정렬 알고리즘 비교 보고서

20213062 이찬우

목차

1. 정렬 알고리즘 코드

1.1. 버블 정렬 코드 (Bubble Sort)

```java
package sortTest;

// 버블 정렬 - 반복적 방법
public class BubbleSort2Test{
    public static void bubbleSort(int[] data){
        for(int j = 0; j < data.length; j++)
            for(int i = 1; i < data.length - j; i++){
                if(data[i - 1] > data[i]) swap(data, i - 1, i);
            }
    }

    public static void swap(int[] data, int s1, int s2){
        int tmp = data[s1];
        data[s1] = data[s2];
        data[s2] = tmp;
    }

    public static void printArray(int[] data){
        for(int key : data){
            System.out.print(key + ", ");
        }
        System.out.println();
    }

    public static void main(String[] args){
        int[] data = {2, 8, 5, 3, 9, 4, 1};
        printArray(data);
        bubbleSort(data);
        printArray(data);
    }
}

```

1.2. 삽입 정렬 코드 (Insertion Sort)

```java
package sortTest;

// 삽입 정렬 - 반복적 방법
public class InsertionSort2Test{
    public static void insertionSort(int[] data){
        int pos = 0; int insert;
        for(int s = 1; s < data.length; s++){
            insert = data[s];
            for(pos = 0; pos < s; pos++)
                if(data[pos] > insert) break;  // find
            for(int i = s; i > pos; i--)
                data[i] = data[i - 1];  // move
            data[pos] = insert;
        }
    }

    public static void printArray(int[] data){
        for(int key : data){
            System.out.print(key + ", ");
        }
        System.out.println();
    }

    public static void main(String[] args){
        int[] data = {2, 8, 5, 3, 9, 4, 1};
        printArray(data);
        insertionSort(data);
        printArray(data);
    }
}

```

1.3. 선택 정렬 코드 (Selection Sort)

```java
package sortTest;

// 선택 정렬 - 반복적 방법
public class SelectionSort2Test{
    public static void selectionSort(int[] data){
        int min_index = 0;
        for(int s = 0; s < data.length - 1; s++){
            min_index = s;
            for(int i = s; i < data.length; i++){
                if(data[i] < data[min_index]) min_index = i;
            }
            swap(data, s, min_index);  // Copy from bubble sort;
        }
    }

    public static void swap(int[] data, int s1, int s2){
        int tmp = data[s1];
        data[s1] = data[s2];
        data[s2] = tmp;
    }

    public static void printArray(int[] data){
        for(int key : data){
            System.out.print(key + ", ");
        }
        System.out.println();
    }

    public static void main(String[] args){
        int[] data = {2, 8, 5, 3, 9, 4, 1};
        printArray(data);
        selectionSort(data);
        printArray(data);
    }
}

```

1.4. 퀵 정렬 코드 (Quick Sort)

```java
package sortTest;

// 퀵 정렬 - 반복적 방법
import java.util.Deque;

public class QuickSort2Test{
    public static void quickSort(int[] data){
        quickSort(data, 0, data.length - 1);
    }

    private static void quickSort(int data[], int start, int end){
        Deque<Integer> stack = new LinkedList<Integer>();
        stack.addLast(start);
        stack.addLast(end);
        while(!stack.isEmpty()){
            end = stack.removeLast();
            start = stack.removeLast();
            int part2 = partition(data, start, end);
            if(start < part2 - 1){
                stack.addLast(start);
                stack.addLast(part2 - 1);
            }
            if(part2 < end){
                stack.addLast(part2);
                stack.addLast(end);
            }
        }
    }

    public static int partition(int[] data, int start, int end){
        int pivot = data[start];
        while(start <= end){
            while(data[start] < pivot) start++;
            while(data[end] > pivot) end--;
            if(start <= end){
                swap(data, start, end);
                start++; end--;
            }
        }
        return start;
    }

    public static void swap(int[] data, int s1, int s2){
        int tmp = data[s1];
        data[s1] = data[s2];
        data[s2] = tmp;
    }

    public static void printArray(int[] data){
        for(int key : data){
            System.out.print(key + ", ");
        }
        System.out.println();
    }

    public static void main(String[] args){
        int[] data = {2, 8, 5, 3, 9, 4, 1};
        printArray(data);
        quickSort(data);
        printArray(data);
    }
}
```

1.5. 병합 정렬 코드 (Merge Sort)

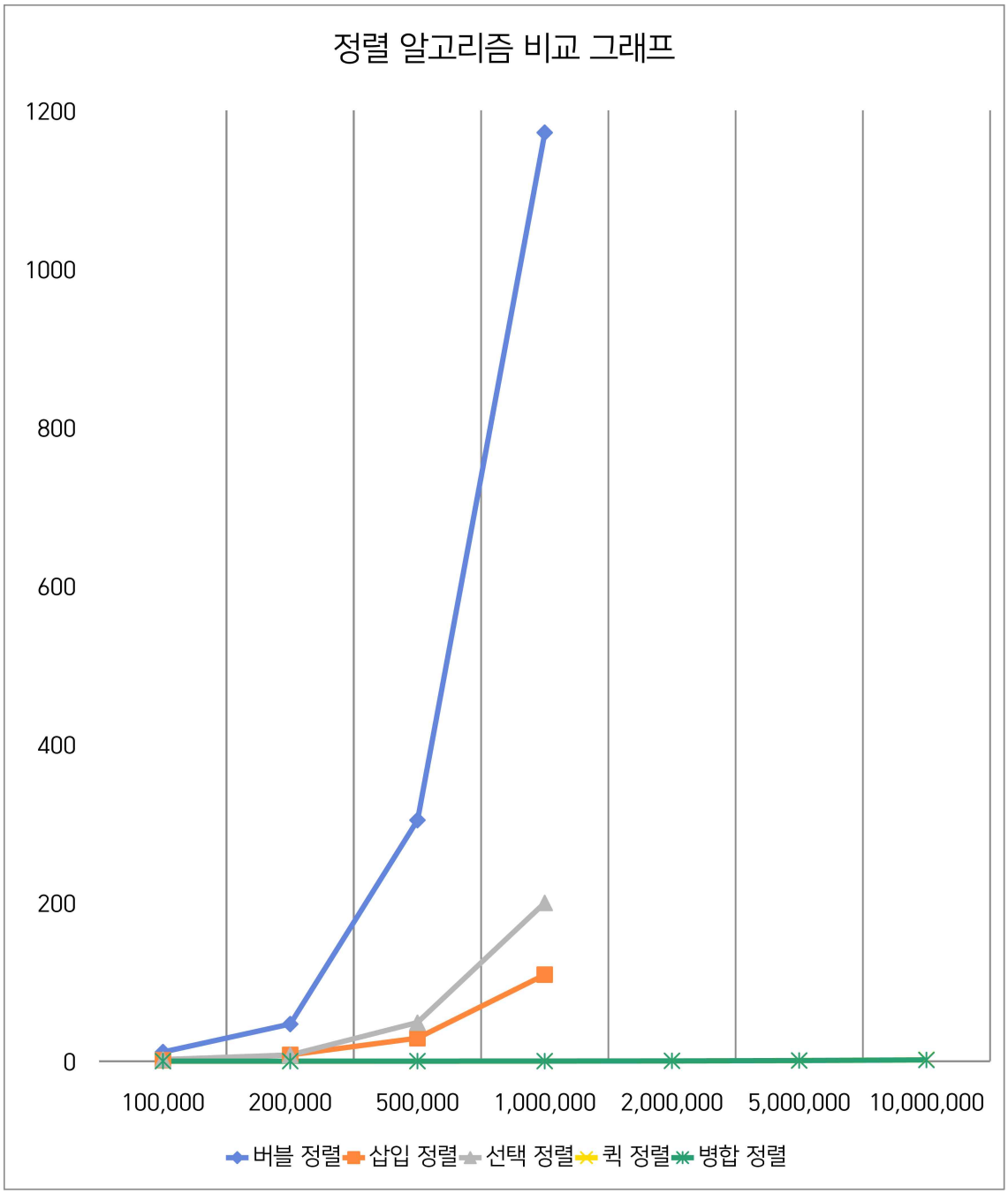
```java
package sortTest;

// 합병 정렬 - 재귀적 방법
public class MergeSortTest{
    public static void mergeSort(int[] data){
        int[] divide = new int[data.length];
        mergeSort(data, divide, 0, data.length - 1);
    }

    public static void mergeSort(int[] data, int[] divide, int s, int e){
        if(s >= e) return;
        int mid = (s + e) / 2;
        mergeSort(data, divide, s, mid);
        mergeSort(data, divide, mid + 1, e);
        merge(data, divide, s, mid, e);
    }

    private static void merge(int[] data, int[] divide, int s, int m, int e) {
        for(int i = s; i <= e; i++) divide[i] = data[i];
        int p1 = s; int p2 = m + 1; // L partition and R partition index
        int idx = s; // merged index
        while(p1 <= m && p2 <= e) {
            if(divide[p1] <= divide[p2]) { data[idx] = divide[p1]; p1++; }
            else {data[idx] = divide[p2]; p2++;}
            idx++;
        } // remain part copy
        for(int i = 0; i <= m - p1; i++) {data[idx + i] = divide[p1 + i];}
    }

    public static void printArray(int[] data){
        for(int key : data){
            System.out.print(key + ", ");
        }
        System.out.println();
    }

    public static void main(String[] args){
        int[] data = {2, 8, 5, 3, 9, 4, 1};
        printArray(data);
        mergeSort(data);
        printArray(data);
    }
}

```

2. 정렬 알고리즘 간 비교

2.1. 정렬 알고리즘 성능 비교 코드

```java
package sortTest;

import java.io.*;

public class SortTest {
    public static void main(String[] args) {
        String filePath = "C:\\Users\\chado\\Desktop\\2022-1\\자료구조\\20220601\\input.txt";

        int numN = 100000;
        //int numN = 200000;
        //int numN = 500000;
        //int numN = 1000000;
        //int numN = 2000000;
        //int numN = 5000000;
        //int numN = 10000000;

        long beforeTime = System.currentTimeMillis(); // 코드 실행 전에 시간 받아오기

        try (BufferedReader reader = new BufferedReader(new FileReader(filePath))) {
            int numCount = 0;
            String line;
            int[] data = new int[numN];

            reader.readLine();
            while (numCount < numN) {
                line = reader.readLine();
                int N = Integer.parseInt(line);
                data[numCount] = N;
                numCount++;
            }

            BubbleSort2Test.bubbleSort(data);  // 버블 정렬
            //InsertionSort2Test.insertionSort(data);  // 삽입 정렬
            //SelectionSort2Test.selectionSort(data);  // 선택 정렬
            //QuickSort2Test.quickSort(data);  // 퀵 정렬
            //MergeSortTest.mergeSort(data);  // 병합 정렬

//          // 정렬 결과 출력
//          for(int i = 0; i < numN; i++) {
//              System.out.println(data[i]);
//          }
        }
        catch (FileNotFoundException e) {System.out.println("FileNotFoundException");}
        catch (IOException e) {System.out.println("IOException");}

        long afterTime = System.currentTimeMillis(); // 코드 실행 후에 시간 받아오기
        double diffTime = (afterTime - beforeTime)/1000.0; // 두 시간의 차 계산 (초단위로 변환)
        System.out.println("시간차이(sec) : "+ diffTime);
    }
}
```

2.2. 정렬 알고리즘 성능 비교 그래프 및 표



개수 \ 정렬	버블 정렬	삽입 정렬	선택 정렬	퀵 정렬	병합 정렬
100,000	11.568	1.214	1.993	0.068	0.042
200,000	46.87	8.015	7.754	0.09	0.064
500,000	304.31	29.17	48.715	0.151	0.114
1,000,000	1172.105	109.17	199.72	0.234	0.189
2,000,000				0.422	0.36
5,000,000				0.955	0.841
10,000,000				1.773	1.688

(단위: 초(sec))