

Shri Ramdeobaba College of Engineering & Management Nagpur-13

Department of Computer Application

Session: 2023-2024



Submission for

Course Name: Design Analysis and Algorithm Lab

Course Code: MCP546

Name of the Student: Jayesh Lalit Nandanwar

Class Roll No: 26

Semester: MCA II semester

Shift: 2

Batch: 2

Under the Guidance of

Prof. Manda Ukey

Date of submission: 07/04/2024

Practical 6

Aim: Perform QUICK sort on the data sets that you have created in practical_0.

Display the time taken to sort the elements from the files in ascending order. Consider random repeated and random unrepeated files.

Compare its time with the time taken for selection, insertion and merge sort.

For file 1 (Sequential unrepeated numbers):

```
Code: import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.util.Scanner;

public class QuickSortFile1 {
    public static void main(String[] args) throws IOException {
        String file_name="sequentialUnrepeatedNumbers";
        FileReader f = new FileReader("./"+file_name+".txt");

        // Reading numbers from file
        Scanner fileScanner = new Scanner(f);
        int[] array = new int[100001];
        int size = 0;

        while (fileScanner.hasNextInt()) {
            array[size++] = fileScanner.nextInt();
        }
        fileScanner.close();

        long start = System.currentTimeMillis();
        quickSort(array, 0, array.length - 1);
        long finish = System.currentTimeMillis();
        long timeElapsed = finish - start;

        writeToFile(array,file_name);

        System.out.println("\nTime taken for sorting: " + timeElapsed + "
milliseconds");
    }

    public static void quickSort(int[] array, int low, int high) {
        if (low < high) {
            int pivotIndex = partition(array, low, high);
```

```

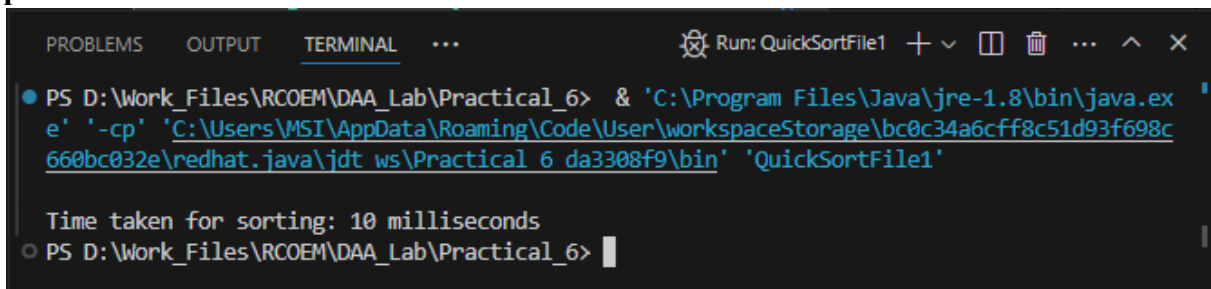
        quickSort(array, low, pivotIndex - 1);
        quickSort(array, pivotIndex + 1, high);
    }
}

public static int partition(int[] array, int low, int high) {
    int pivot = array[(low + high) / 2];
    while (low <= high) {
        while (array[low] < pivot) {
            low++;
        }
        while (array[high] > pivot) {
            high--;
        }
        if (low <= high) {
            int temp = array[low];
            array[low] = array[high];
            array[high] = temp;
            low++;
            high--;
        }
    }
    return low;
}

public static void writeToFile(int arr[], String file_name) throws IOException {
    FileWriter writer = new FileWriter("./"+file_name+"SortedOutput.txt");
    for (int i = 0; i < arr.length; i++) {
        writer.write(arr[i] + "\n");
    }
    writer.close();
}
}

```

Output:



```

PROBLEMS  OUTPUT  TERMINAL  ...
Run: QuickSortFile1 + - [ ] [ ] ... ^ X

● PS D:\Work_Files\RCOEM\DAA_Lab\Practical_6> & 'C:\Program Files\Java\jre-1.8\bin\java.exe' '-cp' 'C:\Users\MSI\AppData\Roaming\Code\User\workspaceStorage\bc0c34a6cff8c51d93f698c660bc032e\redhat.java\jdt_ws\Practical_6_da3308f9\bin' 'QuickSortFile1'

Time taken for sorting: 10 milliseconds
○ PS D:\Work_Files\RCOEM\DAA_Lab\Practical_6>

```

Time Taken: 10 ms

For file 2 (Random unrepeated numbers):

Code: import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.util.Scanner;

```
public class QuickSortFile2 {  
    public static void main(String[] args) throws IOException {  
        String file_name="randomUnrepeatedNumbers";  
        FileReader f = new FileReader("./"+file_name+".txt");  
  
        // Reading numbers from file  
        Scanner fileScanner = new Scanner(f);  
        int[] array = new int[100001];  
        int size = 0;  
  
        while (fileScanner.hasNextInt()) {  
            array[size++] = fileScanner.nextInt();  
        }  
        fileScanner.close();  
  
        long start = System.currentTimeMillis();  
        quickSort(array, 0, array.length - 1);  
        long finish = System.currentTimeMillis();  
        long timeElapsed = finish - start;  
  
        writeToFile(array,file_name);  
  
        System.out.println("\nTime taken for sorting: " + timeElapsed + "  
milliseconds");  
    }  
  
    public static void quickSort(int[] array, int low, int high) {  
        if (low < high) {  
            int partitionIndex = partition(array, low, high);  
  
            quickSort(array, low, partitionIndex - 1);  
            quickSort(array, partitionIndex + 1, high);  
        }  
    }  
  
    public static int partition(int[] array, int low, int high) {  
        int mid = low + (high - low) / 2;  
        int pivot = array[mid];  
        int i = low - 1;
```

```

        for (int j = low; j < high; j++) {
            if (array[j] < pivot) {
                i++;
                swap(array, i, j);
            }
        }

        swap(array, i + 1, mid);

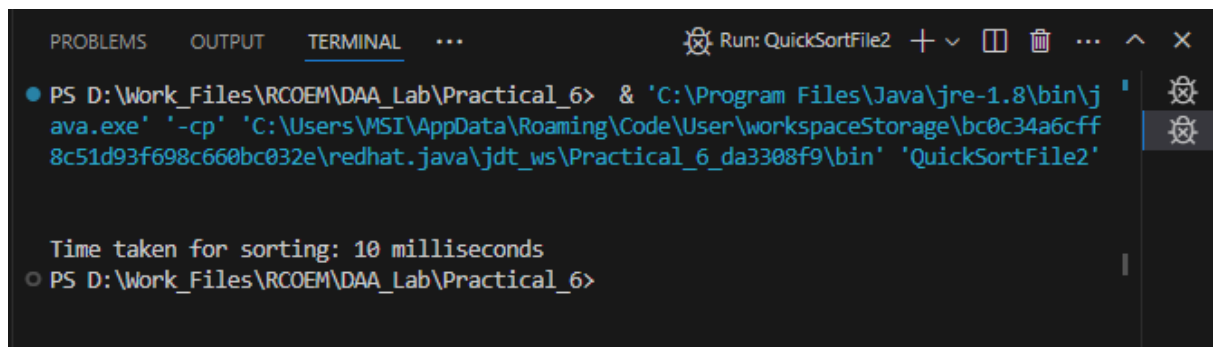
        return i + 1;
    }

    public static void swap(int[] array, int i, int j) {
        int temp = array[i];
        array[i] = array[j];
        array[j] = temp;
    }

    public static void writeToFile(int arr[], String file_name) throws IOException {
        FileWriter writer = new FileWriter("./"+file_name+"SortedOutput.txt");
        for (int i = 0; i < arr.length - 1; i++) {
            writer.write(arr[i] + "\n");
        }
        writer.close();
    }
}

```

Output:



```

PROBLEMS  OUTPUT  TERMINAL  ...
Run: QuickSortFile2

PS D:\Work_Files\RCEM\DAA_Lab\Practical_6> & 'C:\Program Files\Java\jre-1.8\bin\j
ava.exe' '-cp' 'C:\Users\MSI\AppData\Roaming\Code\User\workspaceStorage\bc0c34a6cff
8c51d93f698c660bc032e\redhat.java\jdt_ws\Practical_6_da3308f9\bin' 'QuickSortFile2'

Time taken for sorting: 10 milliseconds
PS D:\Work_Files\RCEM\DAA_Lab\Practical_6>

```

Time Taken: 10 ms

For file 3 (Random repeated numbers):

```
Code: import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.util.Scanner;

public class QuickSortFile3 {
    public static void main(String[] args) throws IOException {
        String file_name="randomRepeatedNumbers";
        FileReader f = new FileReader("./"+file_name+".txt");

        // Reading numbers from file
        Scanner fileScanner = new Scanner(f);
        int[] array = new int[100001];
        int size = 0;

        while (fileScanner.hasNextInt()) {
            array[size++] = fileScanner.nextInt();
        }
        fileScanner.close();

        long start = System.currentTimeMillis();
        quickSort(array, 0, array.length - 1);
        long finish = System.currentTimeMillis();
        long timeElapsed = finish - start;

        writeToFile(array,file_name);

        System.out.println("\nTime taken for sorting: " + timeElapsed + "
milliseconds");
    }

    public static void quickSort(int[] array, int low, int high) {
        if (low < high) {
            int pivotIndex = partition(array, low, high);
            quickSort(array, low, pivotIndex - 1);
            quickSort(array, pivotIndex + 1, high);
        }
    }

    public static int partition(int[] array, int low, int high) {
        int pivot = array[(low + high) / 2];

        while (low <= high) {
            while (array[low] < pivot) {
                low++;
            }
            while (array[high] > pivot) {
                high--;
            }
            if (low < high) {
                int temp = array[low];
                array[low] = array[high];
                array[high] = temp;
            }
        }
        return low;
    }
}
```

```

    }
    while (array[high] > pivot) {
        high--;
    }

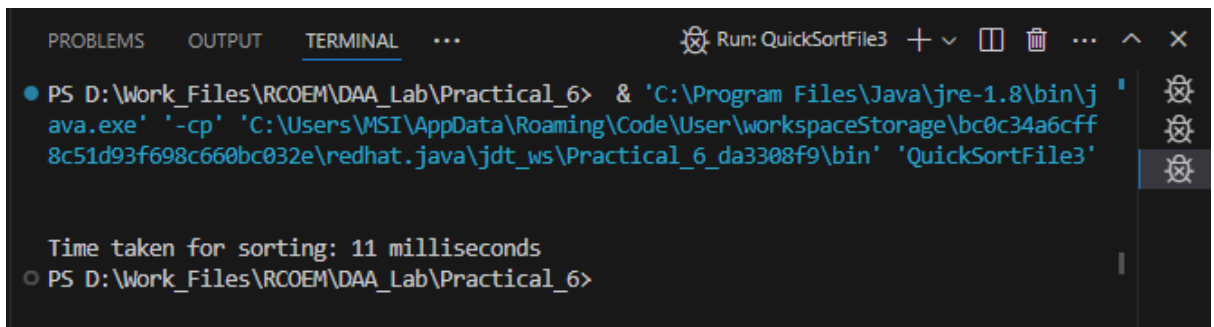
    if (low <= high) {
        int temp = array[low];
        array[low] = array[high];
        array[high] = temp;
        low++;
        high--;
    }
}

return low;
}

public static void writeToFile(int arr[],String file_name) throws IOException {
    FileWriter writer = new FileWriter("./"+file_name+"SortedOutput.txt");
    for (int i = 0; i < arr.length - 1; i++) {
        writer.write(arr[i] + "\n");
    }
    writer.close();
}
}

```

Output:



```

PROBLEMS  OUTPUT  TERMINAL  ...
Run: QuickSortFile3 + - [ ] [X] ... ^ X
● PS D:\Work_Files\RCEM\DAA_Lab\Practical_6> & 'C:\Program Files\Java\jre-1.8\bin\j
ava.exe' '-cp' 'C:\Users\MSI\AppData\Roaming\Code\User\workspaceStorage\bc0c34a6cff
8c51d93f698c660bc032e\redhat.java\jdt_ws\Practical_6_da3308f9\bin' 'QuickSortFile3'

Time taken for sorting: 11 milliseconds
○ PS D:\Work_Files\RCEM\DAA_Lab\Practical_6>

```

Time Taken: 11 ms

Comparison between time taken:

Sorting Algorithm	Time Taken (in ms) For Sorting		
	Sequential Unrepeated Numbers	Random Unrepeated Numbers	Random Repeated Numbers
Selection Sort	1103	1591	1690
Insertion Sort	3	1830	1900
Merge Sort	24	28	27
Quick Sort	10	10	11

Observations:

- Time taken to sort Random Unrepeated Numbers and Random Repeated Numbers is minimum in using Quick Sort Algorithm.
- Time taken to sort Sequential Unrepeated Numbers is less in Insertion Sort (3 ms) than Quick Sort(10 ms) and Merge Sort(24 ms) and Selection Sort (1103 ms).