TP0 - Análisis de Algoritmos

Margni, Lucas A. ~ *Villarroel, Valentina* ~ *Fernandez, Nicanor*
Mendiberri, Ignacio ~ *Fabris, Franco*

Grupo 3

1 Entrada/Salida con Java

1.1 Ejercicio 1

```java
public static void main(String[] args) {
        String linea = null;

        try {
            FileReader fr = new FileReader(archivoEntrada);
            FileWriter fw = new FileWriter(archivoSalida);

            BufferedReader buffLectura = new BufferedReader(fr);
            BufferedWriter buffEscritura = new BufferedWriter(fw);

            while ((linea = buffLectura.readLine()) != null) {
                linea = linea.replace(" ", "");
                System.out.print(linea);
                buffEscritura.write(linea);
            }

            buffLectura.close();
            buffEscritura.close();
        } catch (FileNotFoundException ex) {
            System.err.println("No existe el archivo que se quiere leer");
        } catch (IOException ex) {
            System.err.println("Error leyendo o escribiendo en algun archivo");
        }
    }
```

1.2 Ejercicio 2

```java
public static void main(String[] args) {
        String linea = null;
        int cont = 0;

        try {
            FileReader fr = new FileReader(archivoEntrada);
            FileWriter fw = new FileWriter(archivoSalida);

            BufferedReader buffLectura = new BufferedReader(fr);
            BufferedWriter buffEscritura = new BufferedWriter(fw);

            while ((linea = buffLectura.readLine()) != null) {
                if((cont % 2) == 0) {
```

```
                    System.out.println(linea);
                    buffEscritura.write(linea + "\n");
                }
                cont++;
            }

            buffLectura.close();
            buffEscritura.close();
        } catch (FileNotFoundException ex) {
            System.err.println("No existe el archivo que se quiere leer");
        } catch (IOException ex) {
            System.err.println("Error leyendo o escribiendo en algun archivo");
        }
    }
```

1.3 Ejercicio 3

```
public static void main(String[] args) {
        Random generador = new Random();
        double num;

        try {
            BufferedWriter buff = new BufferedWriter(new FileWriter(archivoSalida)
                );

            for (int i = 0; i < 100 ; i++){
                        num = generador.nextDouble(200) - 100;
                        buff.write(num + "\n");
                        System.out.println(num);
                }

            buff.close();
        } catch (FileNotFoundException ex) {
            System.err.println("No existe el archivo que se quiere leer");
        } catch (IOException ex) {
            System.err.println("Error leyendo o escribiendo en algun archivo");
        }
    }
```

1.4 Ejercicio 4

```
public static void main(String[] args) {
        Random generador = new Random();
        int cantLetras = 'z' - 'a' + 1, num;
        char carac;

        try {
            BufferedWriter buff = new BufferedWriter(new FileWriter(archivoSalida)
                );

            for (int i = 0; i < 10 ; i++){
                        num = generador.nextInt(cantLetras*2+10);
                if(num < cantLetras) {
                    carac = (char)((int)'a' + num);
                } else if(num < cantLetras*2) {
                    carac = (char)((int)'A' + num - cantLetras);
                } else {
```

```
                    carac = (char)((int)'0' + num - cantLetras*2);
                }

                    buff.write(Character.toString(carac) + " - ");
                    System.out.print(String.valueOf(carac) + " - ");
                //Character.toString(carac) y String.valueOf(carac)
                //transforman un caracter a string, pongo las dos opciones porque quiero
                }

            buff.close();
        } catch (FileNotFoundException ex) {
            System.err.println("No existe el archivo que se quiere leer");
        } catch (IOException ex) {
            System.err.println("Error leyendo o escribiendo en algun archivo");
        }
    }
```

1.5 Ejercicio 5

```
static final String archivoEntrada = "entrada.txt";
    static final String archivoSalida = "salida.txt";
    //Generamos 100 numeros aleatorios del 1 al 1000 sin que se repitan
    public static void main(String[] args) {
        Random generador = new Random();
        HashMap numeros = new HashMap();
        int num, cant = 0;

        try {
            BufferedWriter buff = new BufferedWriter(new FileWriter(archivoSalida));

            while(cant < 100) {
                        num = generador.nextInt(1000);

                if(numeros.put(num, num) == null) {
                    //si nos devuelve null, es que el numero no estaba en el hashmap
                    buff.write(num + "\n");
                            System.out.println(num);
                    cant++;
                }
                }

            buff.close();
        } catch (FileNotFoundException ex) {
            System.err.println("No existe el archivo que se quiere leer");
        } catch (IOException ex) {
            System.err.println("Error leyendo o escribiendo en algun archivo");
        }
    }
```

2 Repaso de Algoritmia

2.1 Ejercicio 1

i	j	suma
1	3	13
1	2	12
1	1	11
1	0	10
2	3	23
2	2	22
2	1	21
2	0	20
3	3	33
3	2	32
3	1	31
3	0	30
4	3	43
4	2	42
4	1	41
4	0	40

2.2 Ejercicio 2

El método recorre una vez el arreglo, y cada vez si encuentra que una vocal es menor que la siguiente la cambia de lugar (de esta forma coloca el caracter mayor al final) El resultado queda {'e', 'o', 'i', 'a', 'u'}

2.3 Ejercicio 3

```
1  public static void main(String[] args) {
2          Scanner sc = new Scanner(System.in);
3          int limite, cont, cantPrimos = 0;
4          boolean esPrimo;
5
6          System.out.println("Ingrese el limite de los numeros primos");
7          limite = sc.nextInt();
8
9          for (int i = 2; i <= limite; i++) {
10             //los primos son impares excepto por el 2
11             esPrimo = ((i % 2) != 0) || i == 2;
12             cont = 3;
13
14             while(esPrimo && cont <= Math.sqrt(i)) {
15                 esPrimo = (i % cont) != 0;
16                 cont++;
17             }
18
19             if(esPrimo) {
20                 cantPrimos++;
21             }
22         }
23
24         System.out.println("Entre 1 y " + limite + " hay " + cantPrimos + " numero
               primos");
25     }
```

2.4 Ejercicio 4

```java
public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        Random generador = new Random();
        boolean adivino;
        int num, intento, cant = 1;

        num = generador.nextInt(100) + 1;

        System.out.println("Adivina el numero! (entre 1 y 100)");
        intento = sc.nextInt();
        adivino = num == intento;

        while(!adivino) {
            if(intento < num) {
                System.out.println("No :(, el numero es mayor");
            } else {
                System.out.println("No :(, el numero es menor");
            }

            intento = sc.nextInt();
            adivino = num == intento;
            cant++;
        }

        System.out.println("Adivinaste! " + "Cantidad de intentos: " + cant);
    }
```

2.5 Ejercicio 5

Si tenemos que adivinar un numero que esta entre 1 y 1000000 (1 millón) de forma óptima, siempre deberíamos elegir el número del medio, y dependiendo si nos dice que el número es mayor o menor, elegimos el número del medio de esa mitad nuevamente, y así sucesivamente hasta obtener el número buscado. Esto es una búsqueda binaria En el peor de los casos, este algoritmo realizarán $\lfloor log2(n+1) \rfloor intentos, queenestecasoseriaigual a \lfloor log2(1000001) \rfloor = \lfloor 19.93157 \rfloor = 19$

2.6 Ejercicio 6

Tenemos distintos algoritmos de búsqueda:

- **Búsqueda secuencial**. Algoritmo más simple, en el cual buscamos uno a uno los elementos hasta encontrarlo o llegar al final de la estructura
- **Búsqueda binaria**. Algoritmo en donde tenemos como requisito que la estructura en la que están los datos esté ordenada. Entonces empezamos buscando del medio de la misma, si el elemento buscado es mayor al observado, entonces buscamos en la parte derecha y lo mismo si es menor en la izquierda. Repetimos el proceso recursivamente hasta encontrar el elemento o llegar al final

2.7 Ejercicio 7

En el algoritmo de quicksort se suele elegir al primer o último elemento del arreglo como pivote, pero esto en realidad podría ser cualquiera. Como en un comienzo el arreglo está siempre desordenado, lo único que habría que hacer para que funcione de la misma manera una vez elegido el pivote que sea es cambiar este de lugar con el primer (o último) elemento del arreglo, y ejecutar el código de igual manera

2.8 Ejercicio 8

```java
public static void main(String[] args) {
        Random random = new Random();
        int[][] notas = new int[4][5];
        double[] mediaAlumnos = new double[4];
        double[] mediaMateria = new double[5];
        double mediaTotal = 0;

        for (int i = 0; i < notas.length; i++) {
            for (int j = 0; j < notas[0].length; j++) {
                notas[i][j] = random.nextInt(11);
            }
        }

        System.out.println("Media de notas de los alumnos");
        for (int i = 0; i < notas.length; i++) {
            mediaAlumnos[i] = 0;
            for (int j = 0; j < notas[0].length; j++) {
                mediaAlumnos[i] += notas[i][j];
            }
            mediaAlumnos[i] /= notas[0].length;
            System.out.println(mediaAlumnos[i]);
        }

        System.out.println("Media de notas de las materias");
        for (int j = 0; j < notas[0].length; j++) {
            mediaMateria[j] = 0;
            for (int i = 0; i < notas.length; i++) {
                mediaMateria[j] += notas[i][j];
            }
            mediaMateria[j] /= notas.length;
            System.out.println(mediaMateria[j]);
        }

        System.out.println("Media total");
        for (int i = 0; i < mediaAlumnos.length; i++) {
            mediaTotal += mediaAlumnos[i];
        }
        mediaTotal /= mediaAlumnos.length;
        System.out.println(mediaTotal);
        //podemos clacular la media realizando lo mismo, pero con las materias

        System.out.println("Ordenamos medias de alumnos");
        //bubble sort
        for (int i = mediaAlumnos.length-1; i > 0; i--) {
            for (int j = 0; j < i; j++) {
                if(mediaAlumnos[j] > mediaAlumnos[j+1]) {
                    double aux = mediaAlumnos[j];
                    mediaAlumnos[j] = mediaAlumnos[j+1];
                    mediaAlumnos[j+1] = aux;
                }
            }
        }
        int min;
        //selection sort
        for (int i = 0; i < mediaAlumnos.length-1; i++) {
            min = i;
            for (int j = i+1; j < mediaAlumnos.length; j++) {
```

```
                if(mediaAlumnos[j] < mediaAlumnos[min]) {
                    min = j;
                }
                double aux = mediaAlumnos[i];
                mediaAlumnos[i] = mediaAlumnos[min];
                mediaAlumnos[min] = aux;
            }
        }
        for (int i = 0; i < mediaAlumnos.length; i++) {
            System.out.println(mediaAlumnos[i]);
        }
    }
```

2.9 Ejercicio 9

```
public static void main(String[] args) {
        Random random = new Random();
        int[] listaA = new int[100];
        int[] listaB = new int[60];
        int[] listaC = new int[listaA.length + listaB.length];
        boolean cambio = true;

        for (int i = 0; i < listaA.length; i++) {
            listaA[i] = random.nextInt(1000);
        }
        for (int i = 0; i < listaB.length; i++) {
            listaB[i] = random.nextInt(100);
        }

        //ordenamos listaA (insertion sort)
        for (int i = 1; i < listaA.length; i++) {
            int elem = listaA[i];
            int j = i - 1;
            while(j >= 0 && listaA[j] > elem) {
                listaA[j+1] = listaA[j];
                j--;
            }
            listaA[j+1] = elem;
        }

        //ordenamos listaB (bubble sort mejorado)
        int i = listaB.length-1;
        while(i > 0 && cambio) {
            cambio = false;
            for (int j = 0; j < i; j++) {
                if(listaB[j] > listaB[j+1]) {
                    int aux = listaB[j];
                    listaB[j] = listaB[j+1];
                    listaB[j+1] = aux;
                    cambio = true;
                }
            }
            i--;
        }

        //unificamos ambas listas ordenadas en listaC
        int posA = 0, posB = 0, posC = 0;
        while(posA < listaA.length && posB < listaB.length) {
```

```
44              if(listaA[posA] < listaB[posB]) {
45                  listaC[posC] = listaA[posA];
46                  posA++;
47              } else {
48                  listaC[posC] = listaB[posB];
49                  posB++;
50              }
51              posC++;
52          }
53          while(posA < listaA.length) {
54              //solo entro si termine listaB
55              listaC[posC] = listaA[posA];
56              posA++;
57              posC++;
58          }
59          while(posB < listaB.length) {
60              //solo entro si termine listaA
61              listaC[posC] = listaB[posB];
62              posB++;
63              posC++;
64          }
65
66          for (int j = 0; j < listaC.length; j++) {
67              System.out.print(listaC[j] + " - ");
68          }
69      }
```