

알고리즘 실습 보고서

-QuickSort-

전공 :컴퓨터공학과

분반 : 05반

학번 :201701988

이름 :김수빈

1. 실행 환경

본 실습은 Windows10 64bit, jdk 1.8.0_221, Eclipse EE가 설치된 환경에서 실행되었다

2. 과제 설명

- 출제된 과제에 대한 설명

이번 과제는 data05.txt 파일을 읽어 원소들을 QuickSort로 정렬하는 것이다.

pivot을 배열의 끝에 있는 원소로 하는 partition과 random하게 pivot을 정하는 randomizedPartition,

3. 문제 해결 방법

- 문제를 해결하기 위해 자신이 사용한 방법, 아이디어에 대한 설명

data05.txt 파일을 읽어 ,를 기준으로 나눠 동적 배열 numbers에 삽입한다. 이 배열은 quickSort 메소드를 호출할 때 사용할 배열이다. quickSort_withRandom을 호출할 배열은 이 numbers를 clone하여 humbersRandom에 저장한다.

각각 quickSort 메소드와 quickSort_withRandom를 호출한 결과 배열을 file_output에 넣어, 출력할 String을 저장하여, data05_sorted.txt의 String과 hashCode가 같은지 확인한다.

이때 file_output은 ArrayList를 출력하는 메소드로, 파일명과 ArrayList를 매개변수로 받아 파일에 출력하는 String을 반환한다.

▶ partition 메소드

pivot을 r 으로 하여 인덱스 pivot의 값보다 작거나 같은 값을 갖는 경우, pivot 앞에, pivot의 값보다 큰 경우 pivot 뒤에 위치하도록 한다.

l을 p-1로 초기화 한다. 인덱스 j = p부터 r-1까지 다음을 반복한다.

인덱스 j에 있는 값이 pivot에 있는 값보다 작거나 같은 경우 l을 증가시키고 l와 j에 있는 값을 swap한다. 마지막으로 pivot에 있는 값을 l에 있는 값과 swap하여 pivot을 기준으로 pivot 위치의 값보다 작은 값, pivot의 값, pivot의 값보다 큰 값으로 구분되도록 한다. 인덱스 l를 리턴한다.

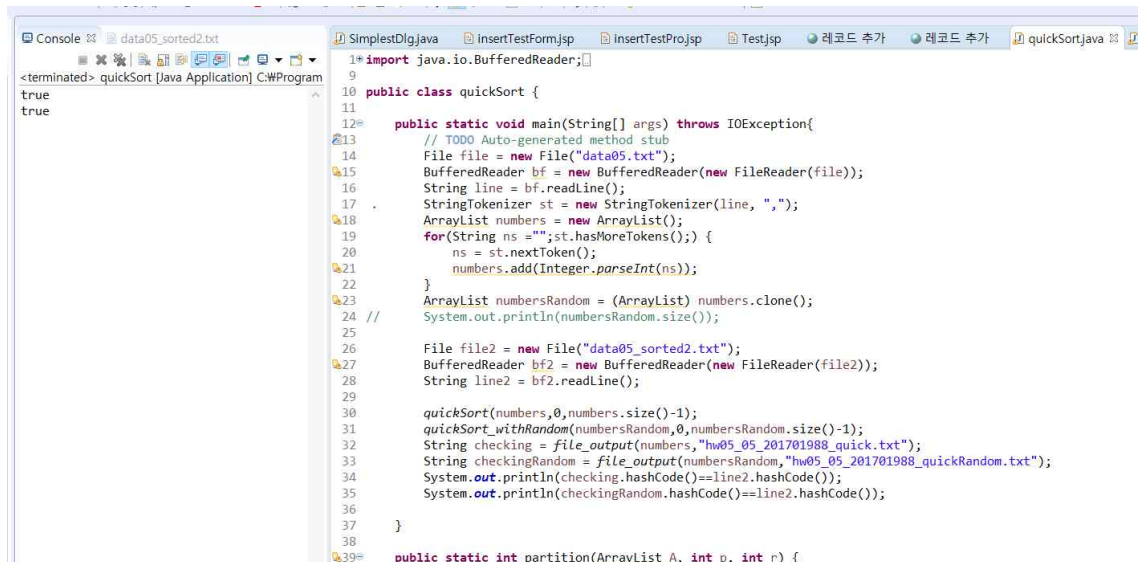
▶ randomizedPartition 메소드

randomizedPartition 메소드는 pivot을 r로 하지 않고, p부터 r까지 random하게 받아 l에 저장한 후, l와 r을 swap하여 partition을 호출한다. 결과적으로 l를 pivot으로 하여 구분한 결과이다.

▶ quick_sort 메소드 & quickSort_withRandom 메소드

quick_sort 메소드는 r이 p보다 크지 않은 경우 종료하고, partition을 통해 받은 pivot을 기준으로 배열을 나누어 각각 quick_sort를 호출한다. 각각 호출된 quick_sort는 또 partition을 진행하고, 다시 배열을 나누면 정렬될 때까지 호출된다. quickSort_withRandom은 r이 p보다 크지 않은 경우 종료하고, randomizedPartition을 통해 받은 pivot을 기준으로 배열을 나누어 각각 quickSort_withRandom을 호출한다.

4. 결과 화면



The screenshot shows an IDE with two windows. The top window, 'SimpleDlg.java', contains a Java program for a quick sort algorithm. It reads numbers from 'data05.txt', sorts them, and writes the sorted numbers to 'data05_sorted2.txt'. The bottom window, 'Console', shows the output of the program, which is a long list of sorted numbers.

```
1* import java.io.BufferedReader;
9
10 public class quickSort {
11
12     public static void main(String[] args) throws IOException{
13         // TODO Auto-generated method stub
14         File file = new File("data05.txt");
15         BufferedReader bf = new BufferedReader(new FileReader(file));
16         String line = bf.readLine();
17         StringTokenizer st = new StringTokenizer(line, ",");
18         ArrayList numbers = new ArrayList();
19         for(String ns = ""; st.hasMoreTokens();){
20             ns = st.nextToken();
21             numbers.add(Integer.parseInt(ns));
22         }
23         ArrayList numbersRandom = (ArrayList) numbers.clone();
24         System.out.println(numbersRandom.size());
25
26         File file2 = new File("data05_sorted2.txt");
27         BufferedReader bf2 = new BufferedReader(new FileReader(file2));
28         String line2 = bf2.readLine();
29
30         quickSort(numbers,0,numbers.size()-1);
31         quickSort_withRandom(numbersRandom,0,numbersRandom.size()-1);
32         String checking = file_output(numbers,"hw05_05_201701988_quick.txt");
33         String checkingRandom = file_output(numbersRandom,"hw05_05_201701988_quickRandom.txt");
34         System.out.println(checking.hashCode()==line2.hashCode());
35         System.out.println(checkingRandom.hashCode()==line2.hashCode());
36
37     }
38
39     public static int partition(ArrayList A, int p, int r) {
```

Console Output:

```
<terminated> quickSort [Java Application] C:\Program
true
```

(data05_sorted와 출력할 파일의 String을 hashCode 비교한 화면입니다.)



The screenshot shows two text files. The top file, 'hw05_05_201701988_quick - Windows 메모장', contains a long list of sorted numbers. The bottom file, 'hw05_05_201701988_quickRandom - Windows 메모장', contains the same list of numbers, but each number is followed by its hashCode value, separated by a space. The files are used to compare the hashCode of the sorted numbers with the hashCode of the original numbers to verify the sorting process.

hw05_05_201701988_quick - Windows 메모장

```
0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,
3,284,285,286,287,288,289,290,291,292,293,294,295,296,297,298,299,300,301,302,303,304,305,306,307,308,309,310,311,312,313,314,315,316,317,318,319,320,3
9,540,541,542,543,544,545,546,547,548,549,550,551,552,553,554,555,556,557,558,559,560,561,562,563,564,565,566,567,568,569,570,571,572,573,574,575,576,5
5,796,797,798,799,800,801,802,803,804,805,806,807,808,809,810,811,812,813,814,815,816,817,818,819,820,821,822,823,824,825,826,827,828,829,830,831,832,8
041,1042,1043,1044,1045,1046,1047,1048,1049,1050,1051,1052,1053,1054,1055,1056,1057,1058,1059,1060,1061,1062,1063,1064,1065,1066,1067,1068,1069,11
1246,1247,1248,1249,1250,1251,1252,1253,1254,1255,1256,1257,1258,1259,1260,1261,1262,1263,1264,1265,1266,1267,1268,1269,1270,1271,1272,1273,1274,
,1451,1452,1453,1454,1455,1456,1457,1458,1459,1460,1461,1462,1463,1464,1465,1466,1467,1468,1469,1470,1471,1472,1473,1474,1475,1476,1477,1478,1479,
5,1656,1657,1658,1659,1660,1661,1662,1663,1664,1665,1666,1667,1668,1669,1670,1671,1672,1673,1674,1675,1676,1677,1678,1679,1680,1681,1682,1683,1684
60,1861,1862,1863,1864,1865,1866,1867,1868,1869,1870,1871,1872,1873,1874,1875,1876,1877,1878,1879,1880,1881,1882,1883,1884,1885,1886,1887,1888,188
065,2066,2067,2068,2069,2070,2071,2072,2073,2074,2075,2076,2077,2078,2079,2080,2081,2082,2083,2084,2085,2086,2087,2088,2089,2090,2091,2092,2093,21
2270,2271,2272,2273,2274,2275,2276,2277,2278,2279,2280,2281,2282,2283,2284,2285,2286,2287,2288,2289,2290,2291,2292,2293,2294,2295,2296,2297,2298,
,2475,2476,2477,2478,2479,2480,2481,2482,2483,2484,2485,2486,2487,2488,2489,2490,2491,2492,2493,2494,2495,2496,2497,2498,2499,2500,2501,2502,2503,
```

hw05_05_201701988_quickRandom - Windows 메모장

```
0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,
3,284,285,286,287,288,289,290,291,292,293,294,295,296,297,298,299,300,301,302,303,304,305,306,307,308,309,310,311,312,313,314,315,316,317,318,319,320,3
9,540,541,542,543,544,545,546,547,548,549,550,551,552,553,554,555,556,557,558,559,560,561,562,563,564,565,566,567,568,569,570,571,572,573,574,575,576,5
5,796,797,798,799,800,801,802,803,804,805,806,807,808,809,810,811,812,813,814,815,816,817,818,819,820,821,822,823,824,825,826,827,828,829,830,831,832,8
041,1042,1043,1044,1045,1046,1047,1048,1049,1050,1051,1052,1053,1054,1055,1056,1057,1058,1059,1060,1061,1062,1063,1064,1065,1066,1067,1068,1069,11
1246,1247,1248,1249,1250,1251,1252,1253,1254,1255,1256,1257,1258,1259,1260,1261,1262,1263,1264,1265,1266,1267,1268,1269,1270,1271,1272,1273,1274,
,1451,1452,1453,1454,1455,1456,1457,1458,1459,1460,1461,1462,1463,1464,1465,1466,1467,1468,1469,1470,1471,1472,1473,1474,1475,1476,1477,1478,1479,
5,1656,1657,1658,1659,1660,1661,1662,1663,1664,1665,1666,1667,1668,1669,1670,1671,1672,1673,1674,1675,1676,1677,1678,1679,1680,1681,1682,1683,1684
60,1861,1862,1863,1864,1865,1866,1867,1868,1869,1870,1871,1872,1873,1874,1875,1876,1877,1878,1879,1880,1881,1882,1883,1884,1885,1886,1887,1888,188
065,2066,2067,2068,2069,2070,2071,2072,2073,2074,2075,2076,2077,2078,2079,2080,2081,2082,2083,2084,2085,2086,2087,2088,2089,2090,2091,2092,2093,21
2270,2271,2272,2273,2274,2275,2276,2277,2278,2279,2280,2281,2282,2283,2284,2285,2286,2287,2288,2289,2290,2291,2292,2293,2294,2295,2296,2297,2298,
,2475,2476,2477,2478,2479,2480,2481,2482,2483,2484,2485,2486,2487,2488,2489,2490,2491,2492,2493,2494,2495,2496,2497,2498,2499,2500,2501,2502,2503,
```

5. 느낀점 및 고찰

pseudo code가 있어, 구현하는 데에 큰 어려움은 없었습니다.