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 : `pila(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: tab
, *jugadores*: array_dimensionable(jugador)
, *tiempo*: nat
, *repositorio*: pila(letra)
, *variante*: variante

)

y jugador es tupla(

puntaje: nat
, *historial*: lista(tupla(*ocurrencia*: ocurrencia, *tiempo*: nat))
, *jugadasSinCalcularPuntaje*: nat
, *cantFichasPorLetra*: array_dimensionable(nat)
, *cantFichasTotal*: nat

)

Invariante de Representación

Rep : juego_estr $\longrightarrow$ bool

$$
\begin{aligned}
\text{Rep}(e) \equiv\ & tamaño(e.tablero) = tamañoTablero(e.variante) \wedge \\
& (\forall i : \texttt{nat})(i < tam(e.jugadores) \Rightarrow_{\text{L}} (\\
& \quad tam(e.jugadores[i].cantFichasPorLetra) = \#fichas(e.variante) \wedge \\
& \quad tam(e.jugadores[i].cantFichasPorLetra) = \text{DOM}() \wedge_{\text{L}} \\
& \quad \textstyle\sum_{f<\text{DOM}()} e.jugadores[i].cantFichasPorLetra[f] = e.jugadores[i].cantFichasTotal \wedge \\
& \quad e.jugadores[i].jugadasSinCalcularPuntaje \leq tam(e.jugadores[i].historial) \wedge_{\text{L}} \\
& \quad tam(e.jugadores[i].historial) = \lceil e.tiempo/tam(e.jugadores) \rceil \wedge \\
& \quad (\forall h : \texttt{nat})(h < long(e.jugadores[i].historial) \Rightarrow_{\text{L}} \\
& \quad\quad e.jugadores[i].historial[h].tiempo = h * tam(e.jugadores) + i \wedge \\
& \quad\quad ocurrenciaFormaPalabra?(e.jugadores[i].historial[h].ocurrencia) \\
& \quad) \\
&)) \wedge_{\text{L}} \\
& ocurrenciasVálidas?(nuevoTablero(tamaño(e.tablero)), historiales(e.jugadores, 0)) \wedge_{\text{L}} \\
& e.tablero =_{\text{obs}} ponerOcurrencias(nuevoTablero(tamaño(e.tablero)), historiales(e)) \wedge_{\text{L}} \\
& (\forall i : \texttt{nat})(i < tam(e.jugadores) \Rightarrow_{\text{L}} \\
& \quad e.jugadores[i].puntaje = \textstyle\sum_{k<tam(e.jugadores[i].historial)-e.jugadores[i].jugadasSinCalcularPuntaje} \\
& \qquad\qquad\qquad\qquad puntajeOcurrencia(e, i, k) \\
&)
\end{aligned}
$$

donde

$historiales$: juego_estr $\longrightarrow$ multiconj(tupla(ocurrencia,nat))
$historiales(e') \equiv historialesDesdeHastaTiempo(e'.jugadores, 0, 0, e.tiempo)$

$historialesDesdeHastaTiempo$: ad(jugador) $\times$ nat $\times$ nat $\times$ nat
$\longrightarrow$ multiconj(tupla(ocurrencia,nat))
$historialesDesdeHastaTiempo(js, k, i, j) \equiv$

if $k \geq tam(js)$

then $\emptyset$
else $historialDesdeHastaTiempo(js[k].historial, i, j) \cup$
$historialesDesdeHastaTiempo(js, k + 1, i, j)$

fi

$historialDesdeHastaTiempo$: `lista(tupla(ocurrencia,nat))` × `nat` × `nat`
⟶ `multiconj(tupla(ocurrencia,nat))`
$historialDesdeHastaTiempo(ls, i, j) \equiv$

if $vacía?(ls)$

then $\emptyset$
else $historialDesdeHastaTiempo(fin(ls), i, j) \cup$

if $i \leq \pi_2(prim(ls)) < j$

then $prim(ls)$
else $\emptyset$

fi

fi

$ocurrenciasVálidas?$: `tab` × `multiconj(tupla(ocurrencia,nat))` ⟶ `bool`
$ocurrenciasVálidas?(t, os) \equiv$

if $vacía?(os)$

then $true$
else $celdasLibres?(t, dameUno(\pi_1(os))) \wedge_{\text{L}}$
$ocurrenciasVálidas?(ponerLetras(t, dameUno(\pi_1(os))), sinUno(os))$

fi

$ponerOcurrencias$: `tab` × `multiconj(tupla(ocurrencia,nat))` ⟶ `tab`
$ponerOcurrencias(t, os) \equiv$

if $vacía?(os)$

then t
else $ponerOcurrencias(ponerLetras(t, dameUno(\pi_1(os))), sinUno(os))$

fi

$puntajeOcurrencia(e', i', k') \equiv \sum \ \sum_{i<tam(e.jugadores)} e.jugadores[i].puntaje = \sum_{t<e.tiempo}$

if $e'.tablero[i'][j'] \neq$ NULL

then PUNTAJELETRA($e'.variante, *e'.tablero[i'][j'].letra$)
else 0

fi

Función de Abstracción

Abs : juego_estr e ⟶ juego {Rep(e)}

Abs(e) =$_{\text{obs}}$ J: juego | $e.variante =_{\text{obs}} variante(J) \wedge$
$e.repositorio =_{\text{obs}} repositorio(J) \wedge$
$e.jugadorActual =_{\text{obs}} turno(J) \wedge$
$(tam(e.jugadores) =_{\text{obs}} \#jugadores(J) \wedge_{\text{L}}$
$(\forall i : \texttt{nat})(i < tam(e.jugadores) \Rightarrow_{\text{L}} ($
PUNTAJE($e.jugadores[i]$) =$_{\text{obs}}$ $puntaje(J, i))) \wedge$
$(\forall l : \texttt{letra})(e.cantFichasPorLetra[\text{ORD}(l)] = \#(l, fichas(J, i)))$

$$
\begin{array}{l}
)) \wedge \\
tam(e.tablero) =_{\text{obs}} tamaño(tablero(J)) \wedge_{\text{L}} \\
(\forall i, j : \texttt{nat})(enTablero?(tablero(J), i, j) \Rightarrow_{\text{L}} (\\
\quad hayLetra?(tablero(J), i, j) \iff e.tablero[i][j] \neq \text{NULL} \wedge \\
\quad hayLetra?(tablero(J), i, j) \Rightarrow_{\text{L}} letra(tablero(J), i, j) =_{\text{obs}} *e.tablero[i][j] \\
)) \\
\end{array}
$$

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 (Trie)

4.2.1. Interfaz

se explica con: VARIANTE

géneros: `variante`

usa: ??

operaciones:

NUEVAVARIANTE(

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

) $\rightarrow$ *res* : `variante`
Pre $\equiv \{n > 0 \land f > 0\}$
Post $\equiv \{res =_{\text{obs}} nuevaVariante(n, f, puntajes, legítimas)\}$
Complejidad: $O(\#legítimas \cdot L_{\text{máx}})$
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: $O(1)$
Descripción: Devuelve el puntaje de una letra.

[1] Una buena opción es usar un Enumerado.

Aliasing: ??

PALABRALEGÍTIMA?(**in** v : `variante`, **in** l : `secu(letra)`) $\rightarrow$ res : `bool`
Pre $\equiv$ {true}
Post $\equiv$ $\{res =_{\text{obs}} palabraLegítima(v, l)\}$
Complejidad: $O(L_{\texttt{máx}})$
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 Tablero

4.3.1. Interfaz

se explica con: TABLERO

géneros: `tab`

usa: ??

operaciones:

NUEVOTABLERO(**in** n : `nat`) $\rightarrow$ res : `tab`
Pre $\equiv$ $\{n > 0\}$
Post $\equiv$ $\{res =_{\text{obs}} nuevoTablero(n)\}$
Complejidad: $O(N^2)$
Descripción: Genera un tablero de tamaño n.
Aliasing: ??

PONERLETRA(**in** t : `tab`, **in** i : `nat`, **in** j : `nat`, **in** l : `letra`, **in** tm : `nat`) $\rightarrow$ res : `tab`
Pre $\equiv$ $\{enTablero(t, i, j) \wedge_{\text{L}} \neg hayLetra?(t, i, j)\}$
Post $\equiv$ $\{res =_{\text{obs}} nuevoTablero(n)\}$
Complejidad: $O(N^2)$
Descripción: Genera un tablero de tamaño n.

[??]

TAMAÑOTABLERO(**in** v : `variante`) $\rightarrow$ res : `nat`
Pre $\equiv$ {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 $\equiv$ {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: $O(1)$
Descripción: Devuelve el puntaje de una letra.
Aliasing: ??

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

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

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

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

4.3.2. Implementación

4.4. Módulo Ocurrencia

es renombre de conj(tupla(nat,nat,letra)) es fichas del jugador a mano, no incluimos las que están alineadas en el tablero

4.5. Módulo Notificación

asumimos que existe el tipo notif