Análisis de Algoritmos 2021

Trabajo Práctico 0 - Repaso

27 de agosto de 2021

Este curso tiene dos ejes conceptuales principales, el análisis de la correctitud y de la eficiencia de los algoritmos, por lo tanto necesitamos introducir prácticas de validación de los algoritmos que se implementen, así como la medición de tiempo de ejecución de los mismos.

Por ello, es importante generar casos de prueba apropiados que permitan evaluar el rendimiento algorítmico en condiciones límite, válidas para el problema, asegurando un comportamiento correcto en cada caso. Normalmente, estos casos de prueba serán tomados como lotes de entrada junto con un conjunto de resultados esperados para cada caso. Esto implica desarrollar buenos casos de prueba y una buena gestión de la entrada y salida de la información. En este curso trabajaremos fundamentalmente funcionalidad básida de entrada y salida a partir de archivos.

1. Entrada/Salida con Java

Para implementar los ejercicios siguientes:

1. Descargar el archivo *ejemplosInOut.zip*.
2. Crear un proyecto llamado *ejemplosInOut*
3. En el archivo *Ejemplo1InOut.java* hay una clase auto-explicativa de cómo usar clases de Java para gestión de archivos. En el archivo *Ejemplo2randomInts.java* hay una clase que explica cómo usar clases de Java para generar números aleatorios. Ambas clases son ejecutables auto documentadas. Lee su código, analizalo, ejecutalo y utilizalo para los ejercicios siguientes que lo requieran.

1. Utilizando un archivo de entrada que contenga un texto con varias palabras y varias líneas de texto y uno o más espacios en blanco entre ellas, generar *sinEspacios.txt* con el miso texto pero eliminando todos los espacios en blanco.

2. Con el mismo archivo de entrada, generar el archivo *lineasImpares.txt* con sólo las líneas impares del texto.

3. Generar un archivo de texto que contenga 100 números reales (double o float) generados aleatoriamente con valores entre -100 y 100.

4. Generar un archivo de texto con cadenas aleatorias de 10 caracteres alfanuméricos (0-9, a-z, A-Z).

5. Generar un archivo con números aleatorios de los números del 1 al 1000 sin que los elementos se repitan.

2. Repaso de Algoritmia

Para implementar los ejercicios siguientes:
En el archivo *Ejemplo3Scanner.java* hay una clase ejecutable autodocumentada que explica cómo usar las clases de Java Scanner que simplifica el manejo de la entrada. Lee el código, analizalo y ejecutalo antes de comenzar la siguiente sección.

1. Realiza detenidamente una traza al siguiente programa y muestra cuál sería la salida por pantalla:

```
ALGORITMO ej1
    VARIABLES
    suma,i, j:ENTERO
    PARA i ← 1 HASTA 4 HACER
        PARA j ← 3 HASTA 0 PASO -1 HACER
            suma ← i * 10 + j
            escribir(suma)
        FIN PARA
    FIN PARA
FIN ALGORITMO
```

2. ¿Qué imprime el siguiente programa?

```
class Ejercicio {
    public static void main (String [] args){
        char [] matriz←{'e','u','o','i','a'};
        metodo(matriz);
        for (int i←0;i<matriz.length;i++){
            System.out.print(matriz[i];
        }
    }
```

```
9       public static void metodo (char [] vocales){
10          char aux;
11
12          for (int i←1;i<vocales.length;i++){
13              if (vocales[i-1]>vocales[i]){
14                  aux←vocales[i-1];
15                  vocales[i-1]←vocales[i];
16                  vocales[i]←aux;
17              }
18          }
19      }
20  }
```

3. Realizar un programa que nos pida un número n y nos diga cuántos números primos existen entre 1 y n.

4. Realizar un juego para *adivinar* un número: Generar un número entero en forma aleatoria dentro de un rango, y luego, ir pidiendo números indicando *mayor* o *menor* según sea mayor o menor con respecto a n. El proceso termina cuando el usuario acierta.

5. Suponiendo $n \leq 1000000$ y un usuario que siga de forma óptima la lógica del juego y que quiera dar la menor cantidad de pasos hasta *adivinar* el número: ¿Cuál es el máximo número de intentos que puede necesitar el jugador hasta encontrar un número dentro del intervalo?

6. Liste y describa claramente los algoritmos para la resolución del problema de búsqueda que conoce. (tip: recordar distintas implementaciones de interfaz TablaDeBusqueda).

7. Al ordenar una lista de números enteros aplicando el algoritmo quicksort, como pivote se elige el primer elemento de la lista. ¿qué pasaría si se selecciona otro pivote?

8. Resolver el siguiente problema escolar. Dadas las notas de los alumnos de un colegio en el primer curso de bachillerato, en las diferentes asignaturas (5 por comodidad) se desea:

   a) Calcular la media de cada alumno.

   b) Calcular la media de cada asignatura.

   c) Calcular la media total de la clase.

   d) Ordenar el listado de los alumnos por orden decreciente de notas medias individuales, que incluyen todas las materias.

   Nota: Utilizar dos algoritmos de ordenación diferentes para resolver el problema, justificando la elección. Generar los casos de prueba antes de hacer el ejercicio.

9. Se leen dos listas de números enteros, A y B de 100 y 60 elementos, respectivamente. Se desea resolver mediante procedimientos las siguientes tareas:

a) Ordenar aplicando un método de ordenación distinto a cada una de las listas A y B

b) Crear una lista C a partir de la mezcla de las listas A y B ya ordenadas.

c) Mostrar la lista C