자료구조 실습 보고서

숙제명: [제10주]Quicksort 클래스
제출일: 2018.05.10
학번/이름: 201404377 / 진승언

<프로그램 설명서>

이번시간에는 퀵 정렬 구현 실습을 하였다. 퀵 정렬은 데이터리스트에서 임의의 값을 골라 pivot이라는 이름을 갖고 이것은 퀵정렬에서 값을 나눌 때 기준이 되는 값이다.(실습문제에서는 가장처음, 즉 맨 왼쪽에 있는 데이터를 pivot으로 설정한다.) 그 다음부터는 아래와 같이 하면 된다.

(알고리즘)

1. pivot보다 큰 값을 pivot index 보다 왼쪽에서 찾고 (큰 값 이 나타날 때까지 left index 를 증가시키도록 한다.)

2. pivot 보다 작은 값을 pivot index 보다 오른쪽에서 찾는다 (작은 값이 나타날 때까지 right index를 감소시키도록 한다.)

3. pivot을 기준으로 값 비교가 완료되었다면 index 결과 left , right 를 비교 해본다. left 값이 right 값 보다 작거나 같다면 분명 pivot 을 기준으로 교환을 해야할 값이 있다는 뜻이 된다.

4. 교환한 뒤 left 인덱스는 증가 right 인덱스는 감소 연산을 수행한다.

5. left 인덱스가 right 인덱스보다 작거나 같다면 계속 반복해서 수행한다.

6. 위 와 같은 과정은 pivot을 기준으로 왼쪽으로 정렬된 list 에서는 최초 Left 값이 감소된 right 보다 작다면 계속 재귀하면 되고, pivot을 기준으로 오른쪽으로 정렬된 list 에서는 최초 Right 값이 증가된 i 값보다 크다면 계속 재귀하면 된다.

```
public int partition(int left, int right) {
    int pivot = left;
    int data = this.array[pivot];
    right++;  //그래야 밑에 do-while문에서 감소시켰을때 데이터있는값부터 시작할 수 있음.
    do {
        //채워넣기
        do {
            left++;
        }while((array[left] < data) && left <= right);  //ㅍ

        do {
            right--;
        }while(array[right] > data && right >= left); //위도
        if(left<right) {
            this.swap(left, right);
        }
    }while(left < right);

    this.swap(pivot , right);
    return right;
}

 public void swap(int a , int b) {
     int tmp;
     tmp = array[a];
     array[a] = array[b];
     array[b] = tmp;

 }
```

알고리즘 과정을 위와 같이 구현하였다. 값을 바꾸는 swap함수는 데이터 값을 배열로 참조하므로 위와 같이 값을 바꾸게 할 수 있었다. partition 메소드가 끝나면 pivot값을 기준으로 pivot 왼쪽은 pivot보다 작은값, 오른쪽은 pivot보다 큰 값으로 정렬이 된다. 이 과정을 반반으로 나눠서 이 함수를 재귀로 돌려서 완벽하게 모든 데이터가 오름차순으로 정렬되게 해야 한다. 그 재귀구현은 밑 사진과 같다.

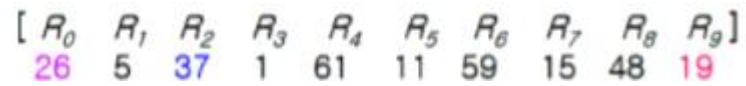
```
public void quickSortRecursively(int left, int right) {
    if(left < right) {
    int mid = partition(left, right);
    quickSortRecursively(left, mid-1);
    quickSortRecursively(mid+1, right);
    }
}
```

left가 right보다 작으면 partition 메소드를 실행해주고 마지막으로 정렬된 인덱스 값을 mid란 변수에 받아서 처음 인덱스에서 mid-1까지와 mid+1에서 마지막인덱스 까지 재귀로 돌려준다.

TestRecursion 클래스는 평소와 같이 구현해주었다.

<실행결과 분석>

예시는 과제PDF파일에 있는 예시로 하였다.



1. add로 데이터를 넣어주었다.

```
1.[Recursive] add
2.[Recursive] sort
3.[Recursive] print
4. 종료
1
추가하고 싶은 내용을 입력하세요 :
26
[26]가 추가 되었습니다.
1.[Recursive] add
2.[Recursive] sort
3.[Recursive] print
4. 종료
1
추가하고 싶은 내용을 입력하세요 :
5
[5]가 추가 되었습니다.
1.[Recursive] add
2.[Recursive] sort
3.[Recursive] print
4. 종료
1
추가하고 싶은 내용을 입력하세요 :
37
[37]가 추가 되었습니다.
1.[Recursive] add
2.[Recursive] sort
3.[Recursive] print
4. 종료
1
추가하고 싶은 내용을 입력하세요 :
1
```

2. 데이터를 다 추가하고 잘 넣어졌는지 print로 확인하고 sort로 정렬한 다음에 다시 print로 확인하였더니 정렬이 잘 수행됐음을 볼 수 있었다.

```
[19]가 추가 되었습니다.
1.[Recursive] add
2.[Recursive] sort
3.[Recursive] print
4. 종료
3
[26][5][37][1][61][11][59][15][48][19]
1.[Recursive] add
2.[Recursive] sort
3.[Recursive] print
4. 종료
2
1.[Recursive] add
2.[Recursive] sort
3.[Recursive] print
4. 종료
3
[1][5][11][15][19][26][37][48][59][61]
1.[Recursive] add
2.[Recursive] sort
3.[Recursive] print
4. 종료
```