

Tarea 2 - Análisis de Algoritmos y Estructura de Datos

Jovanni Schneider Pizarro
Universidad de Santiago de Chile
1-2022

I. INTRODUCCIÓN

Se tiene el siguiente problema: Mintual exige a todas/os sus empleadas/os al menos dos horas semanales para tener cubierta la atención personalizada que no puede ser resuelta por los chatbots. Dichas horas no necesariamente son las mismas todas las semanas, por lo que es necesario hacer una planificación semana a semana. Dado que el servicio de atención al cliente vía chat es continuado, el encargado necesita identificar los intervalos de tiempo en los que no tendrá a algún/a empleado/a y así contratar a algunas personas para horarios de atención puntuales. Así, cada empleada/o de Mintual entrega un listado de intervalos de tiempo en los que tiene disponibilidad para atender el chat de la empresa. Para uniformizar esos intervalos disponibles, se debe especificar un horario de inicio y un horario de término (con posibilidad solo entre las horas 8 y 22, que es el horario de atención al cliente de Mintual, de lunes a viernes). Los horarios de inicio y de término deben anteponer el día de la semana con un dígito del 1 al 5 (1 corresponde al día lunes, 5 al día viernes). Por ejemplo:

[113, 115]

El intervalo indicaría que existe disponibilidad para atender el chat entre las 13 y las 15 horas del día lunes (1). Mientras que el intervalo

[409, 410]

indicaría que existe disponibilidad para atender el chat entre las 09 y las 10 horas del día jueves (4).

El objetivo de esta tarea es diseñar una estrategia para obtener los intervalos de tiempo en los que sería necesario contratar a personal adicional para cubrir el horario de atención de lunes a viernes de 8:00 a 22:00 hrs para garantizar que siempre habrá al menos un/a agente para responder algún requerimiento que el chatbot de Mintual no pueda manejar. Para la solución de este problema se usa el lenguaje de programación C

II. SOLUCION PROPUESTA

Se tiene que hay que comparar los intervalos dados por los empleados con los que están y ver si está utilizado o no. De esta manera se estandarizan los valores de los intervalos dados por los empleados como [a,b] y cada intervalo libre como [c,d], de esta

manera se tiene que comparar a con c y b con d, teniendo 3 posibles casos entre cada uno: <, = o >. Dado que tenemos 2 elementos comparándose a la vez nos quedan 3^2 casos posibles distintos, los cuales quedan dados por la [tabla 1](#).

Teniendo esto en cuenta solo nos queda saber que horarios hay disponibles, para esto se asume que ningún horario está tomado al momento de iniciar el programa, ya que semanalmente los horarios que cada empleado puede dedicar cambian. Para esto se genera una lista con todos los horarios disponibles de lunes a viernes utilizando el algoritmo [construirListaHorarios\(\)](#).

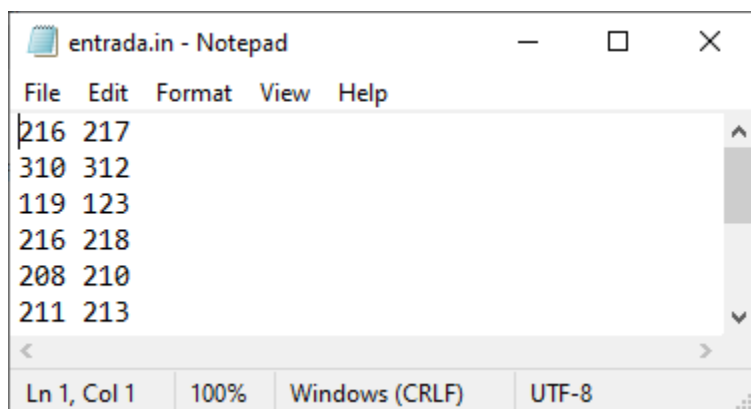
Teniendo esta lista construida ahora solo nos queda leer el archivo de

entrada y guardarlo en la lista enlazada, para esto no se adjunta pseudocódigo ya que depende totalmente del lenguaje en el que se trabaja. (se adjunta estructura del tda lista enlazada en el anexo).

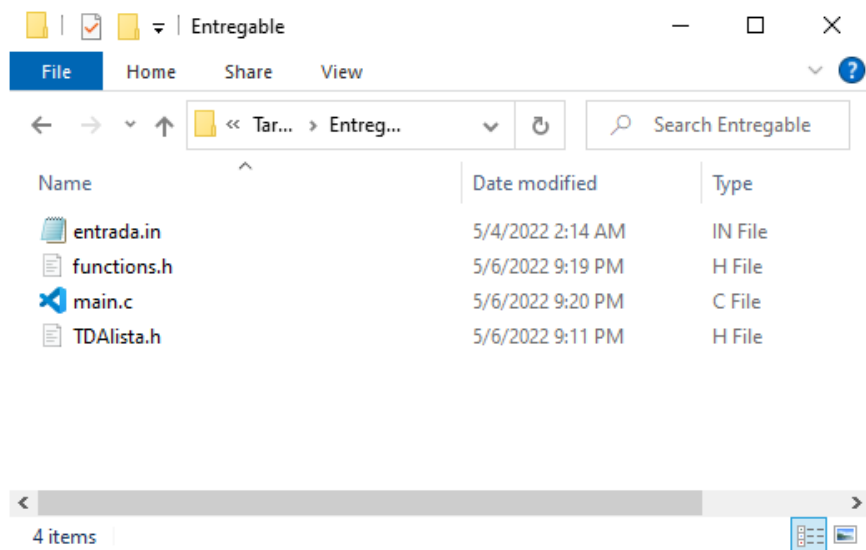
Ahora teniendo las 2 listas solo se debe ir comparando los intervalos de horarios y ver cual de los 9 casos diferentes se tienen y tomar las acciones correspondientes, esta tarea la realiza el algoritmo [calcularContratos\(\)](#) en conjunto con el algoritmo [separarRangos\(\)](#).

ANEXOS Y MANUAL DE USUARIO

Para la correcta ejecución del programa es necesario contener en el mismo directorio los archivos “main.c”, “functions.h” y “TDAlista.h”, además del archivo de entrada el cual su contenido debe seguir el formato:



De esta manera un directorio valido para la ejecución del programa sería el siguiente:

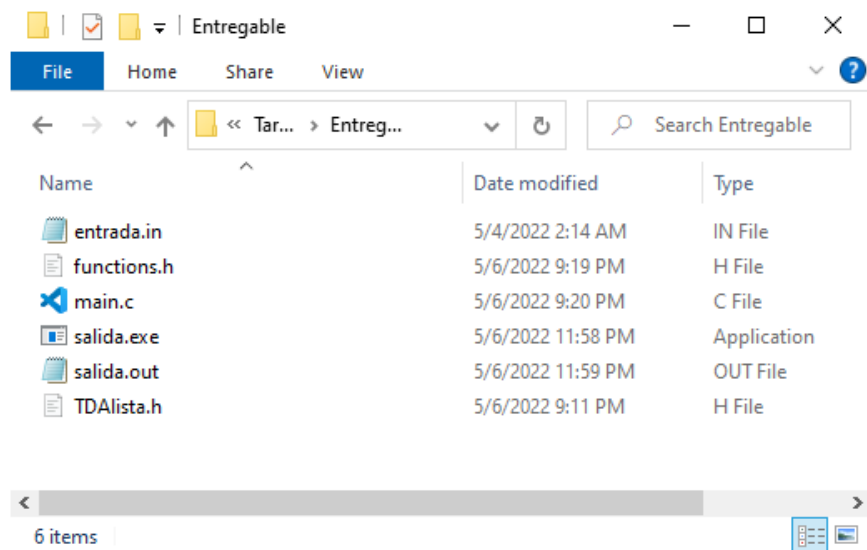


Se debe compilar el archivo main.c y ejecutar su resultante dentro del mismo directorio, el nombre del archivo de entrada se debe modificar en el mismo código en la siguiente sección:

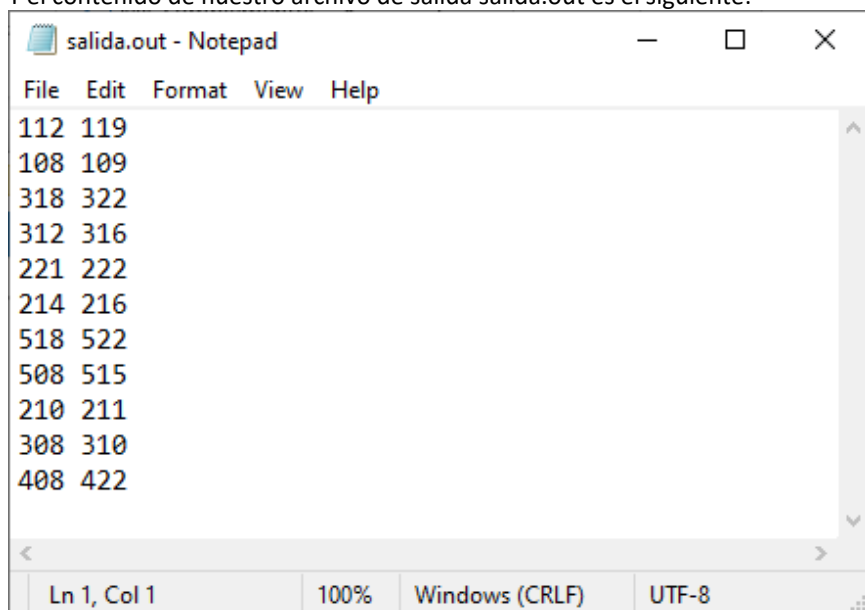
```
//Se guardan los datos del archivo c  
leerArchivo("entrada.in", lista);  
construirListaHorarios(horarios);
```

Si no existe un archivo de entrada el programa nos avisará de aquello.

Al ejecutar los comandos anteriores se tiene el siguiente resultado como directorio:



Y el contenido de nuestro archivo de salida salida.out es el siguiente:



```
salida.out - Notepad
File Edit Format View Help
112 119
108 109
318 322
312 316
221 222
214 216
518 522
508 515
210 211
308 310
408 422
Ln 1, Col 1 100% Windows (CRLF) UTF-8
```

Referencias:

Función construirListaHorarios ()

```
construirListaHorarios(TDALista lista):
    int rango <- inicializarArreglo() ...de tamaño 2
    rango[0] <- 508
    rango[1] <- 522
    for i <- 0 to 5:
        insertarInicio(lista,rango)
        rango[0] <- rango[0]-100
        rango[1] <- rango[1]-100
```

Función para construir la lista de horarios de O(c), ya que siempre es un for de 0 a 5

Función calcularContratos()

```
calcularContratos(TDAlista lista, TDAlista horarios):  
    nodo auxiliar <- lista->inicio  
    while auxiliar <> NULL do  
        separarRangos(auxiliar,horarios)  
        auxiliar <- auxiliar->siguiente
```

Función que recorre la lista de horarios de posibilidad de los empleados y ocupa ese horario en la lista de horarios disponibles. Función de $O(n1) * O(n2)$, donde $n1$ es el largo de "lista" y $n2$ el largo de "horarios".d

Función separarRangos()

```
separarRangos(nodo node, TDAlista lista):  
    nodo auxiliar <- lista->inicio  
    int rango <- inicializarArreglo() ...de tamaño 2  
    a <- node->dato[0]  
    b <- node->dato[1]  
    while auxiliar <> NULL do  
        caso <- 0  
        c <- auxiliar->dato[0]  
        d <- auxiliar->dato[1]  
        if a < c then  
            if c - a < 14 then  
                if b < d AND b > c then  
                    rango[0] <- b  
                    rango[1] <- d  
                    caso <- 1  
                else  
                    if b >= d AND b - d < 14 then  
                        caso <- 2
```

```
      caso <- 1
    else
      if a = c then
        if b < d then
          rango[0] <- b
          rango[1] <- d
          caso <- 1
        else
          if b >= d AND b - d < 14 then
            caso <- 2
      else
        if a > c then
          if b < d then
            caso <- 3
          else
            if b >= d AND b - d < 14 AND a < d then
              rango[0] <- c
              rango[1] <- a
              caso <- 1
```

```

if caso = 1 then
    insertarInicio(lista, rango)
    rango[0] <- c
    rango[1] <- d
    eliminarData(lista, rango)
else
    if caso = 2 then
        rango[0] <- c
        rango[1] <- d
        eliminarData(lista, rango)
    else
        if caso = 3 then
            rango[0] <- c
            rango[1] <- a
            insertarInicio(lista, rango)
            rango[0] <- b
            rango[1] <- d
            insertarInicio(lista, rango)
            rango[0] <- c
            rango[1] <- d
            eliminarData(lista, rango)
        auxiliar <- auxiliar -> siguiente

```

Tabla 1

Caso	$r(a,c)$	$r(b,d)$	$r(x,y)$	Accion	rango	Considerar
caso 1	$a < c$	$b < d$	$b > c$	cubrir	[b,d]	$c - a < 14$
caso 2	$a < c$	$b = d$		está cubierto		$c - a < 14$
caso 3	$a < c$	$b > d$		está cubierto		$c - a < 14 > b - d$
caso 4	$a = c$	$b < d$		cubrir	[b,d]	-
caso 5	$a = c$	$b = d$		está cubierto		-
caso 6	$a = c$	$b > d$		está cubierto		$b - d < 14$
caso 7	$a > c$	$b < d$		cubrir	[c,a] y [b,d]	-
caso 8	$a > c$	$b = d$		cubrir	[c,a]	-
caso 9	$a > c$	$b > d$	$a < d$	cubrir	[c,a]	$b - d < 14$