Algoritmos y Estructuras de Datos II

Trabajo Práctico 1

Departamento de Computación
Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

Integrante	LU	Correo electrónico
Guberman, Diego Andrés	469/17	diego98g@hotmail.com
Ramis Folberg, Ezequiel Leonel	881/21	ezequielramis.hello@gmail.com
Sabetay, Kevin Damian	476/16	kevin.sabetay96@gmail.com

Reservado para la cátedra

Instancia	Docente	Nota
Primera entrega		
Segunda entrega		

Índice

1. Preámbulo

Antes de presentar los módulos, definimos las siguientes variables para las complejidades temporales:

- N — tamaño del tablero.
- K — cantidad de jugadores.
- $|\Sigma|$ — cantidad de letras en el alfabeto.
- F — cantidad de fichas por jugador.
- $L_{\texttt{máx}}$ — longitud de la palabra legítima más larga definida por la variante del juego de la que se trate.

2. Módulo Juego

2.1. Interfaz

se explica con: JUEGO

géneros: `juego`

usa: BOOL, NAT, COLA, LETRA, OCURRENCIA, VARIANTE

operaciones:

NUEVOJUEGO(**in** k : `nat`, **in** v : `variante`, **in** r : `cola(letra)`) $\rightarrow$ res : `juego`
Pre $\equiv$ $\{tamaño(r) \geq tamañoTablero(v) * tamañoTablero(v) + k * \#fichas(v) \wedge k > 0\}$
Post $\equiv$ $\{res =_{\text{obs}} nuevoJuego(k, v, r)\}$
Complejidad: $O(N^2 + |\Sigma|K + FK)$
Descripción: Dada una cantidad de jugadores, una variante de juego y un repositorio de fichas, se inicia un nuevo juego con el tablero vacío.
Aliasing: ??

JUGADAVÁLIDA?(**in** j : `juego`, **in** o : `occurrencia`) $\rightarrow$ res : `bool`
Pre $\equiv$ {true}
Post $\equiv$ $\{res =_{\text{obs}} jugadaVálida?(j, o)\}$
Complejidad: $O(L^2_{\texttt{máx}})$
Descripción: Determina si una jugada es válida.
Aliasing: ??

UBICAR(**in/out** j : `juego`, **in** o : `occurrencia`)
Pre $\equiv$ $\{jugadaVálida(j, o) \wedge j =_{\text{obs}} J_0\}$
Post $\equiv$ $\{j =_{\text{obs}} ubicar(J_0, o)\}$
Complejidad: $O(m)$, donde m es el número de fichas que se ubican.
Descripción: Ubica un conjunto de fichas en el tablero.
Aliasing: ??

VARIANTE(**in** j : `juego`) $\rightarrow$ res : `variante`
Pre $\equiv$ {true}
Post $\equiv$ $\{res =_{\text{obs}} variante(j)\}$
Complejidad: $O(1)$
Descripción: Obtiene información sobre la variante del juego.
Aliasing: ??

TURNO(**in** j : `juego`) $\rightarrow$ res : `nat`
Pre $\equiv$ {true}
Post $\equiv$ $\{res =_{\text{obs}} turno(j)\}$

Complejidad: $O(1)$
Descripción: Obtiene al jugador del turno actual.
Aliasing: ??

PUNTAJE(**in** j : `juego`, **in** i : `nat`) $\rightarrow$ res : `nat`
Pre $\equiv \{i < \#jugadores(j)\}$
Post $\equiv \{res =_{\text{obs}} puntaje(j, i)\}$
Complejidad: $O(1 + m \cdot L_{\texttt{máx}})$, donde m es la cantidad de fichas que ubicó el jugador desde la última vez que se invocó a esta operación.
Descripción: Obtiene el puntaje de un jugador.
Aliasing: ??

ENTABLERO?(**in** J : `juego`, **in** i : `nat`, **in** j : `nat`) $\rightarrow$ res : `bool`
Pre $\equiv \{\text{true}\}$
Post $\equiv \{res =_{\text{obs}} enTablero?(tablero(J), i, j)\}$
Complejidad: $O(1)$
Descripción: Determina si una coordenada (i, j) está en el rango del tablero.
Aliasing: ??

HAYLETRA?(**in** J : `juego`, **in** i : `nat`, **in** j : `nat`) $\rightarrow$ res : `bool`
Pre $\equiv \{enTablero?(tablero(J), i, j)\}$
Post $\equiv \{res =_{\text{obs}} hayLetra?(tablero(J), i, j)\}$
Complejidad: $O(1)$
Descripción: Determina si una celda del tablero dada una coordenada (i, j) está ocupada por una letra.
Aliasing: ??

LETRA(**in** J : `juego`, **in** i : `nat`, **in** j : `nat`) $\rightarrow$ res : `letra`
Pre $\equiv \{enTablero?(tablero(J), i, j) \wedge_{\text{L}} hayLetra?(tablero(J), i, j)\}$
Post $\equiv \{res =_{\text{obs}} letra(tablero(J), i, j)\}$
Complejidad: $O(1)$
Descripción: Obtiene el contenido de una celda del tablero dada una coordenada (i, j).
Aliasing: ??

#LETRATIENEJUGADOR(**in** j : `juego`, **in** x : `letra`, **in** i : `nat`) $\rightarrow$ res : `nat`
Pre $\equiv \{i < \#jugadores(j)\}$
Post $\equiv \{res =_{\text{obs}} \#(x, fichas(j, i))\}$
Complejidad: $O(1)$
Descripción: Dada una cierta letra x del alfabeto, conocer cuántas fichas tiene un jugador de dicha letra.
Aliasing: ??

2.2. Implementación

Representación

`juego` **se representa con** `juego_estr`

donde `juego_estr` es `tupla`(

tablero: `array_dimensionable(array_dimensionable(puntero(letra)))`
, *jugadores*: `array_dimensionable(tupla(`

puntaje: `nat`
, *cantFichasPorLetra*: `array_dimensionable(nat)`

, *cantFichasTotal*: `nat`

))

, *jugadorActual*: `nat`

, *repositorio*: `cola(letra)`

, *variante*: `variante`

)

Invariante de Representación

Rep : juego_estr $\longrightarrow$ bool

$$
\begin{aligned}
\text{Rep}(e) \equiv\ & (\forall i : \texttt{nat})(i < tam(e.jugadores) \Rightarrow (\\
& \quad tam(e.jugadores[i].cantFichasPorLetra) = \text{FICHASPORJUGADOR}(e.variante) \wedge \\
& \quad tam(e.jugadores[i].cantFichasPorLetra) = \text{DOM}() \wedge_{\text{L}} \\
& \quad \textstyle\sum_{f<\text{DOM}()} f = e.jugadores[i].cantFichasTotal \\
&)) \wedge \\
& tam(e.tablero) = \text{TAMAÑOTABLERO}(e.variante) \wedge \\
& e.jugadorActual \leq tam(e.jugadores) \wedge \\
& (\forall i : \texttt{nat})(i < tam(e.tablero) \Rightarrow_{\text{L}} tam(e.tablero[i]) = tam(e.tablero)) \wedge_{\text{L}} \\
& \textstyle\sum_{i<tam(e.jugadores)} e.jugadores[i].puntaje \geq \\
& \textstyle\sum_{i,j<tam(e.tablero)} \textbf{if } e.tablero[i][j] \neq \text{NULL} \\
& \qquad \textbf{then } \text{PUNTAJELETRA}(e.variante, *e.tablero[i][j]) \\
& \qquad \textbf{else } 0 \\
& \quad \textbf{fi}
\end{aligned}
$$

Función de Abstracción

Abs : juego_estr e $\longrightarrow$ juego $\{\text{Rep}(e)\}$

$$
\begin{aligned}
\text{Abs}(e) =_{\text{obs}} \text{J: juego} \mid\ & e.variante =_{\text{obs}} variante(J) \wedge \\
& e.repositorio =_{\text{obs}} repositorio(J) \wedge \\
& e.jugadorActual =_{\text{obs}} turno(J) \wedge \\
& (tam(e.jugadores) =_{\text{obs}} \#jugadores(J) \wedge_{\text{L}} \\
& (\forall i : \texttt{nat})(i < tam(e.jugadores) \Rightarrow_{\text{L}} (\\
& \quad \text{PUNTAJE}(e.jugadores[i]) =_{\text{obs}} puntaje(J, i))) \wedge \\
& \quad (\forall l : \texttt{letra})(e.cantFichasPorLetra[\text{ORD}(l)] = \#(l, fichas(J, i))) \\
&)) \wedge \\
& tam(e.tablero) =_{\text{obs}} tamaño(tablero(J)) \wedge_{\text{L}} \\
& (\forall i, j : \texttt{nat})(enTablero?(tablero(J), i, j) \Rightarrow_{\text{L}} (\\
& \quad hayLetra?(tablero(J), i, j) \iff e.tablero[i][j] \neq \text{NULL} \wedge \\
& \quad hayLetra?(tablero(J), i, j) \Rightarrow_{\text{L}} letra(tablero(J), i, j) =_{\text{obs}} *e.tablero[i][j] \\
&))
\end{aligned}
$$

Algoritmos

HACERGUIA(**in** A : `guia`, **in** *parámetroInútil* : `Nat`) $\longrightarrow$ `bool`

```
1:  i ← 0                                                          ▷ esto es Θ(1)
2:  n ← guia.cantEjercicios()                                      ▷ O(1)
3:  consultas ← DICCVACIO
4:  PREPARARMATE()                                                 ▷ Ω(n^n)
5:  while i < n do
6:      PENSAREJERCICIO(I)
7:      if TENGOCONSULTAS(i) then
8:          ESCRIBIRCONSULTASEJERCICIO(i,consultas)
9:      else
10:         COMERBIZOCHITO()
11:     COMERBIZOCHITO()
12: for miVariable do
13:     hacer algo
14: return VACIO?(consultas)
```

2.3. Servicios usados

3. Módulo Servidor

3.1. Interfaz

se explica con: SERVIDOR

géneros: `servidor`

usa: NAT, JUEGO, OCURRENCIA, VARIANTE

operaciones:

NUEVOSERVIDOR(**in** k : `nat`, **in** v : `variante`) $\rightarrow$ res : `servidor`
Pre $\equiv$ {true}
Post $\equiv$ {$\exists r$: `cola(letra)` | $res =_{\text{obs}} nuevoServidor(k, v, r)$}
Complejidad: $O(N^2 + |\Sigma|K + FK)$
Descripción: Dada una cantidad de jugadores y una variante de juego, se inicia un nuevo servidor y una nueva partida de juego.
Aliasing: ??

CONECTAR(**in/out** s : `servidor`)
Pre $\equiv$ {$\neg empezó?(s) \land s =_{\text{obs}} S_0$}
Post $\equiv$ {$s =_{\text{obs}} conectarCliente(S_0)$}
Complejidad: $O(1)$
Descripción: Conecta un cliente a un servidor.
Aliasing: ??

CONSULTAR(**in/out** s : `servidor`, **in** cid : `nat`)
Pre $\equiv$ {$cid \leq \#conectados(s) \land s =_{\text{obs}} S_0$}
Post $\equiv$ {$s =_{\text{obs}} consultar(S_0, cid)$}
Complejidad: $O(n)$, donde n es la cantidad de mensajes en la cola de dicho cliente.
Descripción: Consulta la cola de notificaciones de un cliente (lo cual vacía dicha cola).
Aliasing: ??

RECIBIR(**in/out** s : `servidor`, **in** cid : `nat`, **in** o : `ocurrencia`)
Pre $\equiv$ {$cid \leq \#conectados(s) \land s =_{\text{obs}} S_0$}
Post $\equiv$ {$s =_{\text{obs}} recibirMensaje(S_0, cid, o)$}
Complejidad: ??
Descripción: Recibe un mensaje de un cliente.
Aliasing: ??

CLIENTESESPERADOS(**in** s : `servidor`) $\rightarrow$ res : `nat`
Pre $\equiv$ {true}
Post $\equiv$ {$res =_{\text{obs}} \#esperados(s)$}
Complejidad: $O(1)$
Descripción: Obtiene el número de clientes esperados.
Aliasing: ??

CLIENTESCONECTADOS(**in** s : `servidor`) $\rightarrow$ res : `nat`
Pre $\equiv$ {true}
Post $\equiv$ {$res =_{\text{obs}} \#conectados(s)$}
Complejidad: $O(1)$
Descripción: Obtiene el número de clientes conectados.
Aliasing: ??

PARTIDA(**in** s : `servidor`) $\rightarrow$ res : `juego`
Pre $\equiv$ {true}
Post $\equiv$ {$res =_{\text{obs}} juego(s)$}
Complejidad: $O(1)$
Descripción: Obtiene el juego que se está jugando en el servidor.
Aliasing: ??

3.2. Implementación

Representación

`servidor` **se representa con** `servidor_estr`

donde `servidor_estr` es `tupla`(

juego: `juego`
, *jugadoresConectados*: `nat`
, *jugadoresEsperados*: `nat`
, *notificaciones*: `array_dimensionable(cola(notif))`

)

Invariante de Representación

Función de Abstracción

Algoritmos

3.3. Servicios usados

4. Módulos auxiliares

4.1. Módulo Letra

Se asume una implementación acorde[1] al módulo de género `letra` con las siguientes operaciones en la interfaz (todas con órden de complejidad $O(1)$):

- DOM : $\rightarrow$ `nat` — Tamaño del dominio del tipo `letra`. Corresponde con la variable A de su especificación.
- ORD : `letra` $\rightarrow$ `nat` — Dada una letra, devuelve su correspondiente índice.
- ORD^{-1} : `nat` $n \rightarrow$ `letra` $\{n < A\}$ — Dado un índice, devuelve su correspondiente letra.

4.2. Módulo Variante

4.2.1. Interfaz

se explica con: VARIANTE

géneros: `variante`

usa: ??

operaciones:

NUEVAVARIANTE(

in n : `nat`,
in f : `nat`,
in *puntajes* : `dicc`[2]`(letra, nat)`,
in *legítimas* : `conj`[3]`(secu(letra))`

) $\rightarrow$ res : `variante`
Pre $\equiv \{n > 0 \land f > 0\}$
Post $\equiv \{res =_{\text{obs}} nuevaVariante(n, f, puntajes, legítimas)\}$
Complejidad: $O(\#legítimas \cdot L_{\texttt{máx}})$
Descripción: Genera una variante de juego.
Aliasing: ??

TAMAÑOTABLERO(**in** v : `variante`) $\rightarrow$ res : `nat`
Pre $\equiv \{\text{true}\}$
Post $\equiv \{res =_{\text{obs}} tamañoTablero(v)\}$
Complejidad: $O(1)$
Descripción: Devuelve el tamaño del tablero.
Aliasing: ??

FICHASPORJUGADOR(**in** v : `variante`) $\rightarrow$ res : `nat`
Pre $\equiv \{\text{true}\}$
Post $\equiv \{res =_{\text{obs}} \#fichas(v)\}$
Complejidad: $O(1)$
Descripción: Devuelve la cantidad de fichas que debe de tener cada jugador.
Aliasing: ??

PUNTAJELETRA(**in** v : `variante`, **in** l : `letra`) $\rightarrow$ res : `nat`
Pre $\equiv \{\text{true}\}$
Post $\equiv \{res =_{\text{obs}} puntajeLetra(v, l)\}$
Complejidad: $O(1)$

[1] Una buena opción es usar un Enumerado.
[2] No se refiere al género del módulo Diccionario Lineal debido a que se necesita que las operaciones de búsqueda sean $O(1)$.
[3] No se refiere al género del módulo Conjunto Lineal debido a que se necesita que las operaciones de búsqueda sean $O(L_{\texttt{máx}})$.

Descripción: Devuelve el puntaje de una letra.
Aliasing: ??

PALABRALEGÍTIMA?(**in** v : `variante`, **in** l : `secu(letra)`) $\rightarrow$ res : `bool`
Pre $\equiv$ {true}
Post $\equiv$ $\{res =_{\text{obs}} palabraLegítima(v, l)\}$
Complejidad: $O(L_{\text{máx}})$
Descripción: Determina si una palabra es legítima dentro de la variante de juego.
Aliasing: ??

LONGPALABRAMÁSLARGA(**in** v : `variante`) $\rightarrow$ res : `nat`
Pre $\equiv$ {true}
Post $\equiv$ {

$$(\exists p : \texttt{secu(letra)})(res =_{\text{obs}} long(p) \wedge palabraLegítima?(v, p) \wedge (\forall p_2 : \texttt{secu(letra)})(palabraLegítima?(v, p_2) \Rightarrow res \geq long(p_2)))$$

}
Complejidad: $O(1)$
Descripción: Obtiene la longitud de la palabra legítima más larga de la variante.
Aliasing: ??

4.2.2. Implementación

4.3. Módulo Ocurrencia

4.3.1. Interfaz

se explica con: OCURRENCIA

géneros: `ocurrencia`

usa: ??

operaciones:

FORMAPALABRA?(**in** o : `ocurrencia`) $\rightarrow$ res : `bool`
Pre $\equiv$ {true}
Post $\equiv$ $\{res =_{\text{obs}} ocurrenciaFormaPalabra?(o)\}$
Complejidad: $O(L^2_{\text{máx}})$
Descripción: Determina si una ocurrencia forma una palabra.
Aliasing: ??

PALABRAQUEFORMA(**in** o : `ocurrencia`) $\rightarrow$ res : `secu(letra)`
Pre $\equiv$ $\{ocurrenciaFormaPalabra?(o)\}$
Post $\equiv$ $\{res =_{\text{obs}} palabraQueFormaLaOcurrencia(o)\}$
Complejidad: $O(L_{\text{máx}})$
Descripción: Obtiene la palabra formada por una ocurrencia.
Aliasing: ??

4.3.2. Implementación

4.4. Módulo Notificación

Se asume una implementación acorde[4] al módulo TIPONOTIFICACIÓN de género `tipoNotif` para usarse en este módulo.

[4]Una buena opción es usar un Enumerado.

4.4.1. Interfaz

se explica con: Notificación

géneros: `notif`

usa: ??

operaciones:

IdCliente(**in** cid : `nat`) $\rightarrow$ res : `notif`
Pre $\equiv$ {true}
Post $\equiv$ {$res =_{\text{obs}}$ IdCliente(cid)}
Complejidad: $O(1)$
Descripción: Genera una notificación de tipo IdCliente.
Aliasing: ??

Empezar(**in** n : `nat`) $\rightarrow$ res : `notif`
Pre $\equiv$ {true}
Post $\equiv$ {$res =_{\text{obs}}$ Empezar(n)}
Complejidad: $O(1)$
Descripción: Genera una notificación de tipo Empezar.
Aliasing: ??

TurnoDe(**in** cid : `nat`) $\rightarrow$ res : `notif`
Pre $\equiv$ {true}
Post $\equiv$ {$res =_{\text{obs}}$ TurnoDe(cid)}
Complejidad: $O(1)$
Descripción: Genera una notificación de tipo TurnoDe.
Aliasing: ??

Ubicar(**in** cid : `nat`, **in** o : `ocurrencia`) $\rightarrow$ res : `notif`
Pre $\equiv$ {true}
Post $\equiv$ {$res =_{\text{obs}}$ Ubicar(cid, o)}
Complejidad: $O(1)$
Descripción: Genera una notificación de tipo Ubicar.
Aliasing: ??

4.4.2. Implementación