 1

    while i <= q:
        tmp[t] = A[i]; t += 1; i += 1

    while j <= r:
        tmp[t] = A[j]; t += 1; j += 1

    i = p; t = 0
    while i <= r:
        A[i] = tmp[t]; t += 1; i += 1
```

▲ mergeSort.py



```python
from quickSort import *
from mergeSort import *
import sys

sys.setrecursionlimit(10**5)

def do_sort(input_file):
    data_file = open(input_file)
    A = []
    cnt = 0

    for line in data_file.readlines():
        if cnt > 10000: break
        lpn = line.split()[0]
        A.append(lpn)
        cnt+=1

    for i in range(10):
        print(A[i], end=" ")
    print()

    # quickSort(A, 0, len(A)-1)
    mergeSort(A, 0, len(A)-1)

    for i in range(10):
        print(A[i], end=" ")
    print()

do_sort("src\sort\linkbench.trc")
```

▲ pageSort.py

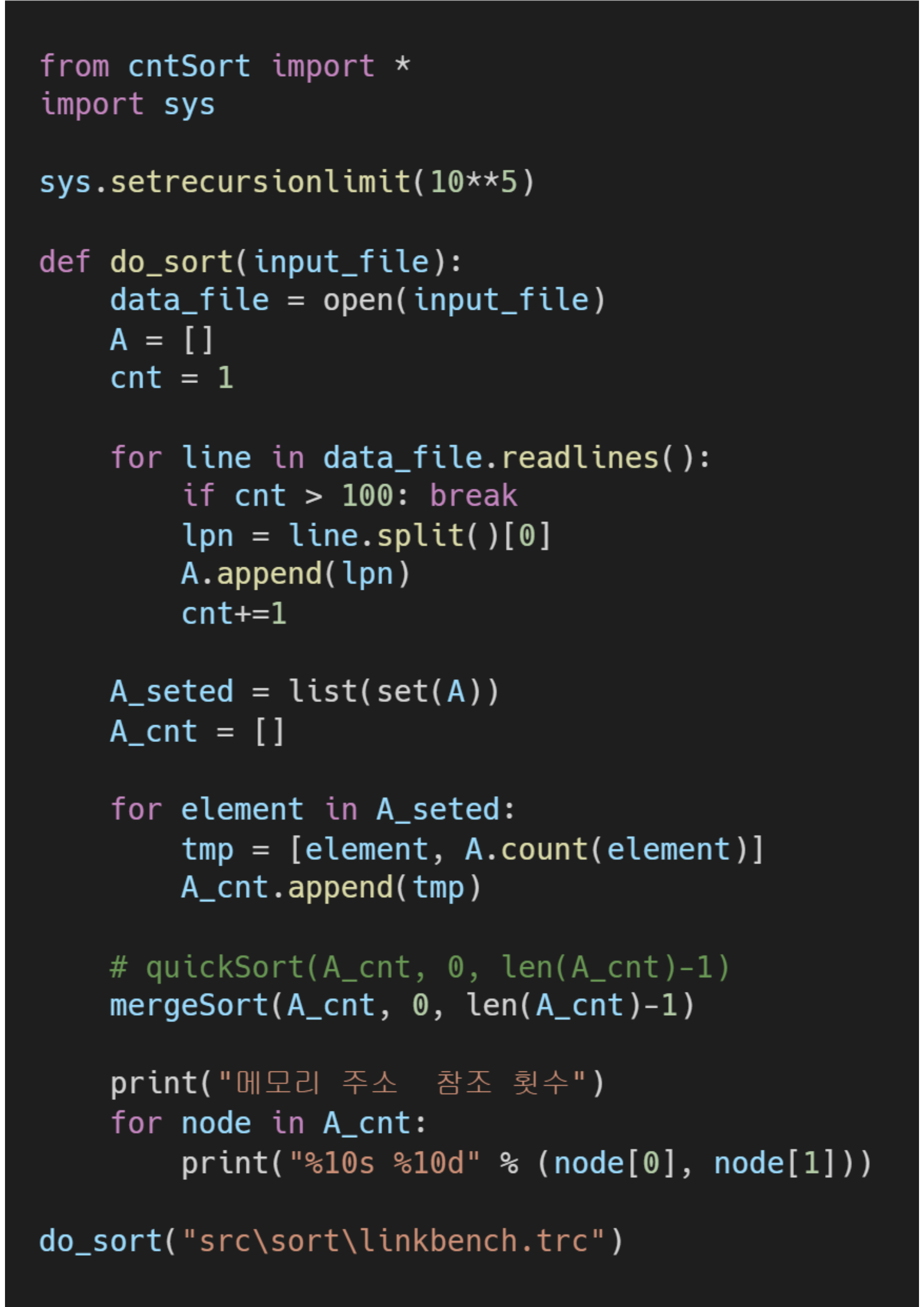


```python
def quickSort(A, p:int, r:int):
    if p<r:
        q=partion(A, p, r)
        quickSort(A, p, q-1)
        quickSort(A, q+1, r)

def partion(A, p:int, r:int) -> int:
    x = A[r][1]
    i = p-1

    for j in range(p, r):
        if A[j][1] > x:
            i+=1
            A[i], A[j] = A[j], A[i]

    A[i+1], A[r] = A[r], A[i+1]
    return i+1

def mergeSort(A, p:int, r:int):
    if p < r:
        q = (p+r) // 2
        mergeSort(A, p, q)
        mergeSort(A, q+1, r)
        merge(A, p, q, r)

def merge(A, p:int, q:int, r:int):
    i = p; j = q+1; t = 0
    tmp = [0 for i in range(len(A))]

    while i <= q and j <= r:
        if A[i][1] >= A[j][1]:
            tmp[t] = A[i]; t += 1; i += 1
        else:
            tmp[t] = A[j]; t += 1; j += 1

    while i <= q:
        tmp[t] = A[i]; t += 1; i += 1

    while j <= r:
        tmp[t] = A[j]; t += 1; j += 1

    i = p; t = 0
    while i <= r:
        A[i] = tmp[t]; t += 1; i += 1
```

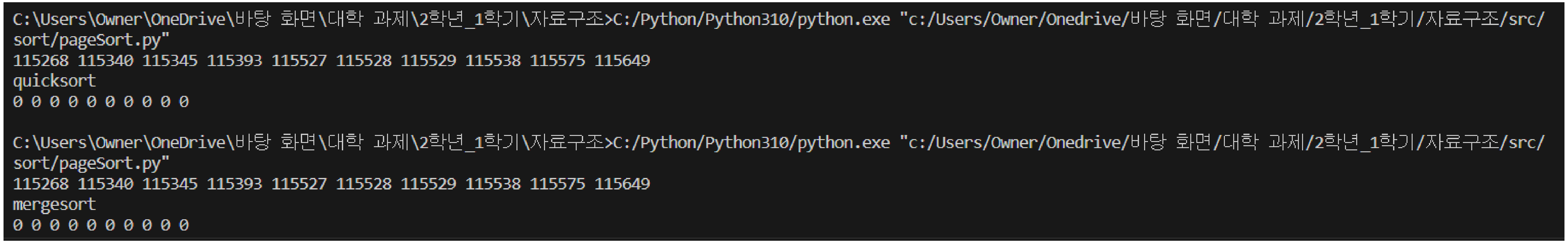
◀ cntSort.py

cnt 기준으로 정렬하기
위해 sort 코드들을 변경

pageSortcout.py ▶

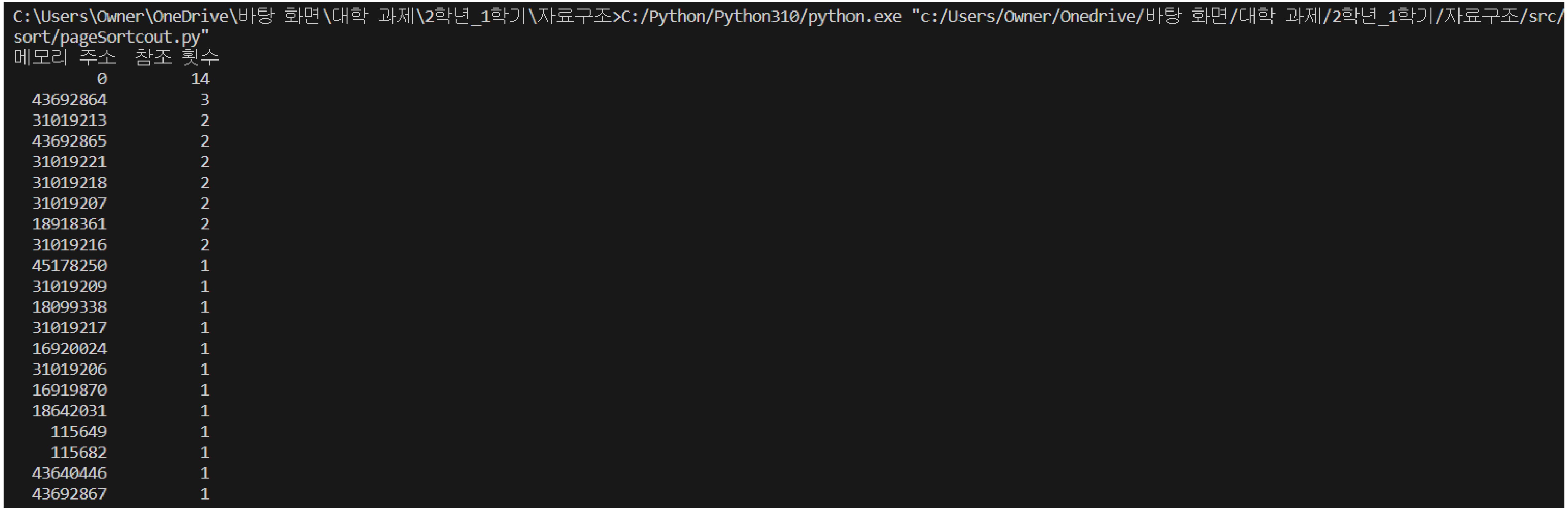


```python
from cntSort import *
import sys

sys.setrecursionlimit(10**5)

def do_sort(input_file):
    data_file = open(input_file)
    A = []
    cnt = 1

    for line in data_file.readlines():
        if cnt > 100: break
        lpn = line.split()[0]
        A.append(lpn)
        cnt+=1

    A_seted = list(set(A))
    A_cnt = []

    for element in A_seted:
        tmp = [element, A.count(element)]
        A_cnt.append(tmp)

    # quickSort(A_cnt, 0, len(A_cnt)-1)
    mergeSort(A_cnt, 0, len(A_cnt)-1)

    print("메모리 주소  참조 횟수")
    for node in A_cnt:
        print("%10s %10d" % (node[0], node[1]))

do_sort("src\sort\linkbench.trc")
```



```
C:\Users\Owner\OneDrive\바탕 화면\대학 과제\2학년_1학기\자료구조>C:/Python/Python310/python.exe "c:/Users/Owner/Onedrive/바탕 화면/대학 과제/2학년_1학기/자료구조/src/
sort/pageSort.py"
115268 115340 115345 115393 115527 115528 115529 115538 115575 115649
quicksort
0 0 0 0 0 0 0 0 0 0

C:\Users\Owner\OneDrive\바탕 화면\대학 과제\2학년_1학기\자료구조>C:/Python/Python310/python.exe "c:/Users/Owner/Onedrive/바탕 화면/대학 과제/2학년_1학기/자료구조/src/
sort/pageSort.py"
115268 115340 115345 115393 115527 115528 115529 115538 115575 115649
mergesort
0 0 0 0 0 0 0 0 0 0
```

▲ pageSort.py (quicksort, mergesort) 실행결과



```
C:\Users\Owner\OneDrive\바탕 화면\대학 과제\2학년_1학기\자료구조>C:/Python/Python310/python.exe "c:/Users/Owner/Onedrive/바탕 화면/대학 과제/2학년_1학기/자료구조/src/
sort/pageSortcout.py"
메모리 주소  참조 횟수
        0        14
  43692864        3
  31019213        2
  43692865        2
  31019221        2
  31019218        2
  31019207        2
  18918361        2
  31019216        2
  45178250        1
  31019209        1
  18099338        1
  31019217        1
  16920024        1
  31019206        1
  16919870        1
  18642031        1
    115649        1
    115682        1
  43640446        1
  43692867        1
```

▲ pageSortcout.py (페이지 접근 횟수 기준 정렬) 실행결과