


Algorithmics	Student information	Date	Number of session
	UO:275688	02/03/21	4
	Surname: Fernández Esparta	 Escuela de Ingeniería Informática Universidad de Oviedo	
	Name: Mikel		



## Activity 1. [Execution times]

We have obtained this table with the measurements of time having different size problems, using the greedy algorithms we previously implemented in the class SegmentsPlacement, in which we had to calculate the midpoints (average) between different points for a given list which we have to read from files.

These are the implementation of the three greedy algorithms:

```
public static List<Double> greedy1(List<Integer> list) {
    List<Double> midPoints = new ArrayList<Double>();
    int i = 0;
    int first = 0;
    int second = list.get(0);
    while(i < list.size()) {
        double value = average(first, second);
        midPoints.add(value);
        first = second;
        i++;
        if(i < list.size())
            second += list.get(i);
        else
            break;
    }
    return midPoints;
}

private static void showMidPoints(List<Double> midPoints, List<Integer> list) {
    int i = 0;
    int first = 0;
    int second = list.get(0);
    for(int k = 0; k < midPoints.size(); k++) {
        System.out.println("S" + k + ": (" + first + " to " + second +
            "),midpoint = " + midPoints.get(k));
        first = second;
        i++;
        if(i < list.size())
            second += list.get(i);
        else
            break;
    }
}
```

Algorithmics	Student information	Date	Number of session
	UO:275688	02/03/21	4
	Surname: Fernández Esparta		
	Name: Mikel		

```
private static double countPufosos(List<Double> midPoints) {
    double pufosos = 0;
    for(Double a : midPoints){
        pufosos += a;
    }
    return pufosos;
}
```

```
private static double average(int x, int y) {
    double average = (x + y) / 2;
    return average;
}
```

```
public static List<Double> greedy2(List<Integer> list) {
    Collections.sort(list);
    Collections.reverse(list);
    List<Double> midPoints = new ArrayList<Double>();
    int i = 0;
    int first = 0;
    int second = list.get(0);|
    while(i < list.size()) {
        double value = average(first, second);
        midPoints.add(value);
        first = second;
        i++;
        if(i < list.size())
            second += list.get(i);
        else
            break;
    }

    return midPoints;
}
```

```
public static List<Double> greedy3(List<Integer> list) {
    Collections.sort(list);

    List<Double> midPoints = new ArrayList<Double>();

    int i = 0;
    int first = 0;
    int second = list.get(0);

    while(i < list.size() && second != 0) {
        double value = average(first, second);
        midPoints.add(value);
        first = second;
        i++;
        if(i < list.size())
            second += list.get(i);
        else
            break;
    }

    return midPoints;
}
```

Algorithmics	Student information	Date	Number of session
	UO:275688	02/03/21	4
	Surname: Fernández Esparta		
	Name: Mikel		

```

public static void main(String args[]) {
    int numberFiles = 2;

    for (int i = 1; i <= numberFiles; i++) {
        String fileName = Paths.get("").toAbsolutePath().toString() + "/src/algstudent/s4/game" + i + ".txt";
        System.out.println("FILE: " + fileName);

        List = readRankingFromFile(fileName);

        List.remove(0); //remove the value of size

        List<Double> midPoints1 = greedy1(List);
        showMidPoints(midPoints1, List);
        System.out.println("Cost of greedy 1 = " + countPufosos(midPoints1) + " pufosos");

        System.out.println("\n-----\n");

        List<Double> midPoints2 = greedy2(List);
        Collections.sort(List);
        Collections.reverse(List);
        showMidPoints(midPoints2, List);
        System.out.println("Cost of greedy 2 = " + countPufosos(midPoints2) + " pufosos");

        System.out.println("\n-----\n");

        List<Double> midPoints3 = greedy3(List);
        Collections.reverse(List);
        showMidPoints(midPoints3, List);
        System.out.println("Cost of greedy 3 = " + countPufosos(midPoints3) + " pufosos");

        System.out.println("\n*****\n");
    }
}

```

And this is the implementation of the SegmentsPlacementTimes:

```

private static ArrayList<Integer> createList(int n){
    ArrayList<Integer> list = new ArrayList<Integer>();
    Random r = new Random();

    for(int i = 0; i < n ; i++) {
        list.add(r.nextInt(100));
    }

    return list;
}

public static void main(String args[]) {
    int num = 204800 * 2;
    for (int i = 100; i <= num; i *= 2) {
        System.out.println("N: " + i);
        ArrayList<Integer> list = createList(i);
        long t1 = System.currentTimeMillis();
        SegmentsPlacement.greedy1(list);
        long t2 = System.currentTimeMillis();
        System.out.println("The time for the algorithm greedy1 is: " + (t2-t1) + " milliseconds");

        long t3 = System.currentTimeMillis();
        SegmentsPlacement.greedy2(list);
        long t4 = System.currentTimeMillis();
        System.out.println("The time for the algorithm greedy2 is: " + (t4-t3) + " milliseconds");

        long t5 = System.currentTimeMillis();
        SegmentsPlacement.greedy3(list);
        long t6 = System.currentTimeMillis();
        System.out.println("The time for the algorithm greedy3 is: " + (t6-t5) + " milliseconds");

        System.out.println("\n*****\n");
    }
}

```

Algorithmics	Student information	Date	Number of session
	UO:275688	02/03/21	4
	Surname: Fernández Esparta		
	Name: Mikel		

Additionally, we obtained these results from the given files, showing the different points and the number of pufosos.

```
FILE: C:\Users\mikel\git\repository\Algoritmia\src\algstudent\s4\game1.txt
S0: (0 to 20),midpoint = 10.0
S1: (20 to 80),midpoint = 50.0
S2: (80 to 84),midpoint = 82.0
S3: (84 to 92),midpoint = 88.0
S4: (92 to 96),midpoint = 94.0
S5: (96 to 113),midpoint = 104.0
Cost of greedy 1 = 428.0 pufosos
```

```
-----
S0: (0 to 60),midpoint = 30.0
S1: (60 to 80),midpoint = 70.0
S2: (80 to 97),midpoint = 88.0
S3: (97 to 105),midpoint = 101.0
S4: (105 to 109),midpoint = 107.0
S5: (109 to 113),midpoint = 111.0
Cost of greedy 2 = 507.0 pufosos
```

```
-----
S0: (0 to 60),midpoint = 2.0
S1: (60 to 80),midpoint = 6.0
S2: (80 to 97),midpoint = 12.0
S3: (97 to 105),midpoint = 24.0
S4: (105 to 109),midpoint = 43.0
S5: (109 to 113),midpoint = 83.0
Cost of greedy 3 = 170.0 pufosos
```

```
FILE: C:\Users\mikel\git\repository\Algoritmia\src\algstudent\s4\game2.txt
S0: (0 to 7),midpoint = 3.0
S1: (7 to 106),midpoint = 56.0
S2: (106 to 120),midpoint = 113.0
S3: (120 to 150),midpoint = 135.0
S4: (150 to 174),midpoint = 162.0
S5: (174 to 234),midpoint = 204.0
S6: (234 to 248),midpoint = 241.0
S7: (248 to 249),midpoint = 248.0
S8: (249 to 325),midpoint = 287.0
S9: (325 to 347),midpoint = 336.0
S10: (347 to 425),midpoint = 386.0
S11: (425 to 433),midpoint = 429.0
S12: (433 to 483),midpoint = 458.0
S13: (483 to 517),midpoint = 500.0
S14: (517 to 604),midpoint = 560.0
S15: (604 to 649),midpoint = 626.0
S16: (649 to 672),midpoint = 660.0
S17: (672 to 767),midpoint = 719.0
S18: (767 to 773),midpoint = 770.0
S19: (773 to 813),midpoint = 793.0
Cost of greedy 1 = 7686.0 pufosos
```

```
S0: (0 to 99),midpoint = 49.0
S1: (99 to 194),midpoint = 146.0
S2: (194 to 281),midpoint = 237.0
S3: (281 to 359),midpoint = 320.0
S4: (359 to 435),midpoint = 397.0
S5: (435 to 495),midpoint = 465.0
S6: (495 to 545),midpoint = 520.0
S7: (545 to 590),midpoint = 567.0
S8: (590 to 630),midpoint = 610.0
S9: (630 to 664),midpoint = 647.0
S10: (664 to 694),midpoint = 679.0
S11: (694 to 718),midpoint = 706.0
S12: (718 to 741),midpoint = 729.0
S13: (741 to 763),midpoint = 752.0
S14: (763 to 777),midpoint = 770.0
S15: (777 to 791),midpoint = 784.0
S16: (791 to 799),midpoint = 795.0
S17: (799 to 806),midpoint = 802.0
S18: (806 to 812),midpoint = 809.0
S19: (812 to 813),midpoint = 812.0
Cost of greedy 2 = 11596.0 pufosos
```

Algorithmics	Student information	Date	Number of session
	UO:275688	02/03/21	4
	Surname: Fernández Esparta		
	Name: Mikel		

S0: (0 to 99),midpoint = 0.0  
 S1: (99 to 194),midpoint = 4.0  
 S2: (194 to 281),midpoint = 10.0  
 S3: (281 to 359),midpoint = 18.0  
 S4: (359 to 435),midpoint = 29.0  
 S5: (435 to 495),midpoint = 43.0  
 S6: (495 to 545),midpoint = 61.0  
 S7: (545 to 590),midpoint = 83.0  
 S8: (590 to 630),midpoint = 107.0  
 S9: (630 to 664),midpoint = 134.0  
 S10: (664 to 694),midpoint = 166.0  
 S11: (694 to 718),midpoint = 203.0  
 S12: (718 to 741),midpoint = 245.0  
 S13: (741 to 763),midpoint = 293.0  
 S14: (763 to 777),midpoint = 348.0  
 S15: (777 to 791),midpoint = 416.0  
 S16: (791 to 799),midpoint = 493.0  
 S17: (799 to 806),midpoint = 575.0  
 S18: (806 to 812),midpoint = 666.0  
 S19: (812 to 813),midpoint = 763.0  
 Cost of greedy 3 = 4657.0 pufosos

Finally, the different values obtained for the times of each method are:

n	tGreedy1	tGreedy2	tGreedy3
100	0	1	0
200	0	1	1
400	1	2	1
800	2	3	2
1600	2	4	3
3200	3	8	4
6400	4	14	4
12800	5	19	8
25600	7	26	16
51200	8	40	30
102400	10	50	67
204800	12	57	81

Algorithmics	Student information	Date	Number of session
	UO:275688	02/03/21	4
	Surname: Fernández Esparta		
	Name: Mikel		

## Activity 2. [Questions]

### Explain if any of the algorithms involves the optimal solution. (maximizing)

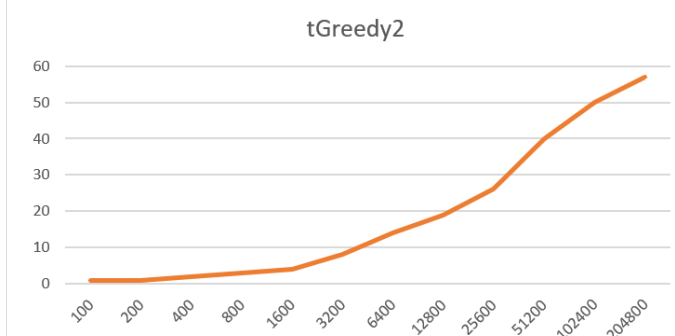
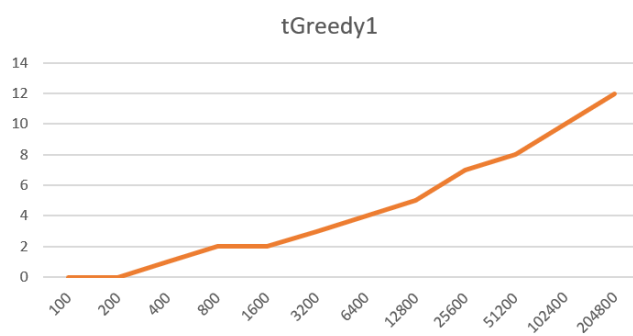
Greedy2 corresponds to the optimal solution from the point of view of wanting to maximize the number of pufosos. Since we have to pick from the biggest value to the smallest ones, meaning that the value of the midpoints is going to be higher than in the previous greedy algorithms, because the values start from the biggest number.

### Explain if any of the algorithms involves the optimal solution. (minimizing)

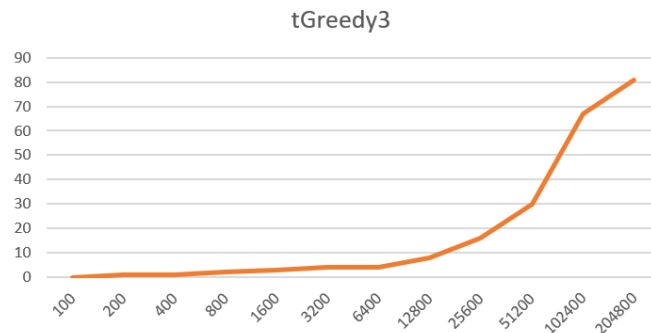
On the other hand, greedy3 is responsible for minimizing the number of pufosos, the reason why, is because it is the opposite to greedy2, we start by picking the smallest values possible on the list. By doing this, the midpoint or average value between two points is going to be lower than the ones we may obtained with either greedy1 or greedy2.

### Explain the theoretical time complexities of the three greedy algorithms

The theoretical complexity for greedy1 is  $O(n)$  as we can see on the previous table showing the times it takes for the program to execute each algorithm given different problem sizes. In the case of greedy2 and greedy3, their corresponding time complexity is  $O(n \log n)$ . To prove this, we show some graphs to check if they are correct.



Algorithmics	Student information	Date	Number of session
	UO:275688	02/03/21	4
	Surname: Fernández Esparta		
	Name: Mikel		



**Explain if the times obtained in the table are in tune or not, with the previous complexities.**

Yes, they make sense since we don't have to sort the list of values on greedy1, instead on greedy2 and greedy3, the values must be sorted in descending and ascending order correspondingly, that is the reason why their complexity is  $O(n \log n)$  as we have shown in the previous laboratory session. This of course having in mind the best case sorting algorithm.