Luke Harrison
CSCI 335 M01

Binary Insertion Sort Project

1. Pseudo Code

```
Algorithm: binInsertionSort(A[0..n-1])
//Sorts a given array by binary insertion sort
//Input: An array A[0...n-1] of n orderable items
//Output: Array A[0...n-1] sorted in nondecreasing order

for i ← 1 to n-1
        temp ← A[i]
        left ← 0
        right ← i-1
        newPos ← i
        while left ≤ right          binary search
                center ← ⌊(left+right)/2⌋
                If temp ≥ A[center]
                        left ← center+1
                else
                        right ← center-1
                        newpos ← center
        If newPos ≠ i               insertion
                For j ← i-1 to newPos
                        A[j+i] ← A[j]
                A[newPos] ← temp
```

2. Worst Case efficiency (comparison)

The worst case efficiency for comparison is O(nlogn). This is because there are log(n) comparisons for each execution of the binary search section of the algorithm and this section is executed n-2 times.

Analysis of binary search section:

Recurrence relation: $B(1) = 1$

$B(n) = B(n/2) + 1$

Soloution:

$[B((n/2)/2)+1]+1$

$B(n/2^2)+2$

$[B((n/2)/2^2)+1]+2$

$B(n/2^3)+3$

$B(n/2^i)+i$

Let $n = 2^k$	$B(2^{k-i})+i$
Let $i = k$	$B(2^0)+k$
	$1+k$
$k = \log(n)$	$\log(n)+1$
	$O(\log(n))$

Since the binary section is executed n-2 times we can conclude that the worst case efficiency for comparison is **O(nlog(n))**

3. Java Implementation

i) Source Code

```
public class Student {
   /*
       Purpose: the purpose of the student class is to store data
associated with each student in the text file
       Fields: the student fields are described as follows
           - (String) name: holds the name of the student
           - (int) id: holds the students id
           - (int) gpa: holds the students gpa
           - (int) age: holds the students age
    */
   private String name;
   private int id;
   private double gpa;
   private int age;

   //Student constructor that initializes all fields with values from
parameters
   //inputs: a string representing the name of the person, and integers
representing their id, gpa, and age
   public Student(String name, int id, double gpa, int age){
       this.name = name;
       this.id = id;
       this.gpa = gpa;
       this.age = age;
   }

   //Returns the students id
   public int getId(){
       return id;
   }

   //The student class's to string method which prints a formatted string
consisting of all its fields
   //Output: A formatted string of all the students fields
```

```
    public String toString(){
        return String.format("Name: %s ID: %d GPA: %.2f AGE: %d",
name,id,gpa,age);
    }
}
```

```
import java.io.File;
import java.io.FileNotFoundException;
import java.util.ArrayList;
import java.util.Scanner;
import java.util.Formatter;

public class StudentParser {
    /*
        Purpose: the purpose of the studentParser class is to parse,
store, and sort a collection of student objects obtained from a text file
        Fields: the studentParser fields are described as follows
            - (ArrayList<String>) students: an array of student objects
parsed from a text file
     */
    private ArrayList<Student> students;

    //StudentParser constructor that initializes the students ArrayList
and calls the parsFile method
    //Inputs: a string representing the file path of the file containing
students
    public StudentParser(String inputFile) throws FileNotFoundException{
        students = new ArrayList<>();
        parseFile(inputFile);
    }

    //Parses the provided text file of students into an ArrayList of
student objects
    //Inputs: a string representing the file path of the file containing
students
    public void parseFile(String inputFile) throws FileNotFoundException{
        students.clear();
        Scanner s = new Scanner(new File(inputFile));
        String line;
        String[] studInfo;
        String name;
        int id;
        int age;
        double gpa;
        Student stud;
        while(s.hasNext()){
            //loops until all lines are parsed
            line = s.nextLine();
```

```
            studInfo = line.split(" ");
            name = studInfo[0];
            id = Integer.parseInt(studInfo[1]);
            gpa = Double.parseDouble(studInfo[2]);
            age =  Integer.parseInt(studInfo[3]);
            stud = new Student(name,id,gpa,age);
            students.add(stud);
        }
    }

    //Writes the ArrayList of students objects to the file as a formatted
string of students
    //Inputs: a string representing the path of an output file or file
name
    public void writeToFile(String outputFile)throws
FileNotFoundException{
        Formatter f = new Formatter(outputFile);
        f.format("%s",toString());
        f.close();
    }

    //Sorts the ArrayList students field using binary insertion sort
    public void sort(){
        Student temp;
        int left, right, center, newPos;
        for(int x = 1; x<students.size(); x++){
            temp =  students.get(x);
            left = 0;
            right = x-1;
            newPos = x;
            //binary search to find new position
            while(left <= right){
                center = (int) Math.floor((left+right)/2);
                if(temp.getId() >=  students.get(center).getId()){
                    left = center+1;
                }
                else{
                    right = center-1;
                    newPos = center;
                }
            }
            //moving value at position x to new position
            if(newPos != x){
                students.remove(x);
                students.add(newPos,temp);
            }
        }
    }
```

```
   //Student parser's toString method that converts the ArrayList
students field into a formatted string
   //Output: a formatted string of the ArrayList student field
   public String toString(){
       String output = "";
       for(Student s: students){
           output += s.toString()+"\n";
       }
       return output;
   }
}
```

```
import java.io.FileNotFoundException;

public class Driver {
   /*
       Purpose: The driver classed contains the program's main method for
sorting students by their id
       1) A student parser object is created using the StudentsInput text
file
       2) The current student file is printed to terminal
       3) The sort (binary insertion sort) method is called on the
student parser object
       4) The sorted student file is printed to terminal
       5) The sorted student file is written to the Output text file

    */
   public static void main(String[] args){
       try {
           StudentParser sp = new StudentParser("StudentsInput.txt");
           System.out.println("Original File");
           System.out.println(sp);
           System.out.println("Sorted File");
           sp.sort();
           System.out.println(sp);
           sp.writeToFile("OutputFile.txt");
       }
       catch (FileNotFoundException e){
           System.out.println(e);
       }
   }
}
```

---

ii) Input text file (Name, id, GPA, Age all separated by space)

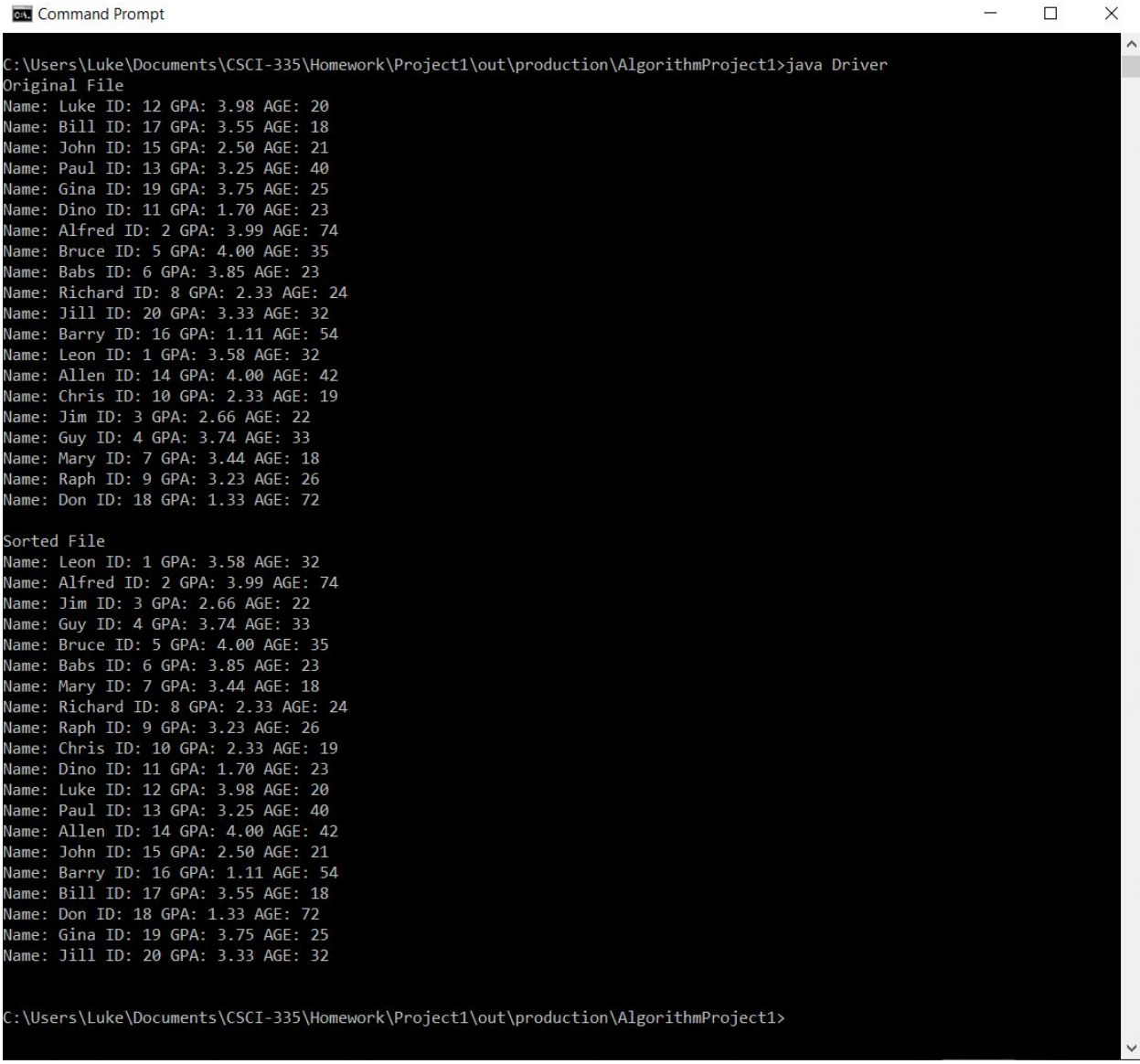
Luke 12 3.98 20

Bill 17 3.55 18
John 15 2.50 21
Paul 13 3.25 40
Gina 19 3.75 25
Dino 11 1.70 23
Alfred 2 3.99 74
Bruce 5 4.0 35
Babs 6 3.85 23
Richard 8 2.33 24
Jill 20 3.33 32
Barry 16 1.11 54
Leon 1 3.58 32
Allen 14 4.00 42
Chris 10 2.33 19
Jim 3 2.66 22
Guy 4 3.74 33
Mary 7 3.44 18
Raph 9 3.23 26
Don 18 1.33 72

iii) output text file

Name: Leon ID: 1 GPA: 3.58 AGE: 32
Name: Alfred ID: 2 GPA: 3.99 AGE: 74
Name: Jim ID: 3 GPA: 2.66 AGE: 22
Name: Guy ID: 4 GPA: 3.74 AGE: 33
Name: Bruce ID: 5 GPA: 4.00 AGE: 35
Name: Babs ID: 6 GPA: 3.85 AGE: 23
Name: Mary ID: 7 GPA: 3.44 AGE: 18
Name: Richard ID: 8 GPA: 2.33 AGE: 24
Name: Raph ID: 9 GPA: 3.23 AGE: 26
Name: Chris ID: 10 GPA: 2.33 AGE: 19
Name: Dino ID: 11 GPA: 1.70 AGE: 23
Name: Luke ID: 12 GPA: 3.98 AGE: 20
Name: Paul ID: 13 GPA: 3.25 AGE: 40
Name: Allen ID: 14 GPA: 4.00 AGE: 42
Name: John ID: 15 GPA: 2.50 AGE: 21
Name: Barry ID: 16 GPA: 1.11 AGE: 54
Name: Bill ID: 17 GPA: 3.55 AGE: 18
Name: Don ID: 18 GPA: 1.33 AGE: 72
Name: Gina ID: 19 GPA: 3.75 AGE: 25
Name: Jill ID: 20 GPA: 3.33 AGE: 32

iv)


Command Prompt
C:\Users\Luke\Documents\CSCI-335\Homework\Project1\out\production\AlgorithmProject1>java Driver
Original File
Name: Luke ID: 12 GPA: 3.98 AGE: 20
Name: Bill ID: 17 GPA: 3.55 AGE: 18
Name: John ID: 15 GPA: 2.50 AGE: 21
Name: Paul ID: 13 GPA: 3.25 AGE: 40
Name: Gina ID: 19 GPA: 3.75 AGE: 25
Name: Dino ID: 11 GPA: 1.70 AGE: 23
Name: Alfred ID: 2 GPA: 3.99 AGE: 74
Name: Bruce ID: 5 GPA: 4.00 AGE: 35
Name: Babs ID: 6 GPA: 3.85 AGE: 23
Name: Richard ID: 8 GPA: 2.33 AGE: 24
Name: Jill ID: 20 GPA: 3.33 AGE: 32
Name: Barry ID: 16 GPA: 1.11 AGE: 54
Name: Leon ID: 1 GPA: 3.58 AGE: 32
Name: Allen ID: 14 GPA: 4.00 AGE: 42
Name: Chris ID: 10 GPA: 2.33 AGE: 19
Name: Jim ID: 3 GPA: 2.66 AGE: 22
Name: Guy ID: 4 GPA: 3.74 AGE: 33
Name: Mary ID: 7 GPA: 3.44 AGE: 18
Name: Raph ID: 9 GPA: 3.23 AGE: 26
Name: Don ID: 18 GPA: 1.33 AGE: 72
Sorted File
Name: Leon ID: 1 GPA: 3.58 AGE: 32
Name: Alfred ID: 2 GPA: 3.99 AGE: 74
Name: Jim ID: 3 GPA: 2.66 AGE: 22
Name: Guy ID: 4 GPA: 3.74 AGE: 33
Name: Bruce ID: 5 GPA: 4.00 AGE: 35
Name: Babs ID: 6 GPA: 3.85 AGE: 23
Name: Mary ID: 7 GPA: 3.44 AGE: 18
Name: Richard ID: 8 GPA: 2.33 AGE: 24
Name: Raph ID: 9 GPA: 3.23 AGE: 26
Name: Chris ID: 10 GPA: 2.33 AGE: 19
Name: Dino ID: 11 GPA: 1.70 AGE: 23
Name: Luke ID: 12 GPA: 3.98 AGE: 20
Name: Paul ID: 13 GPA: 3.25 AGE: 40
Name: Allen ID: 14 GPA: 4.00 AGE: 42
Name: John ID: 15 GPA: 2.50 AGE: 21
Name: Barry ID: 16 GPA: 1.11 AGE: 54
Name: Bill ID: 17 GPA: 3.55 AGE: 18
Name: Don ID: 18 GPA: 1.33 AGE: 72
Name: Gina ID: 19 GPA: 3.75 AGE: 25
Name: Jill ID: 20 GPA: 3.33 AGE: 32
C:\Users\Luke\Documents\CSCI-335\Homework\Project1\out\production\AlgorithmProject1>