Tarea 1 - 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, el cual atiende de lunes a viernes, de 9:00 a 13:00 hrs, se busca aprovechar al máximo el espacio físico y profesionales disponibles para realizar una inoculación de una manera óptima. Este centro cuenta con 4 box, en donde cada box dependiendo del tamaño tiene un aforo máximo permitido, el cual debe ser sanitizado cada vez que se cumple cierta cantidad de pacientes que hayan ingresado. Las características son especificadas en la siguiente tabla:

Box	Aforo máximo	Sanitización	Tiempo de sanitización
Box1	4 personas	Cada 12 vacunadas/os	4 minutos
Box2	4 personas	Cada 12 vacunadas/os	4 minutos
Box3	7 personas	Cada 21 vacunadas/os	6 minutos
Box4	9 personas	Cada 27 vacunadas/os	8 minutos

Figura 1: Características de cada box

Cada profesional demora 4 minutos en atender a una persona, en cambio si hay dos profesionales en el box pueden atender a cada paciente en 2 minutos y si hay tres profesionales el tiempo será de un minuto por paciente, en el caso que existan más de tres profesionales no hay ganancia en tiempo. Es importante destacar que en cambio de profesionales de un box a otro toma 1 minuto.

II. Solución propuesta

Para resolver este problema es necesario encontrar un algoritmo que dado un determinado número de profesionales de salud, en los espacios dados por el centro, busque maximizar la cantidad de pacientes inoculados por día, respetando los tiempos de sanitización a aforos otorgados. Se utilizará búsqueda en espacio de estados (BEE) por anchura en programa realizado en c

Por lo que primero se definen dos estructuras una para box y otra de estados

```
Tipo estados:
    array: box
    horaSanitizacion: array
    id: entero
    sanitizacion: entero
    estadoAnterior: entero
    estado: string
    tiempo: entero
```

Figura 2: Estructura estados

```
Tipo box:
    profesionales: entero
    pacientes: entero
    estadoSanitizacion: entero
    sanitizacion: entero
    tiempoSanitizacion: entero
    bitacoraBox: string
```

Figura 3: Estructura box

Es importante destacar las transiciones que existen y el estado final que tendrá este BEE, en este caso las transiciones se basaron en el cambio de box a box, por lo que al cuatro box, se puede por ejemplo ir de el box 1 al box 2, tanto como al box 1 al box 3 y por último del box 1 al box 4, por lo que al repetir este proceso en todos los box se pueden obtener 12 transiciones de box a box, además de estas transiciones ya explicadas existen otras 4 en donde el personal ingresa al los box sin haber pasado a uno anterior necesariamente agregando nuevas posibilidades de transiciones, dejando un total de 16 transiciones.

- Transiciones
  1. B0B1
  2. B0B2
  3. B0B3
  4. B0B4
  5. Box1 a Box2

6. Box1 a Box3
7. Box1 a Box4
8. Box2 a Box1
9. Box2 a Box3
10. Box2 a Box4
11. Box3 a Box1
12. Box3 a Box2
13. Box3 a Box4
14. Box4 a Box1
15. Box4 a Box2
16. Box4 a Box3

```
...Objetivo: Verificar tarnsicion box n a box m
verificarTransicion(estados):num
    ...Verifica que el box de llegada este vacio
    if(estado.array[m].profesionales = 0
    ...además que el box de salida tenga personas
    and estado.array[n].profesionales <> 0
    ...y por último que box de llaga no se encuentre en estado
    ...de sanitización (0)
    and estado.array[m].estadoSanitizacion = 0) then:
        return 1

    return 0
```

Figura 4: Verificación de transición

```
...Objetivo: crear un estado con la transicion de box n a box m
...Entrada: estado(estados), cantidadProfeisonales(numero)
...Salida: función con datos actualizados
BnBm(estados ,numero):funcion
    ...Se calcula el tiempo que se demora el personal según el cantidad
    tiempo <- tiempoSegunPersonal(cantidadProfesionales)*estado.array[m].sanitizacion
    ...Se vacia el box n con los profesionales
    estado.array[n].profesionales <- 0
    ...Se llena el box m con los profesionales
    estado.array[m].profesionales <- cantidadProfesionales
    ...Se actualiza los pacientes (pacientes ya atendidos + nuevos)
    estado.array[m].pacientes <- estado.array[m].pacientes + estado.array[m].sanitizacion
    ...Cuando termine el box comenzará a sanitizar (1)
    estado.array[m].estadoSanitizacion <- 1
    ...Se actualiza el tiempo transcurrido
    estado.horaSanitizacion[m] = estado.tiempo+tiempo
    bitacora <- ""
    colocarDatosEnBitacora(bitacora)
    //Se alamcena la bitacora creada
    estado.array[1].bitacoraBox <- bitacora

    return crearEstado(estado.array[0],estado.array[1],estado.array[2],estado.array[3],
    estado.id,"BnBm",estado.tiempo+tiempo+1)
```

Figura 5: Transición

Como se puede apreciar en el pseudocódigo, en primera instancia se definen los estados iniciales de cada uno de los box, tomando en cuenta los valores que restringen en sí a cada uno como lo son el número de pacientes que tienen que pasar por cada box para que sea sanitizado, junto con el tiempo que le toma realizar este proceso ya que de esto determinará las posibles combinaciones de horarios del personal. Se crean estados dinámicos para los estados abiertos y cerrados, estos serán dinámicos, además se agrega el estado inicial para posteriormente entrar en ciclo que se verá finalizado solamente cuando no queden estados abiertos, dentro de este ciclo que determinará primero que todo si el estado es el final, este consiste en que el tiempo se ve agotado y los profesionales no pueden seguir trabajando debido a que finalizó su horario laboral, en el caso de no caer en esta iteración se procede a revisar las posibles transiciones abriendo y cerrando estados con las diversas combinaciones que pueden existir.

```
vacunacion (numero): char
    tiempoTotal <- 240
    ...(profesionales,pacientes,sanitizacion,tiempoSanitizacion,
    ...estadoSanitizacion,bitacoraBox)
    box1 <- createBox(0,0,12,4,0,"B1 ")
    box2 <- createBox(0,0,12,4,0,"B2 ")
    box3 <- createBox(0,0,21,6,0,"B3 ")
    box4 <- createBox(0,0,27,8,0,"B4 ")
    bitacora <- ""
    correlativo <- 0

    ...Variable para guardar la cantidad estados se encuentran dentro del arreglo abiertos
    canAbiertos <- 0
    ...Variable para guardar la cantidad estados se encuentran dentro del arreglo cerrados
    canCerrados <- 0
    abiertos <- []
    cerrados <- []
    ...Estados iniciales de los estados
    inicial <- crearEstado(box1,box2,box3,box4,correlativo,"",0)
    ...En abiertos se agregar el estado inical
    abiertos <- agregarEstado(abiertos, canAbiertos, inicial)

    ...Mientras queden estados abiertos se ejecutará
    while (canAbiertos > 0) do:
        bActual <- abiertos[0];
        abiertos <- sacarElemento(abiertos, canAbiertos)
        cerrados <- agregarEstado(cerrados, canCerrados, bActual)

        ...Estado final
        if(esFinal(bActual,tiempoTotal) = 1) then:
            write(bitacora)
            return bitacora

        ...En caso que no sea el caso final
        ...se recorren las posibles transiciones
        else:

            ...En caso que cumpla con las condiciones para entrar determinada
            ...trancisión se puede evaluar
            if (ejecucionValidaciones(bActual) = 1) then:
                bSiguiente <- transicion(bActual,numeroProfesionales)
                ...Se procura que el estado no se encuentre repetido
                if((estaEstado(abiertos,canAbiertos,bSiguiente) = 0)
                and (estaEstado(cerrados,canCerrados,bSiguiente) = 0)) then:
                    abiertos <- agregarEstado(abiertos,canAbiertos,bSiguiente)

    return bitacora
```

Figura 6: Pseudocódigo

III. Resultados y análisis

Al evaluar el tiempo que tarda en ejecutar el código evaluando las diversas tamaños de entrada se pueden apreciar los resultados mostrados en la siguiente tabla:

n	Tiempo (s)	Estados abiertos
1	0.341000	19
2	2.17700	90
3	3.107000	187

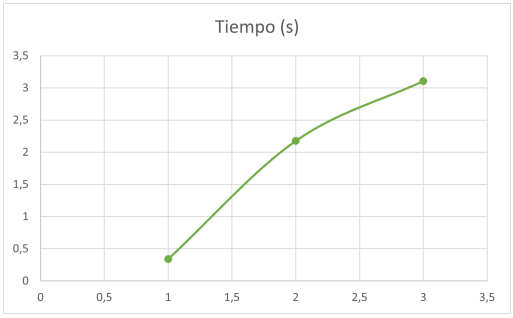

Figura 7: Tiempos de ejecuión

Tras apreciar los resultados mostrados en la tabla y en gráfico se concluir que existe un aumento proporcional entre el aumento del tamaño 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 estados que se deben abrir para realizar el programa generando así un aumento sustancial en el tiempo ya que se deben analizar cada uno.

Además al existir un aumento en el tiempo dado un número n, se deduce que el código presenta una complejidad O(n) debido que a medida que aumenta el número ingreso avanza el tiempo de ejecución proporcionalmente.

IV. Conclusiones

A través del programa realizado en c, se logró optimizar los pacientes atendidos durante las horas determinadas determinando cual es la mejor ruta para los profesionales que trabajan en el establecimiento de la salud, llegando así a la mayor cantidad de personas inoculadas en un día a través de una búsqueda en espacio de estados por anchura. Al variar la dimensión de la entrada se puede apreciar un aumento proporcional entre el tiempo y la entrada generado por el aumento de estados generados por el programa.

Si bien el programa logra trabajar bien con uno, dos y tres profesionales, para una cantidad mayor no logra encontrar la manera óptima de trabajar esto debido a que no se pudo implementar una manera de que varios grupos de profesionales trabajarán de manera simultánea. En base al código generado sale la opción de realizar un doble ciclo, para los box y los los profesionales, generando así todas la combinaciones posibles, pero no se logró llevar esta idea al programa. Con esta idea aumentaría la complejidad del algoritmo. Para próximas actividade s es necesario leer las diversas aplicaciones que tienen las estructuras al momento de programar, enfocado en el lenguaje c, para no tener mayor dudas sobre la sintaxis de este.

V. Manual de usuario

El programa no requiere archivos anexos para iniciar, al momento de iniciar el programa se solicita indicar la cantidad de profesionales de la salud que trabajan en el centro de salud, en donde lo que debe ingresar el usuario es un número entero positivo entre 1 y 20.

El programa generará tres archivos, uno llamado box.out el cual indicará los horarios de salida y entrada de los profesionales, junto con el rango horario de sanitización de cada box que existe . También se puede encontrar el archivo profesionales.out el cual indica los horarios en los que debe estar determinada persona en cada box. Por último existe un archivo llamado resumenesAtenciones.out el cual contiene dos números, el primero indica el número de profesionales con el cual se trabajo y el segundo el número de profesionales que se logró atender.