

# **udp** FACULTAD DE INGENIERÍA Y CIENCIAS

## **LABORATORIO N°1**

Asignatura: Estructura Datos y Algoritmos

Profesor: Martin Gutierrez

---

Carrera: Ingeniería Civil Informática y Telecomunicaciones

---

Alumnos: Valentina García, Vicente Pérez

---

Fecha: 2 de Abril del 2021

---

## **Análisis de complejidad computacional del código**

En el presente proyecto se tiene que organizar de manera óptima los datos provenientes de un archivo de dataset que describe los productos más vendidos en Amazon en el país de México, con parámetros como tiempo, clasificación, nombre del producto, estrellas, comentarios, etc, es decir, características específicas, de dicho producto, estos son representados en columnas, y por otro lado en las filas se muestran los productos individuales.

La idea de la organización es volcar los productos provenientes de distintos datasets, tanto de manera creciente como decreciente, además de mostrar en pantalla los productos más solicitados por categoría y finalmente mostrar los productos ordenados de forma creciente.

Para dar solución al problema se implementó un código que tiene como finalidad el uso de pilas y colas, se analizará con la notación "Big O". mecanismo que nos permitirá saber el comportamiento del código en el peor de los casos.

Se implementa un ciclo para abrir los archivos, ingresarlos a un ArrayList, para luego usar una estructura llamada tripleta que nos servirá para ordenar los productos según cuantas veces sean comprados

Diremos que el número máximo que se puede ejecutar el for es “ $n$ ” veces, entonces tenemos que de la (figura 1) hay 43 líneas de código que por estar dentro de un ciclo, pasan a ser “ $43n$ ”, ya que el crecimiento de esta función es de carácter lineal, es decir, el tiempo de ejecución es proporcional al tamaño de la entrada.

```

40      for(String aux : arr){
41          int i = 0;
42          if(cont == 0){
43              nombre = "amazon_devices";
44          }else if(cont == 1){
45              nombre = "automotive";
46          }else if(cont == 2){
47              nombre = "baby";
48          }else if(cont == 3){
49              nombre = "books";
50          }else if(cont == 4){
51              nombre = "digital_text";
52          }else if(cont == 5){
53              nombre = "dvd";
54          }else if(cont == 6){
55              nombre = "electronics";
56          }else if(cont == 7){
57              nombre = "grocery";
58          }else if(cont == 8){
59              nombre = "handmade";
60          }else if(cont == 9){
61              nombre = "hpc";
62          }else if(cont == 10){
63              nombre = "kitchen";
64          }else if(cont == 11){
65              nombre = "music";
66          }else if(cont == 12){
67              nombre = "musical_instruments";
68          }else if(cont == 13){
69              nombre = "officeproduct";
70          }else if(cont == 14){
71              nombre = "pet_supplies";
72          }else if(cont == 15){
73              nombre = "shoes";
74          }else if(cont == 16){
75              nombre = "software";
76          }else if(cont == 17){
77              nombre = "sports";
78          }else if(cont == 18){
79              nombre = "tools";
80          }else if(cont == 19){
81              nombre = "toys";
82          }else if(cont == 20){
83              nombre = "videogames";
84          }

```

Figura 1: Código Lab1.java desde línea 40 al 84.

Se entra a la creación de un ArrayList de ArrayLists( Figura 2) que nos ayudará a almacenar los datos entregados por el dataset, donde en primer lugar, nos enfocaremos en los ciclos, se tiene un while que está dentro del ciclo anterior mencionado, por ende sería un " $O(n^2)$ ", es decir, su crecimiento es cuadrático, y todas sus funciones que están dentro de él también serían " $O(n^2)$ ", el ciclo de la línea 95 corresponde a un " $O(n^3)$ " ya que se encuentra dentro de él. Posteriormente en la línea de código 101 se crea otro ciclo con notación " $O(n^2)$ ", por los mismos motivos del while, pero dentro hay otro ciclo que tendrá como notación un " $O(n^3)$ " y será de crecimiento cúbico por ende, todo lo que está incluido ahí será " $O(n^3)$ ".

En resumen tenemos:

- $47n$  (sumando las anteriores)
- $9n^2$
- $5n^3$

```

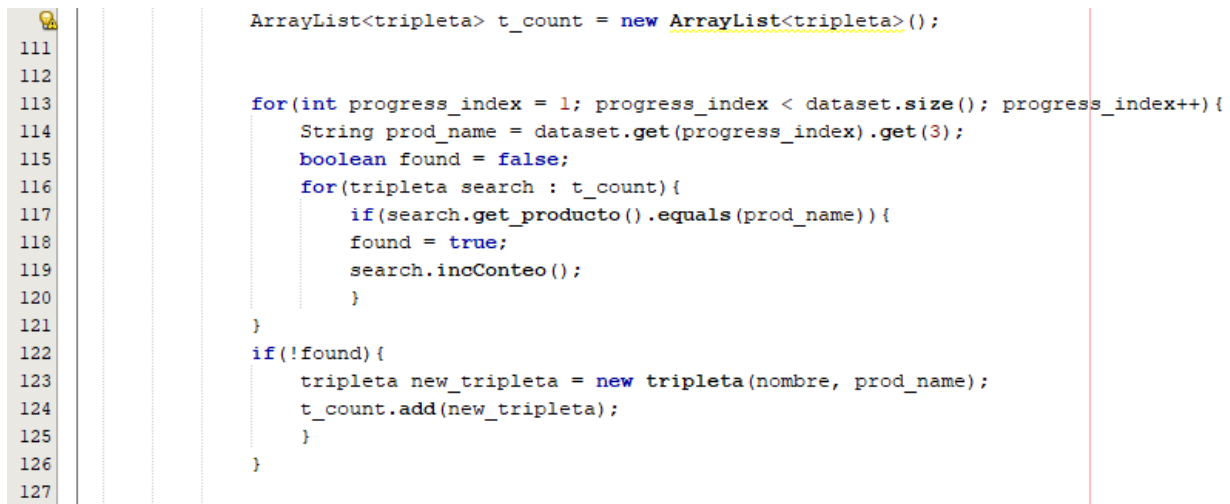
86 ArrayList<ArrayList<String>> dataset = new ArrayList<ArrayList<String>>();
87 reader = new BufferedReader(new FileReader(aux));
88 writer3 = new BufferedWriter(new FileWriter("mx-"+nombre+"_salida.csv", false));
89
90 String line = reader.readLine();
91 while (line != null){
92     ArrayList<String> parsing1 = new ArrayList<String>();
93     String[] row1;
94     row1 = line.split("\\|",-1);
95     for(String x : row1){
96         parsing1.add(x);
97     }
98     dataset.add(parsing1);
99     line = reader.readLine();
100 }
101 for(ArrayList a : dataset) {
102     for( i = 0; i < a.size()-1; i++){
103         writer3.write(a.get(i).toString());
104         writer3.write("|");
105     }
106     writer3.write(a.get(i).toString());
107     writer3.write("\n");
108 }
109

```

Figura 2: Código Lab1.java desde línea 85 al 109.

A continuación en (Figura 3) se crea un ArrayList de tripletas, que nos servirá para conocer cuántas veces fueron comprados los productos de las distintas categorías, analizando esto sería, un " $O(n)$ " representando a la línea donde se crea "t\_count", luego en el ciclo se tiene 6 líneas de " $O(n^2)$ ", y 4 líneas de " $O(n^3)$ ", entonces:

- $48n$
- $15n^2$
- $9n^3$



```

111 ArrayList<tripleta> t_count = new ArrayList<tripleta>();
112
113 for(int progress_index = 1; progress_index < dataset.size(); progress_index++){
114     String prod_name = dataset.get(progress_index).get(3);
115     boolean found = false;
116     for(tripleta search : t_count){
117         if(search.get_producto().equals(prod_name)){
118             found = true;
119             search.incConteo();
120         }
121     }
122     if(!found){
123         tripleta new_tripleta = new tripleta(nombre, prod_name);
124         t_count.add(new_tripleta);
125     }
126 }
127

```

Figura 3: Código Lab1.java desde línea 110 al 127.

Posteriormente, en la figura 4, se muestra como se ordenan los productos a través del método insertion sort de manera decreciente, tomando en cuenta los datos anteriores, se obtiene lo siguiente:

- $49n$
- $20n^2$
- $12n^3$

```

128
129         int largest;
130
131         for (int k=0; k < t_count.size () - 1; k++)
132         {
133             largest = 0;
134             for (int j=largest + 1; j < t_count.size () - k; j++)
135             {
136                 if ((t_count.get (largest)).compareTo (t_count.get (j)) < 0)
137                 {
138                     largest = j;
139                 }
140             }
141             tripleta tempTrip = t_count.get (t_count.size () - 1 - k);
142             t_count.set (t_count.size () - 1 - k, t_count.get (largest));
143             t_count.set (largest, tempTrip);
144         }
145

```

Figura 4: Código Lab1.java desde línea 128 al 145.

Después, se ingresan las tripletas dentro de una cola, y se imprimen en un archivo txt los productos más vendidos de las distintas categorías, analizando y considerando los datos anteriores tenemos:

- $53n$
- $25n^2$

Manteniéndose los  $12n^3$ .

```

146         int aux2 = 0;
147         for(tripleta t_d : t_count){
148             colal.enqueue(t_d);
149             if(aux2 == 0){
150                 writer1.write(t_d.get_categoria() + "/" + t_d.get_producto() + "/" + t_d.get_conteo() + "\n");
151                 aux2 ++;
152             }
153
154         }
155
156
157
158
159         reader.close();
160         writer3.close();
161         cont ++;
162

```

Figura 5: Código Lab1.java desde línea 146 al 162.

Finalmente, se invierte la cola en una pila, imprimiendo en otro archivo txt los productos más solicitados en orden creciente, por último se tiene lo siguiente:

- $59n$
- $4$

Manteniéndose  $25n^2$  y  $12n^3$ .

```

163         while(!colal.isEmpty()){
164             pilal.push((tripleta)colal.dequeue());
165         }
166     }
167
168     while(!pilal.isEmpty()){
169         tripleta t_p;
170         t_p = (tripleta)pilal.pop();
171
172         writer2.write(t_p.get_categoria() + "/" + t_p.get_producto() + "/" + t_p.get_conteo() + "\n");
173     }
174
175     writer1.close();
176     writer2.close();
177
178     }catch(IOException e){
179         e.printStackTrace();
180     }
181 }
182
183
184
185
186

```

Figura 6: Código Lab1.java desde línea 163 al 186.

Obteniendo finalmente la siguiente ecuación:

$$4 + 59n + 25n^2 + 12n^3.$$

En conclusión sólo consideramos  $n^3$ , ya que los otros datos son insignificativos para nuestro cálculo, para que después nos quede  $O(n^3)$ , lo que significa que en términos de notación Big O obtenemos  $O(n^3)$ , es decir, la búsqueda se ejecutará en un tiempo de  $O(n^3)$ , el código se haría más eficiente si en vez de ocupar Insertion Sort para ordenar los productos, se hubiese utilizado otro tipo de ordenamiento con menos ciclos dentro de sí, algo que lo hubiese empeorado es que ocupáramos BubbleSort por ejemplo, para ordenar ya que es un método que incluye muchos ciclos dentro de otros, lo que hubiese aumentado el tiempo de ejecución.

## BIBLIOGRAFÍA:

Cormen T, Balkcom D. (s.f.). *Notación O grande (Big-O)*. Obtenido de Khan academy  
<https://es.khanacademy.org/computing/computer-science/algorithms/asymptotic-notation/a/big-o-notation>

Guitierrez, M. (Marzo de 2021). *tripleta.java*. Obtenido de la Universidad Diego Portales  
<https://udp.instructure.com/courses/7772/files/folder/Laboratorio/Lab%201>

Guitierrez, M. (Marzo de 2021). *Volcado0.java*. Obtenido de la Universidad Diego Portales  
<https://udp.instructure.com/courses/7772/files/folder/Laboratorio/Lab%201>

López, E. T. (September de 2020). *Amazon Best Sellers*. Obtenido de Kaggle:  
<https://www.kaggle.com/edwardtoledolpez/amazon-mexico-top-50-best-sellers>

Sedgewick R, Wayne K. (s.f.). *cola.java*. Obtenido de la Universidad Diego Portales  
<https://udp.instructure.com/courses/7772/files/folder/Laboratorio/Lab%201?>

Sedgewick R, Wayne K. (s.f.). *pila.java*. Obtenido de la Universidad Diego Portales  
<https://udp.instructure.com/courses/7772/files/folder/Laboratorio/Lab%201?>

unknown, u. (Marzo de 2011). *sorting an arraylist of objects using insertion sort*. Obtenido de  
stackoverflow:  
[https://stackoverflow.com/questions/5346500/sorting-an-arraylist-of-objects-using-insertio](https://stackoverflow.com/questions/5346500/sorting-an-arraylist-of-objects-using-insertion-sort)  
n-sort