Problem 2

Isabel Martínez Gómez
Algorithm and Complexity
2019-2020

Implementation

To solve the problem by a Divide and Conquer algorithm, I have been dividing the vector always by two until it is not possible to continue dividing it, that is, when the vector's size is one.

To calculate the average grade of a student, I have divided each element by the total number of grades inside the vector when the base case is reached. Finally, all the elements are added getting the average.

For example, if I have a vector of 3 elements [1,2,3], the total number of grades in the vector is 3 and the average will be: 1/3 + 2/3 + 3/3 = 2.

I will explain it step by step in the following slides.

average: base case

- If i and j have the same value, it means that we have reached the base case. This is, when the arraylist grades is simply enough and we can't continue dividing by two the arraylist. That is, when the arraylist grades has size 1.

- So, in this case, we return the element divided by the initial size of the arraylist grades.

```
public float average(ArrayList grades,int i, int j, int size) throws IOException
{
    //if grades is simply enough we return de grade divided by size
    if (i==j)
    {
        file.write(grades.get(i)+"/"+size+" = "+ (float) grades.get(i)/size+"\n");
        return (float) grades.get(i)/size;
    }
    else
    {
        file.write(printArrayList(grades,i,j));
        //Gets the position in the half of grades by a floor division
        int half = (i+j)/2;
        //Descompose the grades array into two halves
        float firstHalf = average(grades,i,half,size);
        float secondHalf = average(grades,half+1,j,size);
        file.write("Average value --> "+firstHalf+" + "+secondHalf+" = "+(firstHalf+secondHalf)+"\n");
        return firstHalf + secondHalf;
    }
}
```

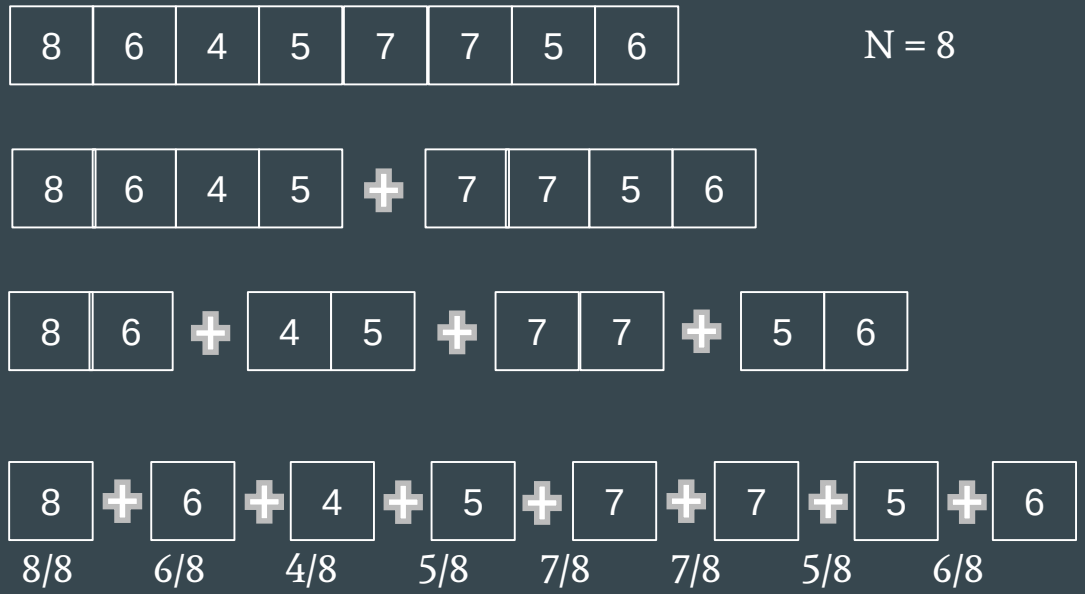
average: body function

- If the base case hasn't been reached, the value half of the arraylist is calculated and the function is called with the first half of the arraylist added with the second half of the arraylist.

- firstHalf will calculate the average of the fist half of the arraylist and secondhalf the average of the second half of the arraylist grades.

```
public float average(ArrayList grades,int i, int j, int size) throws IOException
{
    //if grades is simply enough we return de grade divided by size
    if (i==j)
    {
        file.write(grades.get(i)+"/"+size+" = "+ (float) grades.get(i)/size+"\n");
        return (float) grades.get(i)/size;
    }
    else
    {
        file.write(printArrayList(grades,i,j));
        //Gets the position in the half of grades by a floor division
        int half = (i+j)/2;
        //Descompose the grades array into two halves
        float firstHalf = average(grades,i,half,size);
        float secondHalf = average(grades,half+1,j,size);
        file.write("Average value --> "+firstHalf+" + "+secondHalf+" = "+(firstHalf+secondHalf)+"\n");
        return firstHalf + secondHalf;
    }
}
```

Example: with N being a power of 2



Average = 8/8 + 6/8 + 4/8 + 5/8 + 7/8 + 7/8 + 5/8 + 6/8 = 6

Example: with N not a power of 2

N = 7

Average = 8/7 + 6/7 + 4/7 + 5/7 + 7/7 + 7/7 + 5/7 = 6

Main function

- This is the main function. First we initialize the arraylist grades from data in the input file “example_d&c.txt” and then we calculate the average of the degrees in the txt with the function average.

```
public class Problem2 {

    public static void main(String[] args) throws IOException {
        // ******************** PROBLEM 2 **********************
        AverageGrades p2 = new AverageGrades();
        ArrayList grades = new ArrayList<>();
        p2.initializeFromTxt("example_d&c.txt",grades);
        float average = p2.average(grades, 0, grades.size()-1, grades.size());
        p2.getFile().write("\nAverage value obtained: "+average);
        p2.getFile().close();
    }
```

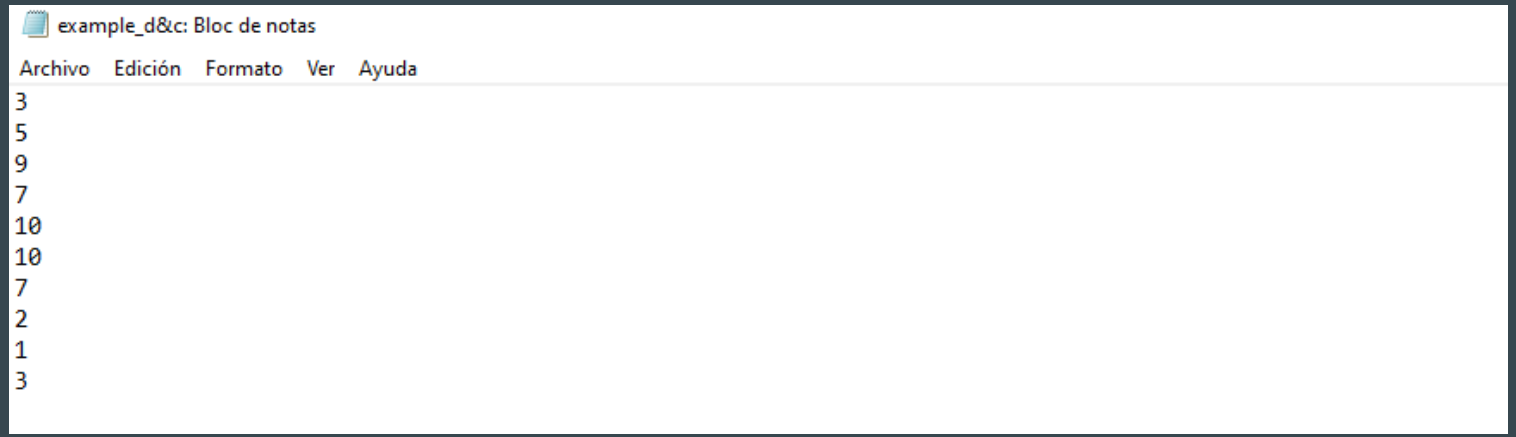
Input file: Initialize ArrayList grades

- This function takes the grades from the input file and initializes the ArrayList grades.
- For this, the function uses FileReader and BufferedReader.

```
public void initializeFromTxt(String file, ArrayList grades) throws FileNotFoundException, IOException {
    FileReader f = new FileReader(file);
    BufferedReader b = new BufferedReader(f);
    String str;
    while((str = b.readLine())!=null)
    {
        float grade = (float) Integer.valueOf(str);
        grades.add(grade);
    }
    b.close();
}
```

Input file

- This is an example of the format of my input file “example_d&c.txt”



Output file: printArrayList

```
public String printArrayList(ArrayList grades, int i, int j)
{
    String str = "[";
    for (int k=i;k<j;k++)
    {
        if (k==(j-1))
        {
            str+=grades.get(k);
        }else
        {
            str+=grades.get(k)+", ";
        }
    }
    str+="]\n";
    return str;
}
```

I have created a printArrayList function to write the arraylist in the output file so that you can see how they are dividing by two in each iteration.
I use this function in average()

Output file: file.write in average()

To show the recursive stages of the divide and conquer process, in the average function I use file.write to write in the output file those stages that are interesting such as the result of the base case and how the average is being calculated.

```
public float average(ArrayList grades,int i, int j, int size) throws IOException
{
    //if grades is simply enough we return de grade divided by size
    if (i==j)
    {
        file.write("\n"+printArrayList(grades,i,j+1));
        file.write(grades.get(i)+"/"+size+" = "+ (float) grades.get(i)/size+"\n");
        return (float) grades.get(i)/size;
    }
    else
    {
        file.write(printArrayList(grades,i,j+1));
        //Gets the position in the half of grades by a floor division
        int half = (i+j)/2;
        //Descompose the grades array into two halves
        float firstHalf = average(grades,i,half,size);
        float secondHalf = average(grades,half+1,j,size);
        file.write("\nAverage value --> "+firstHalf+" + "+secondHalf+" = "+(firstHalf+secondHalf)+"\n");
        return firstHalf + secondHalf;
    }
}
```

Output file

- This is an example of “output_d&c.txt”:

output_d&c: Bloc de notas

Archivo Edición Formato Ver Ayuda

```
The total number of subjects of the student is 10

[3.0, 5.0, 9.0, 7.0, 10.0, 10.0, 7.0, 2.0, 1.0, 3.0]
[3.0, 5.0, 9.0, 7.0, 10.0]
[3.0, 5.0, 9.0]
[3.0, 5.0]

[3.0]
3.0/10 = 0.3

[5.0]
5.0/10 = 0.5

Average value --> 0.3 + 0.5 = 0.8

[9.0]
9.0/10 = 0.9

Average value --> 0.8 + 0.9 = 1.7
[7.0, 10.0]

[7.0]
7.0/10 = 0.7

[10.0]
10.0/10 = 1.0

Average value --> 0.7 + 1.0 = 1.7
```

```
Average value --> 1.7 + 1.7 = 3.4
[10.0, 7.0, 2.0, 1.0, 3.0]
[10.0, 7.0, 2.0]
[10.0, 7.0]

[10.0]
10.0/10 = 1.0

[7.0]
7.0/10 = 0.7

Average value --> 1.0 + 0.7 = 1.7

[2.0]
2.0/10 = 0.2

Average value --> 1.7 + 0.2 = 1.9000001
[1.0, 3.0]

[1.0]
1.0/10 = 0.1

[3.0]
3.0/10 = 0.3

Average value --> 0.1 + 0.3 = 0.4

Average value --> 1.9000001 + 0.4 = 2.3000002

Average value --> 3.4 + 2.3000002 = 5.7000003

Average value obtained: 5.7000003
```

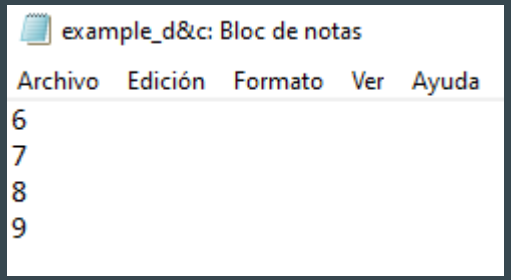
Efficiency

- The divide and conquer algorithm has a complexity of O(n):
  - n: size of our original case
  - Number of subcases, k = 2
  - Exists a constant b such that the size of the k subcases is approximately n/b, b = 2
  - p = 0

- $k > b^p \rightarrow T(n) = O(n^{\log_b k}) \rightarrow O(n^{\log_2 2}) = O(n)$
- In this case, we have obtained a complexity of O(n), so the divide and conquer algotithm does not improve at all the simple iterative version.

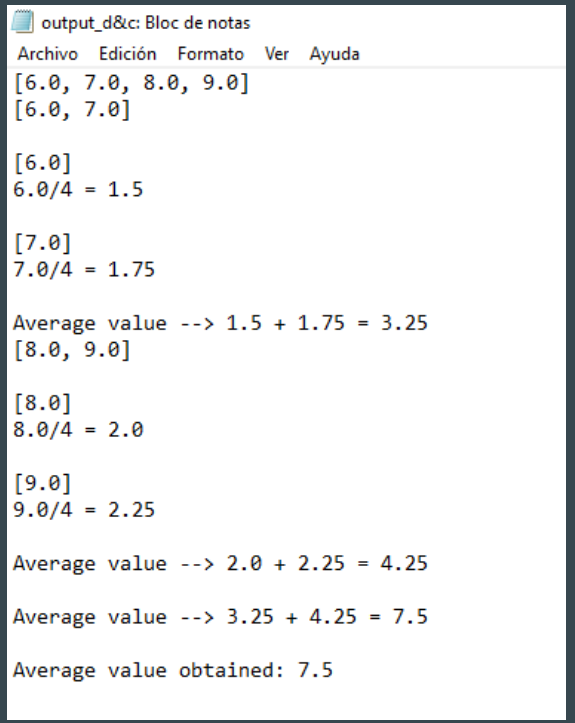
Examples: First example (n being a power of 2)

INPUT FILE



N=4
30/4 = 7,5

OUTPUT FILE



Examples: Second example (n being a power of 2)

INPUT FILE

example_d&c: Bloc de notas

Archivo Edición Formato Ver Ayuda

```
7
8
5
4
8
7
10
6
```

N=8

55/8 = 6,875

OUTPUT FILE

output_d&c: Bloc de notas

Archivo Edición Formato Ver Ayuda

```
[7.0, 8.0, 5.0, 4.0, 8.0, 7.0, 10.0, 6.0]
[7.0, 8.0, 5.0, 4.0]
[7.0, 8.0]

[7.0]
7.0/8 = 0.875

[8.0]
8.0/8 = 1.0

Average value --> 0.875 + 1.0 = 1.875
[5.0, 4.0]

[5.0]
5.0/8 = 0.625

[4.0]
4.0/8 = 0.5

Average value --> 0.625 + 0.5 = 1.125

Average value --> 1.875 + 1.125 = 3.0
[8.0, 7.0, 10.0, 6.0]
[8.0, 7.0]
```

```
[8.0]
8.0/8 = 1.0

[7.0]
7.0/8 = 0.875

Average value --> 1.0 + 0.875 = 1.875
[10.0, 6.0]

[10.0]
10.0/8 = 1.25

[6.0]
6.0/8 = 0.75

Average value --> 1.25 + 0.75 = 2.0

Average value --> 1.875 + 2.0 = 3.875

Average value --> 3.0 + 3.875 = 6.875

Average value obtained: 6.875
```

Examples: Third example (n not a power of 2)

INPUT FILE

example_d&c: Bloc de notas

Archivo Edición Formato Ver Ayuda

```
7
8
1
0
5
```

N=5
21/5 = 4,2

OUTPUT FILE

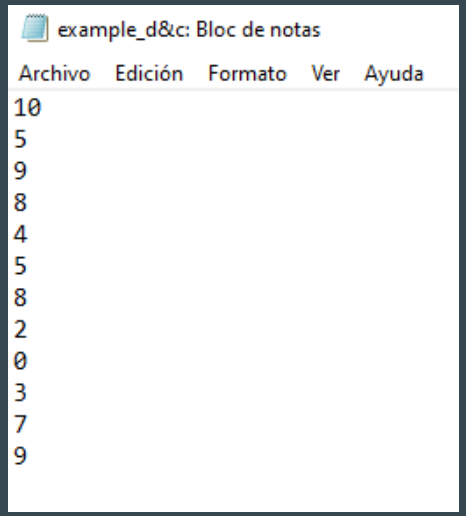
output_d&c: Bloc de notas

Archivo Edición Formato Ver Ayuda

```
[7.0, 8.0, 1.0, 0.0, 5.0]
[7.0, 8.0, 1.0]
[7.0, 8.0]

[7.0]
7.0/5 = 1.4

[8.0]
8.0/5 = 1.6

Average value --> 1.4 + 1.6 = 3.0

[1.0]
1.0/5 = 0.2

Average value --> 3.0 + 0.2 = 3.2
[0.0, 5.0]

[0.0]
0.0/5 = 0.0

[5.0]
5.0/5 = 1.0

Average value --> 0.0 + 1.0 = 1.0

Average value --> 3.2 + 1.0 = 4.2

Average value obtained: 4.2
```

Examples: Fourth example (n not a power of 2)

INPUT FILE



N=12
70/12 = 5,8333...

OUTPUT FILE