# Tarea 2 - Análisis de Algoritmos y Estructura de Datos

Catalina Riquelme Zamora
*Departamento de Ingeniería Informática*
*Universidad de Santiago de Chile*, Santiago, Chile
1-2021

## I. Introducción

Tras el inicio de la pandemia de COVID-19, en donde se vio afectada casi en la totalidad los países, se desarrollaron diversas vacunas, por lo que en el centro de vacunación Clotario Blest de la comuna de Maipú desea realizar un registro de las personas inoculadas dentro de su centro, ordenadas por fecha de vacunación y apellido de manera ascendente. Además de definir una estrategia con el fin de calcular cuándo será necesaria una segunda dosis para cada una de estas personas, dependiendo de la vacuna administrada. El centro dispone de tres diferentes vacunas para administrar, en la siguiente imagen se muestran sus características.

| Nombre de la vacuna | Laboratorio | Tiempo entre primera y segunda dosis |
|---|---|---|
| BNT 162B2 | Pfizer | 3 semanas |
| CoronaVac | Sinovac | 4 semanas |
| AZD1222 | Oxford/AztraZeneca | 10 semanas |

Figura 1: Vacunas aprobadas en Chile

## II. Solución propuesta

Para resolver este problema es necesario definir tres listas enlazadas, una de ellas contendrá las personas que se encuentran vacunadas con una dosis, otra en donde se encuentren las personas que ya disponen de ambas dosis y por último una lista enlazada en donde se encuentren las vacunas existentes. Todas las estructuras se compondrán de dos partes tal como se puede apreciar en las figura 2,3 y 4, en donde uno de los datos va ser otra estructura que hace referencia a los datos técnicos de esta del componente de la lista, a modo de ejemplo para el caso de las personas que se encuentran registradas con una sola dosis se almacenará rut, nombre, apellido, edad, fecha en que se colocara la próxima dosis y el id de la vacuna con el que fue inoculado. La segunda parte de este nodo es un punto que apunta al siguiente nodo de la lista.

Para lograr el objetivo de esta tarea, el algoritmo se dividirá en tres partes:

La primera se encuentra relacionada con las vacunas completas, en donde se busca escribir un archivo

```
Estuctura vacunados1dosis:
  rut: char
  nombre: char
  apellido: char
  edad: char
  fecha1dosis: char
  idVacuna: char

Estructura nodo1D:
  vacunados1D: vacunados1dosis
  siguiente: nodo1D
```

Figura 2: Lista primera dosis

```
Estuctura vacunados1dosis:
  rut: char
  nombre: char
  apellido: char
  edad: char
  fecha1dosis: char
  idVacuna: char

Estructura nodo1D:
  vacunados1D: vacunados1dosis
  siguiente: nodo1D
```

Figura 3: Lista primera dosis

vacunacionCompleta.out el cual muestre las personas que ya han sido inoculada dos veces, para lograr esto se recorre en primera instancia la lista en donde se ven ubicados las personas que cuentan con dos dosis, cada vez que se revise una de estas personas se realza la vez un ciclo aparte con el fin de recorrer las personas que han recibido su primera dosis en el establecimiento buscando si la persona seleccionada se encuentra registrada, para posteriormente escribir esto en un archivo, logrando mostrar todas las personas que cuentan con su segunda dosis y si alguna de ellas también se vacuna en ese lugar se encontrará en el archivo también.La complejidad de este algoritmo es de orden cuadrática debido al doble ciclo while que se presenta al recorrer la lista de vacunados con una dosis y dos dosis, ya que en el peor de los casos por cada persona registrada en la lista de dos dosis se deberá revisar cada una de las personas registradas en la lista con una dosis. El pseudocódigo de este algoritmo se ve reflejado

```
Estructura vacunas:
  numero: char
  nombre: char
  fabricante: char
  periodo: char

Estructura nodoVacunas:
  vacunas: vacunas
  siguiente: nodoVacunas
```

Figura 4: Lista vacunas

en la Figura 5.

```
...Al momento de escribir se escribe cada uno de los aspectos de la estructura
vacunacionCompleta(inicio1D,inicio2D)
   ...Se recorre la lista con la segunda dosis escribiendo el archivo a mediada que se recorre
   if !esListaVacia2D(inicio2D) then:
      auxiliar<-inicio2D
      while auxiliar <> NULL do:
        write(auxiliar->vacunados2D)
        ...Se busca si la persona fue vacunada en el mismo recinto con la primera dosis
        ...(se recorre lista con primera dosis)
        inicio2<-inicio1D
        persona<-auxiliar
        if (!esListaVacia1D(inicio1D))
          auxiliar2<-inicio1D;
          while auxiliar2 <> NULL do:
            ...En el caso que se encuentre la misma persona, se escribe y rompe el ciclo
            if auxiliar2->vacunados1D.rut = persona->vacunados2D.rut then:
              write(vacunados1D)
              break

            auxiliar2<-auxiliar2->siguiente

        auxiliar<-auxiliar->siguiente
```

Figura 5: Algoritomo vacunación completa. Complejitdad O($n^2$)

La segunda parte se encuentra relacionada con la escritura de un archivo que almacena un listado con las personas que se encuentran vacunadas con una dosis de la vacuna, está lista se encuentra ordenado de manera cronológica, para lograr esto se aplica un ordenamiento tipo burbuja a lista enlazada con la finalidad de ordenar de manera ascendente, en primera instancia se ordena por año, luego por mes y finalmente. En el caso que dos o más personas compartan día de vacunación estos son ordenados en primera instancia por apellido de manera ascendente, en el caso siga existiendo coincidencia se ordena por nombre de manera ascendente. Todos los ordenamientos funcionan de manera similar usando de fuente el ordenamiento tipo burbuja. Este ordenamiento presenta una complejidad tipo cuadrática debido a que presenta dos ciclos while en donde en el peor de los casos se deberá por cada elemento del primer ciclo realizar un ciclo.El pseudocódigo de este algoritmo se ve reflejado en la Figura 6.

Finalmente se busca escribir un archivo provision.out en donde se indique cuántas vacunas se necesitan los próximos meses, para esto se crea una estructura year que iniciará en cero en todos los meses que la componen,esta estructura se encuentra representada en la Figura 7, posteriormente se recorre la lista de personas que se encuentran vacunadas con una sola dosis y se calcula cuando necesitan las siguiente dosis, una vez calcula se almacena en en la estructura en el

```
...Ordenamiento génerico (dato se puede reemplazar por la
...comparación de fecha, apellido y nombre
ordenamientoBurbuja(inicio)
    nodo2<-NULL
    swapped <- 0
    while swapped do:
        swapped <-0
        nodo1<-inicio

        while nodo1->siguiene <>nodo2 do:
            aux<- nodo1
            if nodo1->dato> nodo1->siguiene->dato then:
                intercambiar(nodo1,nodo1->siguiente)
                swapped <-1

            nodo1<- nodo1 -> siguiente
        nodo2 <- nodo1
```

Figura 6: Ordenamiento burbuja. Complejitdad O($n^2$)

mes que le corresponda, para que al final del recorrido la estructura cuente con la suma de todas las vacunas separadas por meses.Este algoritmo presenta una complejidad O(n) debido a que se realiza un único ciclo while para recorrer el listado, siendo esta la parte más significativa.El pseudocódigo de este algoritmo se ve reflejado en la Figura 8.

```
Estructura year:
  enero: int
  febrero: int
  marzo: int
  abril: int
  mayo: int
  junio: int
  julio: int
  agosto: int
  septiembre: int
  octubre: int
  noviembre: int
  diciembre: int
```

Figura 7: Estructura year

```
provision(lista1D,inicioVacunas):estructura
    proviciones<-seCreaEstructura()
    ...Se recorre la lista con una dosis calculando cuando será la proxima dosis y almacenandola en el mes
    ...que corresponde
    if !esListaVacia1D(lista1D) then:
      auxiliar<-lista->siguiente...nodo1D
      ...Se recorre la lista que neceseitarán una segunda dosis (vacunados con una dosis)
      while auxiliar<>NULL do:
        ...Se busca el periodo en las lista que contiene todas las vacunas
        periodo <- peridoSegunVacuna(inicioVacunas,auxiliar->vacunados1D.idVacuna)...retorna semanas
        ..Se calcula fecha segunda dosis
        fechaSalida <- proximaVacuna(fecha1,periodo)..retorna array con fecha [dia,mes,año]
        ..Se almacena en la estructura
        provisiones<- aumentoMes(fechaSalida[1],provisiones)
        auxiliar<-auxiliar->siguiente;

    return (provisiones)
```

Figura 8: Algoritmo provision.Complejidad O(n)

## III. Resultados y análisis

Al evaluar el tiempo en ejecutar el algoritmo vacunación completa evaluando diversos tamaños de entrada (misma cantidad de información en ambos archivos de entrada) se pueden apreciar los resultados en las siguientes tablas y

gráficos:

| n | Tiempo (s) |
|---|---|
| 5 | 0.013 |
| 10 | 0.025 |
| 10 | 0.064 |

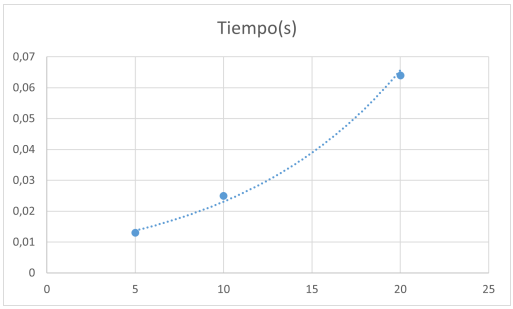


Figura 9: Tiempos de ejecuión vacunacionCompleta

Tras apreciar los resultados mostrados en la tabla y en gráfico en la Figura 9 se concluye que existe un aumento exponencial entre el aumento del tiempo y el aumento de la dimensión de la entrada, esto debido a que como se puede apreciar en la tabla al aumentar la entrada esto genera un aumento en los ciclos a realizar. Con esto se puede confirmar que la complejidad teórica(complejidad cuadrática) calza con el tiempo al evaluar el algoritmo vacunacionCompleta.

| n | Tiempo (s) |
|---|---|
| 5 | 0.017 |
| 10 | 0.033 |
| 10 | 0.098 |

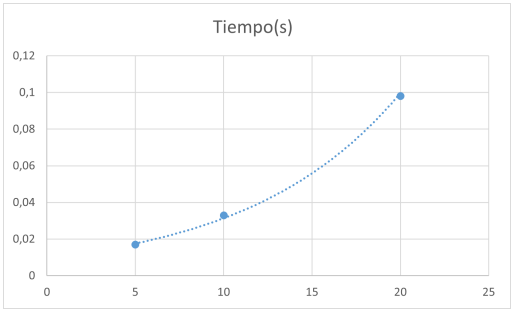


Figura 10: Tiempos de ejecución ordenamiento burbuja.

Para evaluar el tiempo en el ordenamiento se consiero al evaluar el ordenamiento tanto cronologico, como por apellido y nombre, con los resultados mostrados en la tabla y en gráfico en la Figura 10 se concluye que existe un aumento exponencial entre el aumento del tiempo y el aumento de la dimensión de la entrada, esto debido a que como se puede apreciar en la tabla al aumentar la entrada esto genera un aumento en los ciclos a realizar. Con esto se puede confirmar que la complejidad teórica(complejidad cuadrática) calza con el tiempo al evaluar el algoritmo de ordenamiento.

| n | Tiempo (s) |
|---|---|
| 5 | 0.045 |
| 10 | 0.066 |
| 10 | 0.108 |

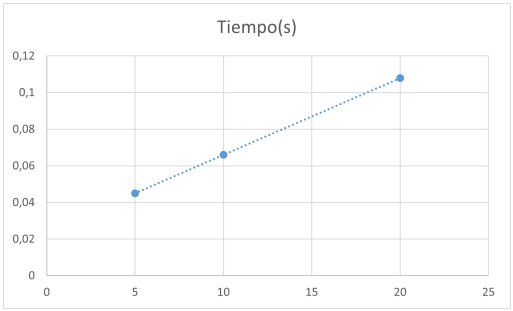


Figura 11: Tiempos de ejecución algoritmo provisiones.

AL apreciar los resultados mostrados en la tabla y en gráfico en la Figura 11 se concluye que existe un aumento lineal entre el aumento del tiempo y el aumento de la dimensión de la entrada, esto debido a que como se puede apreciar en la tabla al aumentar la entrada esto genera proporcial en el tiempo. Con esto se puede confirmar que la complejidad teórica(complejidad lineal) calza con el tiempo al evaluar el algoritmo provisión.

## IV. Conclusiones

A través del programa realizado en c, se logró escribir tres archivos listado.out,provision.out y vacunacionCompleta.out de manera exitosa usando listas enlazadas para este propósito. Al realizar la toma de tiempos de las tres principales funciones todas coincidieron con la complejidad propuesta de manera teórica lo que indica que fueron calculadas de manera correcta.

Una de las desventajas del algoritmo propuesto para realizar esta tarea es la utilización del ordenamiento el cual es más lento en comparación a otros tipos de ordenamientos, para optimizar el algoritmo se debería implementar un algoritmo de ordenamiento QuickSort o MergeSort reduciendo considerablemente el tiempo de ejecución del programa. Una de las ventajas del algoritmo es el uso de de diversas listas mejorando la comprensión del código y permitiendo en futuras alteraciones quitar y agregar elementos con facilidad.

Para mejorar el trabajo el próximas entregas es necesario empezar el algoritmo con anterioridad para evitar que el tiempo afecte la calidad de la entrega final, además de obtener un mayor conocimiento respecto al uso de punteros antes de empezar a trabajar.

## V. Manual de usuario

Para ejecutar de manera correcta el programa se requieren tres archivos los cuales deben tener el nombre

vacunados1D.in, vacunados2D.in y vacunas.in, el incorrecto nombramiento de los archivos puede causar una caída en el programa. Dentro de los archivos vacunados1D.in y vacunados2D.in, debe existir el número de personas registradas en la primera línea, posteriormente en cada línea debe ingresarse los datos de la persona vacunada siguiendo el siguiente ejemplo (19676900-5 Josefina Sepulveda 23 03/05/21 1), donde el primer dato corresponde al rut, seguido por el nombre, apellido, edad, fecha de inoculación e id de la vacuna administrada, ingresar de manera incorrecta alguno de estos parámetros puede causar la caida del programa o la mala salida de alguno de los archivos. Por otra parte para el archivo vacunas.in se debe colocar en la primera linea la cantidad de vacunas existentes en el archivo, en las líneas posteriores se ingresa la información de cada una siguiendo el ejemplo (2 Coronavac Sinovac 4), en donde el primer dato corresponde al identificador de la vacuna, el siguiente al nombre, fabricante y número de semanas en la cual se debe administrar la segunda dosis, en caso de ingresar mal estos parámetros puede resultar en la caida del programa o en la salida incorrecta de los archivos.

El programa al ser ejecutado crea tres archivos listado.out, provision.out y vacunacionCompleta.out. El archivo listado.out contendrá un listado con las personas inoculada con una dosis y la fecha de la próxima dosis, esta fecha se encuentra registrada de manera que el primer dato es el mes y el segundo el día siguiendo el modelo mes/dia/año.Por otra parte el archivo provision.out contendrá las vacunas que deberán ser administradas en las siguientes meses. Finalmente el archivo vacunacionCompleta.out contendrá un listado de las diversas personas que se han administrado la segunda dosis con éxito.