

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	3
2. Módulo Juego	3
2.1. Interfaz	3
2.2. Implementación	5
2.3. Servicios usados	8
3. Módulo Servidor	9
3.1. Interfaz	9
3.2. Implementación	10
3.3. Servicios usados	10
4. Módulos auxiliares	11
4.1. Módulo Letra	11
4.2. Módulo Variante (Trie)	11
4.2.1. Interfaz	11
4.2.2. Implementación	12
4.3. Módulo Tablero	12
4.3.1. Interfaz	12
4.3.2. Implementación	13
4.4. Módulo Ocurrencia	13
4.5. Módulo Notificación	13

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.
- $L_{\text{máx}}$ — 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 $\equiv \{ \text{tamaño}(r) \geq \text{tamañoTablero}(v) * \text{tamañoTablero}(v) + k * \#fichas(v) \wedge k > 0 \}$

Post $\equiv \{ res =_{\text{obs}} \text{nuevoJuego}(k, v, r) \}$

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 : ocurrencia) $\rightarrow res$: bool

Pre $\equiv \{ \text{true} \}$

Post $\equiv \{ res =_{\text{obs}} \text{jugadaVálida?}(j, o) \}$

Complejidad: $O(L_{\text{máx}}^2)$

Descripción: Determina si una jugada es válida.

Aliasing: ??

UBICAR(in/out j : juego, in o : ocurrencia)

Pre $\equiv \{ \text{jugadaVálida}(j, o) \wedge j =_{\text{obs}} J_0 \}$

Post $\equiv \{ j =_{\text{obs}} \text{ubicar}(J_0, o) \}$

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 $\equiv \{ \text{true} \}$

Post $\equiv \{ res =_{\text{obs}} \text{variante}(j) \}$

Complejidad: $O(1)$

Descripción: Obtiene información sobre la variante del juego.

Aliasing: ??

TURNNO(in j : juego) $\rightarrow res$: nat

Pre $\equiv \{ \text{true} \}$

Post $\equiv \{ res =_{\text{obs}} \text{turnno}(j) \}$

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 $\equiv \{res =_{\text{obs}} \text{puntaje}(j, i)\}$

Complejidad: $O(1 + m \cdot L_{\text{máx}})$, 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 $\equiv \{res =_{\text{obs}} \text{enTablero?}(\text{tablero}(J), i, j)\}$

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 $\equiv \{\text{enTablero?}(\text{tablero}(J), i, j)\}$

Post $\equiv \{res =_{\text{obs}} \text{hayLetra?}(\text{tablero}(J), i, j)\}$

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 $\equiv \{\text{enTablero?}(\text{tablero}(J), i, j) \wedge_L \text{hayLetra?}(\text{tablero}(J), i, j)\}$

Post $\equiv \{res =_{\text{obs}} \text{letra}(\text{tablero}(J), i, j)\}$

Complejidad: $O(1)$

Descripción: Obtiene el contenido de una celda del tablero dada una coordenada (i, j) .

Aliasing: ??

#LETRA TIENE JUGADOR(**in** j : juego, **in** x : letra, **in** i : nat) $\rightarrow res$: nat

Pre $\equiv \{i < \#jugadores(j)\}$

Post $\equiv \{res =_{\text{obs}} \#(x, \text{fichas}(j, i))\}$

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

$\text{Rep} : \text{juego_estr} \rightarrow \text{bool}$

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

donde

$\text{historiales} : \text{juego_estr} \rightarrow \text{multiconj}(\text{ocurrencia})$
 $\text{historiales}(e') \equiv \text{historialesHastaTiempo}(e'.\text{jugadores}, 0, e'.\text{tiempo})$

$\text{historialesHastaTiempo} : \text{ad}(\text{jugador}) \times \text{nat} \times \text{nat}$
 $\rightarrow \text{multiconj}(\text{ocurrencia})$
 $\text{historialesHastaTiempo}(js, k, t) \equiv$

if $k \geq \text{tam}(js)$
 then \emptyset
 else $\text{historialHastaTiempo}(js, k, t) \cup$
 $\text{historialesHastaTiempo}(js, k + 1, t)$

fi

$historialHastaTiempo : \text{ad}(\text{jugador}) \times \text{nat} \times \text{nat}$
 $\longrightarrow \text{multiconj}(\text{ocurrencia})$
 $historialHastaTiempo(js, k, t) \equiv historialHastaTiempo'(js[k].historial, t)$

$historialHastaTiempo' : \text{lista}(\text{tupla}(\text{ocurrencia}, \text{nat})) \times \text{nat}$
 $\longrightarrow \text{multiconj}(\text{ocurrencia})$
 $historialHastaTiempo'(ls, t) \equiv$

if $vacía?(ls)$
then \emptyset
else $historialHastaTiempo'(fin(ls), t) \cup$
 if $\pi_2(prim(ls)) < t$
 then $prim(ls)$
 else \emptyset
fi

fi

$ocurrenciasVálidas? : \text{tab} \times \text{multiconj}(\text{ocurrencia}) \longrightarrow \text{bool}$
 $ocurrenciasVálidas?(t, os) \equiv$

if $vacía?(os)$
then $true$
else $celdasLibres?(t, dameUno(os)) \wedge_L$
 $ocurrenciasVálidas?(ponerLetras(t, dameUno(os)), sinUno(os))$
fi

$ponerOcurrencias : \text{tab} \times \text{multiconj}(\text{ocurrencia}) \longrightarrow \text{tab}$
 $ponerOcurrencias(t, os) \equiv$

if $vacía?(os)$
then t
else $ponerOcurrencias(ponerLetras(t, dameUno(os)), sinUno(os))$
fi

$puntajeDeOcurrenciaEnTiempo : \text{estr_juego} \times \text{nat} \times \text{nat} \longrightarrow \text{nat}$
 $puntajeDeOcurrenciaEnTiempo(e, i, k) \equiv$
 $puntajePalabrasEstr(e.variante, t',$
 $palabrasUbicadas(ocurrenciasDePalabras(t'), e.jugadores[i].historial[k].ocurrencia))$

donde

$tiempo \equiv e.jugadores[i].historial[k].tiempo$
 $t' \equiv ponerOcurrencias(nuevoTablero(tamaño(e.tablero)),$
 if $tiempo = 0?$
 then \emptyset
 else $historialesHastaTiempo(e.jugadores, 0, tiempo - 1)$
 fi
 $\cup historialHastaTiempo(e.jugadores, i, tiempo)$
 $\cup \{e.jugadores[i].historial[k].ocurrencia\})$

$puntajePalabrasEstr : \text{variante} \times \text{tab} \times \text{conj}(\text{ocurrencia}) \longrightarrow \text{nat}$
 $puntajePalabrasEstr(v, t, os) \equiv$

```

if vacío?(os)
  then 0
  else  $puntajePalabraEstr(v, t, dameUno(os))$ 
    +  $puntajePalabras(v, t, sinUno(os))$ 
fi

```

$puntajePalabraEstr : \text{variante} \times \text{tab} \times \text{ocurrencia} \longrightarrow \text{nat}$
 $puntajePalabraEstr(v, t, o) \equiv$

```

if vacía?(o)
  then 0
  else  $puntajeLetra(v, \pi_3(dameUno(o)))$ 
    +  $puntajePalabra(v, t, sinUno(o))$ 
fi

```

Función de Abstracción

$\text{Abs} : \text{juego_estr } e \longrightarrow \text{juego}$

$\{\text{Rep}(e)\}$

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

Algoritmos

HACERGUIA(**in** A : guia, **in** parámetroInútil : Nat) \rightarrow bool

1:	$i \leftarrow 0$		\triangleright esto es $\Theta(1)$
2:	$n \leftarrow \text{guia.cantEjercicios}()$		$\triangleright \mathcal{O}(1)$
3:	$\text{consultas} \leftarrow \text{DICC} \text{VACIO}$		
4:	PREPARARMATE()		$\triangleright \Omega(n^n)$
5:	while $i < n$ do		
6:	PENSAREJERCICIO(1)		
7:	if TENGOCONSULTAS(i) then		
8:	ESCRIBIRCONSULTASEJERCICIO($i, \text{consultas}$)		
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:

NUEVOSEVIDOR(**in** $k : \text{nat}$, **in** $v : \text{variante}$) $\rightarrow res : \text{servidor}$

Pre $\equiv \{\text{true}\}$

Post $\equiv \{\exists r : \text{cola}(\text{letra}) \mid res =_{\text{obs}} \text{nuevoServidor}(k, v, r)\}$

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** $s : \text{servidor}$)

Pre $\equiv \{\neg \text{empezó?}(s) \wedge s =_{\text{obs}} S_0\}$

Post $\equiv \{s =_{\text{obs}} \text{conectarCliente}(S_0)\}$

Complejidad: $O(1)$

Descripción: Conecta un cliente a un servidor.

Aliasing: ??

CONSULTAR(**in/out** $s : \text{servidor}$, **in** $cid : \text{nat}$)

Pre $\equiv \{cid \leq \#conectados(s) \wedge s =_{\text{obs}} S_0\}$

Post $\equiv \{s =_{\text{obs}} \text{consultar}(S_0, cid)\}$

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** $s : \text{servidor}$, **in** $cid : \text{nat}$, **in** $o : \text{ocurrencia}$)

Pre $\equiv \{cid \leq \#conectados(s) \wedge s =_{\text{obs}} S_0\}$

Post $\equiv \{s =_{\text{obs}} \text{recibirMensaje}(S_0, cid, o)\}$

Complejidad: ??

Descripción: Recibe un mensaje de un cliente.

Aliasing: ??

CLIENTES ESPERADOS(**in** $s : \text{servidor}$) $\rightarrow res : \text{nat}$

Pre $\equiv \{\text{true}\}$

Post $\equiv \{res =_{\text{obs}} \#esperados(s)\}$

Complejidad: $O(1)$

Descripción: Obtiene el número de clientes esperados.

Aliasing: ??

CLIENTES CONECTADOS(**in** $s : \text{servidor}$) $\rightarrow res : \text{nat}$

Pre $\equiv \{\text{true}\}$

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 $\equiv \{\text{true}\}$
Post $\equiv \{res =_{\text{obs}} \text{juego}(s)\}$
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 acorde¹ al módulo de género **letra** con las siguientes operaciones en la interfaz (todas con orden de complejidad $O(1)$):

- **DOM** : $\rightarrow \text{nat}$ — Tamaño del dominio del tipo **letra**. Corresponde con la variable A de su especificación.
- **ORD** : **letra** $\rightarrow \text{nat}$ — Dada una letra, devuelve su correspondiente índice.
- **ORD**⁻¹ : $\text{nat } n \rightarrow \text{letra } \{n < A\}$ — 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 $n : \text{nat}$,
 in $f : \text{nat}$,
 in $\text{puntajes} : \text{dicc}(\text{letra}, \text{nat})$,
 in $\text{legítimas} : \text{conj}(\text{secu}(\text{letra}))$
) $\rightarrow \text{res} : \text{variante}$
Pre $\equiv \{n > 0 \wedge f > 0\}$
Post $\equiv \{\text{res} =_{\text{obs}} \text{nuevaVariante}(n, f, \text{puntajes}, \text{legítimas})\}$
Complejidad: $O(\#\text{legítimas} \cdot L_{\text{máx}})$
Descripción: Genera una variante de juego.
Aliasing: ??

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

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

PUNTAJELETRA(**in** $v : \text{variante}$, **in** $l : \text{letra}$) $\rightarrow \text{res} : \text{nat}$
Pre $\equiv \{\text{true}\}$
Post $\equiv \{\text{res} =_{\text{obs}} \text{puntajeLetra}(v, l)\}$
Complejidad: $O(1)$
Descripción: Devuelve el puntaje de una letra.

¹Una buena opción es usar un **Enumerado**.

Aliasing: ??

PALABRALEGÍTIMA?(**in** v : variante, **in** l : secu(letra)) $\rightarrow res$: bool

Pre $\equiv \{\text{true}\}$

Post $\equiv \{res =_{\text{obs}} \text{palabraLegítima}(v, l)\}$

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 $\equiv \{\text{true}\}$

Post $\equiv \{$

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

$\}$

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:

NUEVOABLERO(**in** n : nat) $\rightarrow res$: tab

Pre $\equiv \{n > 0\}$

Post $\equiv \{res =_{\text{obs}} \text{nuevoTablero}(n)\}$

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 $\equiv \{\text{enTablero}(t, i, j) \wedge \neg \text{hayLetra?}(t, i, j)\}$

Post $\equiv \{res =_{\text{obs}} \text{nuevoTablero}(n)\}$

Complejidad: $O(N^2)$

Descripción: Genera un tablero de tamaño n .

??

TAMAÑOABLERO(**in** v : variante) $\rightarrow res$: nat

Pre $\equiv \{\text{true}\}$

Post $\equiv \{res =_{\text{obs}} \text{tamañoTablero}(v)\}$

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** $v : \text{variante}$, **in** $l : \text{letra}$) $\rightarrow res : \text{nat}$

Pre $\equiv \{\text{true}\}$

Post $\equiv \{res =_{\text{obs}} \text{puntajeLetra}(v, l)\}$

Complejidad: $O(1)$

Descripción: Devuelve el puntaje de una letra.

Aliasing: ??

PALABRALEGÍTIMA?(**in** $v : \text{variante}$, **in** $l : \text{secu}(\text{letra})$) $\rightarrow res : \text{bool}$

Pre $\equiv \{\text{true}\}$

Post $\equiv \{res =_{\text{obs}} \text{palabraLegítima}(v, l)\}$

Complejidad: $O(L_{\text{máx}})$

Descripción: Determina si una palabra es legítima dentro de la variante de juego.

Aliasing: ??

LONGPALABRAMÁSLARGA(**in** $v : \text{variante}$) $\rightarrow res : \text{nat}$

Pre $\equiv \{\text{true}\}$

Post $\equiv \{$

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

$\}$

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 $\text{conj}(\text{tupla}(\text{nat}, \text{nat}, \text{letra}))$ 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`