NOMBRE: Rafael Fernández Sánchez

PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN

IIC2133 — Estructura de Datos y Algoritmos — 1' 2020

Informe Tarea 0 – Complejidad Evento Ingreso

Para plantear la complejidad del **evento ingreso**, mostraremos un pseudocódigo de las operaciones escenciales realizadas en el evento. Los detalles que no son importantes se omiten.

La estructura usada para representar las colas es una **lista ligada**, donde además se almacena una referencia a los nodos que son los últimos de su tipo. Es decir, se almacenan referencias del último niño, adulto y robot, así como también el índice del nodo al que pertencen.

Plantearemos un pseudocódigo para el proceso de ingresar a **todas las personas** a sus respectivos terminales.

Algorithm 1 Evento Ingreso

```
1:  for t in Terminales do
2:    for p in EsperandoEntrar(t) do
3:      mejorPuerta ← Primera puerta de t
4:      for d in Puertas(t) do
5:        if SiguientePosicionEnCola(d, p) < mejorPuerta then
6:          mejorPuerta ← d
7:        end if
8:      end for
9:      Ingresar p a la cola de mejorPuerta, en la mejor posición correspondiente a su tipo.
10:   end for
11: end for
```

Calcularemos la complejidad para el peor caso:

- Sea T la cantidad total de terminales.
- Sea P el máximo de personas esperando entrar a algún terminal, de modo que es una cota superior.
- Sea D el máximo de puertas entre todos los terminales, de modo que es es una cota superior.
- La subrutina SiguientePosicionEnCola, debido a como definimos las colas, toma una cantidad constante c_1 de pasos.
- La operación de las líneas 5 y 6, toman una cantidad constante c_2 de pasos.
- La operación de la línea 9, debido a como definimos las colas, toma una cantidad constante c_3 de pasos.

Luego, la cantidad de pasos es para el Evento Ingreso es:

$$PasosEventoIngreso(T, P, D) = T \cdot P \cdot (D(c_1 + c_2) + c_3)$$

Luego, es fácil ver que si hacemos variar solo la cantidad de terminales y dejamos constantes las otras dos variables, obtenemos una complejidad lineal:

$$PasosEventoIngreso(T, k_1, k_2) = T \cdot k_1 \cdot (k_2(c_1 + c_2) + c_3) = T \cdot k_3$$

$$PasosEventoIngreso(T, k_1, k_2) \in \mathcal{O}(T)$$

$$PasosEventoIngreso(T, k_1, k_2) \in \mathcal{O}(n)$$

Procediendo de la misma manera, obtenermos que:

$$PasosEventoIngreso(k_1, P, k_2) \in \mathcal{O}(n) \text{ y } PasosEventoIngreso(k_1, k_2, D) \in \mathcal{O}(n)$$

Por lo tanto, el Evento Ingreso tiene una **complejidad lineal**, tanto en función de los terminales, las puertas de los terminales y las personas.

NOMBRE: Rafael Fernández Sánchez

Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

IIC2133 — Estructura de Datos y Algoritmos — 1' 2020

Informe Tarea 0 – Complejidad Evento Abordaje

El algoritmo para abordar un *Escape Pod* es bastante sencillo:

Algorithm 2 Abordaje(t, d)

```
for i in 1..8 do
    Quitar el primer elemento de la cola de puerta d en terminal t
end for
```

Podemos ver entonces que en cada llamada a la subrutina Abordaje, tenemos una cantidad constante de pasos, ya que quitar el primer elemento en una lista ligada no depende del tamaño de la lista.

Dicho esto, es claro que **una sola llamada a la subrutina tiene una complejidad $\mathcal{O}(1)$**.
A continuación llegaremos a una expresión que incluirá tambien terminales y puertas, representando la complejidad de que se aborden todos los *pods* que se puedan formar entre las filas de todos los terminales. Usaremos la misma notacion y supuestos de los primeros tres puntos del apartado anterior (Evento Ingreso).

$$PasosEventoAbordaje(T, P, D) = T \cdot \left\lfloor \frac{P}{8} \right\rfloor \cdot D$$

Siguiendo los mismos pasos del apartado anterior, llegamos a los siguientes resultados:

$$PasosEventoAbordaje(T, k_1, k_2) \in \mathcal{O}(n)$$

$$PasosEventoAbordaje(k_1, P, k_2) \in \mathcal{O}(n)$$

$$PasosEventoAbordaje(k_1, k_2, D) \in \mathcal{O}(n)$$

Por lo que la operacion de abordar todos los pasejoros de todos los terminales, tanto en función de los terminales, pasajeros y puertas, tiene una **complejidad lineal**.

NOMBRE: Rafael Fernández Sánchez

PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN

IIC2133 — Estructura de Datos y Algoritmos — 1' 2020

Informe Tarea 0 – Complejidad Evento Cierre

El proceso del cierre de una puerta d de un terminal t está dado por el siguiente pseudocódigo

Algorithm 3 Cierre(t, d)

```
1: puerta ← puerta de indice d en terminal t
2: Marcar puerta como cerrada.
3: for p in Cola(puerta) do
4:     Quitar persona p de la cola
5:     Ingresar persona p al terminal t
6: end for
```

- La línea 4 toma una cantidad constante c_1 de pasos.
- La línea 5, según lo visto en el Evento Ingreso, toma una cantidad $c_2 \cdot (D-1)$, donde $D-1$ representa la cantidad de puertas abiertas en el terminal d duespués del cierre de *puerta*.

Luego, la cantidad de pasos para el cierre de una puerta está dada por:

$$PasosCierre(P, D) = P \cdot (c_1 + c_2(D-1))$$

Nuevamente, y de manera análoga a los eventos anteriores, es fácil ver que:

$$PasosCierre(P, D) \in O(P \cdot D)$$

$$PasosCierre(P, k_1) \in \mathcal{O}(n) \text{ y } PasosCierre(k_1, D) \in \mathcal{O}(n)$$

Podemos notar que la cantidad de pasos en cerrar una puerta de un terminal no depende de la cantidad total de terminales.

NOMBRE: Rafael Fernández Sánchez

Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

IIC2133 — Estructura de Datos y Algoritmos — 1' 2020

Informe Tarea 0 – Complejidad Evento Clausura

El proceso del cierre de una puerta d de un terminal t está dado por el siguiente pseudocódigo

Algorithm 4 Clausura(t1, t2)

```
1:  cola ← nueva cola vacía
2:  while PersonasEnTerminal(t) > 0 do
3:    for d in puertas(t1) do
4:      if d está abierta and hay personas en cola de d then
5:        Quitar primer pasajero de cola en d y asignarlo en p
6:        Insertar p en cola
7:      end if
8:    end for
9:  end while
10: marcar t1 como cerrado
11: for p2 in cola do
12:   Ingresar p2 en t2
13: end for
```

- Sea P la cantidad máxima de personas entre las colas de $t1$, a modo de cota superior.
- Sea D el máximo de puertas abiertas entre $t1$ y $t2$ al llamar a la subrutina $Clausura$.
- Sea c_1 la cantidad de pasos constante que toma la línea 5, según lo visto en el evento Ingresar.
- Sea c_2 la cantidad de pasos constante que toma la línea 6.
- Sea $D \cdot c_3$ la cantidad de pasos que tomaría, en el peor caso, la línea 12.

Luego, podemos expresar la cota de la cantidad de pasos de la siguiente forma:

$$PasosClausura(P, D) = P \cdot D \cdot (c_1 + c_2) + P \cdot D \cdot D \cdot c_3$$

$$PasosClausura(P, D) \in \mathcal{O}(P \cdot D^2)$$

$$PasosClausura(P, k_1) \in \mathcal{O}(n)$$

$$PasosClausura(k_1, D) \in \mathcal{O}(n^2)$$

Notar que por la notación elegida, al agregar puertas también estamos agregando personas. Si al agregar una nueva puerta repartimos las personas de manera que que todas las puertas tengan $\frac{P}{D}$ personas, nuestra complejidad sería $O(P \cdot D)$

NOMBRE: Rafael Fernández Sánchez

PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN

IIC2133 — Estructura de Datos y Algoritmos — 1' 2020

Informe Tarea 0 – Complejidad Evento Láser

Algorithm 5 Laser(t, d, i)

1: *puerta* $\leftarrow$ puerta d de terminal t
2: *nodoActual* $\leftarrow$ primer nodo en cola de *puerta*
3: **while** $indice(nodoActual) \neq i$ **do**
4:     $nodoActual \leftarrow nodoActual.siguiente$
5: **end while**
6: Quitar *nodoActual* de la cola

- Sea P la cantidad de personas en la cola de la puerta d del terminal t.
- Sea c_1 la cantidad de pasos constante al ejecutar las líneas 1, 2 y 6.
- Sea c_2 la cantidad de pasos constante que toma la línea 4.

Luego, la cantidad de pasos que tomaría la subrutina en el peor caso está dado por:

$$PasosLaser(P) = c_1 + c_2 \cdot P$$

Por lo que podemos concluir que:

$$PasosLaser(P) \in \mathcal{O}(p)$$

$$PasosLaser(P) \in \mathcal{O}(n)$$