Algoritmos y Estructuras de Datos II

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

Trabajo Práctico 1

Integrante	LU	Correo electrónico
Luz del Valle Rodriguez	687/18	rodriguez.luz.del.valle@gmail.com
Francisco Domato	388/19	franciscodomatocavs12@gmail.com
Matias Gangui	155/20	ganguimatias@gmail.com
Juan Manuel Viel	777/18	juan.mnul@gmail.com

Reservado para la cátedra

Instancia	Docente	Nota
Primera entrega		
Segunda entrega		

1. Definicion de Modulos Utilizados

ConjLin sera el Conjunto lineal del apunte de modulos basicos
ConjTrie sera un Conjunto implementado en un Trie
DiccLin sera el Diccionario lineal del apunte de modulos basicos
DiccTrie sera un Diccionario implementado en un Trie
Dicc1 es un diccionaio de Letra nat. Esta implementado como un array, donde en la posicion correspondiente al Letra esta ubicado el significado de la clave Letra. De esta forma obtener(Letra, nat) es en O(1).
String sera un vector de Letra
Cola sera la Cola del apunte de modulos basicos

2. Juego

2.1. Interfaz

Interfaz

se explica con: JUEGO

géneros: `Juego`.

JUEGO(**in** K : `nat`, **in** v : `variante`, **in** $Mazo$: `Cola(pair<Letra, nat>)`) $\rightarrow$ res : `Juego`
Pre $\equiv$ {true}
Post $\equiv$ {$res =_{\text{obs}}$ nuevoJuego(k, v, Mazo)}
Complejidad: $O(N^2 + \Sigma K + FK)$
Descripción: genera un nuevo juego.

JUGADAVALIDA(**in** o: `conj(tupla(nat, nat, nat))`) $\rightarrow$ res : `bool`
Pre $\equiv$ {true}
Post $\equiv$ {$res =_{\text{obs}}$ jugadaVálida(o)}
Complejidad: $\mathcal{O}(L^2_{max})$
Descripción: Dada una ocurrencia chequea que

- chequea que las posiciones estén libres
- las letras forman una linea recta horizontal o vertical
- se forme una palabra valida
- no queden huecos entre las letras al formar una palabra

UBICAR(**in** o: `conj(tupla(nat, nat, nat))`) $\rightarrow$ res : `void`
Pre $\equiv$ {$(\forall x : o)(\pi_0(x) < tamaño(tablero(Juego)) \wedge \pi_1(x) < tamaño(tablero(Juego)))$}
Post$\equiv$ {$(\forall x : o)(tablero(\pi_0(x), \pi_1(x), Juego) = \pi_2(x))$}
Complejidad:$O(\|o\|)$
Descripción: Ubica en el tablero las fichas de la ocurrencia o

VARIANTE() $\rightarrow$ res : `variante`
Pre $\equiv$ {true}
Post $\equiv$ {res = variante(Juego)}
Complejidad: $\mathcal{O}(1)$

TURNODELJUGADOR() $\rightarrow$ res : `nat`
Pre $\equiv$ {true}
Post $\equiv$ {res = turno(Juego)}
Complejidad: $\mathcal{O}(1)$

PUNTAJEDELJUGADOR(**in** id: `nat`) $\rightarrow$ res : `nat`
Pre $\equiv$ {true}
Post $\equiv$ {res = puntaje(Juego, id)}
Complejidad: O(m*L_{max})

TABLEROIJ(**in** $self$: `juego`, **in** i: `nat`, **in** j : `nat`) $\rightarrow$ res : `Letra`
Pre $\equiv$ {true}

Post $\equiv$ {,}
Complejidad: $i < tamaño(tablero(Juego)) \wedge j < tamaño(tablero(Juego)) \wedge hayLetra?(tablero, i, j)$ $res =_{\text{obs}} letra(tablero(Juego), i, j)$ [$\mathcal{O}(1)$] [Obtiene la letra en la posicion dada]

CantFichasPorJugador(**in** *self* : `juego`, **in** *id*: `nat`, **in** *f* : `tupla(Letra, nat)`) $\rightarrow$ *res* : `nat`
Pre $\equiv$ {true}
Post $\equiv$ { res = obtener(fichas(Juego, id), $\pi_1(f)$)}
Complejidad:O(1)

2.2. Representación

PuntosQueSeSuman: nat
TamañoTablero: nat
Ficha: tupla < Letra, nat >
Palabra: secu(letra)
Ocurrencia: conj(tupla(nat, nat, letra))
Casillero : tupla(letra, nat)
PuntosTotales : nat
Mano : multiconj(Ficha)
JugadasPasadas = cola(pair(Ocurrencia, nat))
Jugador = tupla(PuntosTotales, Mano, JugadasPasadas)

`tablero` **se representa con** `tablero_estr`

donde `tablero_estr` es `tupla`(*tamaño*: `nat` , *casilleros*: `secu(secu(casillero))`)

Rep : tablero_estr $\longrightarrow$ bool
Rep(t) $\equiv$ true $\iff$ (t.casilleros.long() = t.tamaño $\wedge$
($\forall i$: nat)($0 \leq i <$ t.casilleros.long() $\Rightarrow$ t.casilleros[i].long() = t.tamaño))

Abs : estr e $\longrightarrow$ tablero {Rep(e)}
Abs(e) $=_{\text{obs}}$ t: tablero | e.tamaño = tamaño(t) $\wedge$
($\forall i, j$: nat)($0 \leq i, j <$ tamaño(t) $\wedge_l$ hayLetra?(t, i, j) $\Rightarrow$
$\pi_0(e.casilleros$[i][j]) = $letra$(t, i, j)) $\wedge$
($\forall i, j$: nat)($0 \leq i, j <$ tamaño(t) $\wedge_l \neg$hayLetra?(t, i, j) $\Rightarrow$ $\pi_0(e.casilleros$[i][j]) =′ ′)

`variante` **se representa con** `variante_estr`

donde `variante_estr` es `tupla`(*tamañoTablero*: `nat` , *cantidadFichas*: `nat` , *letrasPuntaje*: `Dicc1(letra:nat)` , *palabrasLegitimas*: `ConjTrie(palabra)`)

Rep : variante_estr $\longrightarrow$ bool
Rep(v) $\equiv$ true $\iff$ (El cardinal del conjunto de claves de v.letrasPuntaje es menor o igual a v.cantidadFichas $\wedge$
($\forall p$: palabra)((PERTENECE(v.palabrasLegitimas, p) $\Rightarrow$ (palabras.long() $\leq$v.$cantidadFichas$)$\wedge$
(($\forall l$: letra)(l $\in$ *palabra* $\Rightarrow$ existe c letra $\in$ $clavesde$v.$letrasPuntaje/l = c$)))

Abs : estr e $\longrightarrow$ variante {Rep(e)}
Abs(e) $=_{\text{obs}}$ v: variante | e.tamaño = tamañoTablero(v) $\wedge$
e.cantidadFichas = fichas(v) $\wedge$
($\forall l$: letra)(DEFINIDO(e.letrasPuntaje, l) $\Longrightarrow$
$SIGNIFICADO(e.letrasPuntaje, l) = puntajeLetra(v, l)) \wedge$
($\forall p$: palabra)(PERTENECE(e.palabrasLegitimas, p) $\iff$ $palabraLegitima(v, p)$)

`juego` se representa con `juego_estr`

donde `juego_estr` es `tupla`(*variante*: `variante` , *tablero*: `tablero` , *turno*: `nat` , *mazoDeFichas*: `cola(Ficha)` , *Jugadores*: `secu(Jugador)`)

Rep : juego_estr $\longrightarrow$ bool
Rep(j) $\equiv$ true $\Longleftrightarrow$ (

1. ($\forall i$: nat)($0 \leq i <$ e.Jugadores.long() $\Rightarrow$ #(π_1 (j.Jugadores[i])) = j.variante.CantidadFichas
2. j.variante.tamañoTablero = j.tablero.tamaño
3. Las fichas dadas al inicio del juego no se modifican, pueden estar en el mazo, en la mano de los jugadores o en el tablero.
4. Cada palabra presente en el tablero tiene que estar presente en palabrasLegítimas dentro de la variante del juego.

)

Abs : estr e $\longrightarrow$ juego {Rep(e)}
Abs(e) =$_{\text{obs}}$ j: juego | e.variante = variante(j)$\wedge$
e.tablero = tablero(j)$\wedge$
e.turno = turno(j) $\wedge$
e.Jugadores.long() = #jugadores(j) $\wedge$
($\forall i$: nat)($0 \leq i <$ e.Jugadores.long() $\Rightarrow$ π_2(e.Jugadores[i]) = fichas(j, i)) $\wedge$
($\forall i$: nat)($0 \leq i <$ e.Jugadores.long() $\Rightarrow$ π_0 (e.Jugadores[i]) = puntaje(j, i)) $\wedge$
e.mazoDeFichas = repositorio(j)

Todas las palabras validas de la estructura pertenecen al conj de palabras legitimas de la variante juego, y todas las palabras legitimas de la variante del juego pertenecen a las palabras validas de la estructura.

2.3. Pseudocodigo Juego

La funcion LetraToInt del algoritmo Juego toma una Letra y devuelve el int correspondiente en el alfabeto (en español lleva a$-> 0$; b $-> 1$; ... z $-> 25$)

Juego(**in** K : `nat`, **in** v : `variante`, **in** $Mazo$: `cola(letra, nat)` $\rightarrow$ juego

1: self.tablero = array de N por N lleno de espacios " " (N = v.tamañoTablero) ▷ $\mathcal{O}(n^2)$
2: self.Turno = 0 ▷ $\mathcal{O}(1)$
3: self.MazoDeFichas = Mazo ▷ $\mathcal{O}(n^2 + FK)$
4: self.Jugadores se puebla con las fichas tomadas del mazo usando self.mazo.desencolar() para cada jugador hasta alcanzar la cantidad de fichas indicadas por la variante.
5: self.variante $\leftarrow$ v
6: **return** juego

JugadaValida(**in/out** $self$: `juego`, **in** o: `conj(tupla(nat, nat, Letra))` $\rightarrow$ res: $bool$

```
1: if o.size() > self.L_max then
2:     return res ← false
3: end if
4:
5: self.Ubicar(o)
6: horizontal ← EsHorizontal?(o)                                   ▷ O(L_max)
7: vertical ← EsVertical?(o)                                       ▷ O(L_max)
8:
9: if ¬horizontal ∧ ¬vertical then
10:    self.desUbicar(o)
11:    return res ← false
12: end if
13:
14: if vertical then
15:    palabra_main ← string vacía
16:    f_0 ← o[0][0]
17:    c_0 ← o[0][1]
18:    f ← f_0
19:    for f > 0 ∧ self.tablero[f][c_0]! = string vacía; f − − do   ▷ O(L_max)
20:        skip
21:    end for
22:    for f < self.LadoTablero ∧ self.tablero[f][c_0]! = string vacía; f + + do   ▷ O(L_max)
23:        palabra_main+ = self.tablero[f][c_0]
24:    end for
25:    if o ∉ palabra_main ∨ main ∉ self.PalabrasLegitimas then
26:        self.desUbicar(o)
27:        return res ← false
28:    end if
29:    for x in o do                                               ▷ O(L_max^2)
30:        palabra ← string vacía
31:        f_0 ← x[0]
32:        c_0 ← x[1]
33:        c ← c_0
34:        for c > 0 ∧ self.tablero[f_0][c]! = string vacía; c − − do   ▷ O(L_max)
35:            skip
36:        end for
37:        for c < self.LadoTablero ∧ self.tablero[f_0][c]! = string vacía; c + + do   ▷ O(L_max)
38:            palabra+ = self.tablero[f_0][c]
39:        end for
40:        if palabra ∉ self.PalabrasLegitimas then
41:            self.desUbicar(o)
42:            return res ← false
43:        end if
44:    end for
45: end if
46: if horizontal then
47:    análogo a vertical
48: end if
49: self.desUbicar(o)
50: return res ← true
```

EsHorizontal?(**in/out** *self* : `juego`, **in** *o*: `conj(tupla(nat, nat, Letra))` → *res*: *bool*

1: $f_0 \leftarrow o[0][0]$
2: **for** tupla(nat, nat, Letra) x : o **do** ▷ $\mathcal{O}(L_{max})$
3: f ← x[0]
4: **if** $f_0 \neq f$ **then**
5: $res \leftarrow false$
6: **end if**
7: **end for**
8: $res \leftarrow true$

Ubicar(**in/out** *self* : `juego` **in** *o*: `conj(tupla(nat, nat, letra))`

1: **for** tupla(nat, nat, Letra) x : o **do** ▷ $\mathcal{O}(m)$
2: nat f = x[0] ▷ $\mathcal{O}(1)$
3: nat c = x[1] ▷ $\mathcal{O}(1)$
4: self.Tablero[f][c] = make_pair(x[2], self.Turno) ▷ $\mathcal{O}(1)$
5: **end for**

desUbicar(**in/out** *self* : `juego`, **in** *o*: `conj(tupla(nat, nat, Letra))`

1: **for** Tupla(nat, nat, Letra) x : o **do** ▷ $\mathcal{O}(m)$
2: nat f = x[0] ▷ $\mathcal{O}(1)$
3: nat c = x[1] ▷ $\mathcal{O}(1)$
4: self.Tablero[f][c] = string vacio ▷ $\mathcal{O}(1)$
5: **end for**

Variante(**in** *self* : `juego`) → variante

1: return tupla<this.TamañoTablero,this.CantFichasPorJugador,this.DiccFichas,this.PalabrasValidas>

TurnoDelJugador(**in** *self* : `juego`) → nat

1: return self.Turno

PuntajeDeJugador(nat id)

```
1: Jugador J = self.Jugadores[id]                                                    ▷ O(1)
2: Cola(pair(Ocurrencia, nat)) Old = J.JugadasPasadas                                ▷ O(1)
3: while Old.size() > 0 do                                                           ▷ O(m)
4:     (O, turno) = Old.desencolar()                                                 ▷ O(1)
5:     bool horizontal = EsHorizontal?(O)                                            ▷ O(L_max)
6:     if horizontal then
7:         for tupla(nat, nat, Letra) x : O do
8:             f_0 = x[0]                                                          ▷ O(1)
9:             c_0 = x[1]                                                          ▷ O(1)
10:            for int f = f_0; f ≥ 0 ∧ self.tablero[f][c_0].first ≠ ' ' ∧
11:                                self.tablero[f][c_0].second ≤ turno; f -= 1 do   ▷ O(L_max)
12:                skip                                                            ▷ O(1)
13:            end for
14:            for ;f < self.Variante.TamañoTablero ∧ self.tablero[f][c_0].first ≠ ' ' ∧
15:                                self.tablero[f][c_0].second ≤ turno; f++ do      ▷ O(L_max)
16:                J.Puntaje += obtener(x[2], self.Variante.LetrasPuntaje)         ▷ O(1)
17:            end for
18:        end for
19:        f_0 = O[0][0]                                                           ▷ O(1)
20:        c_0 = O[0][1]                                                           ▷ O(1)
21:        for int c = c_0; f ≥ 0 ∧ self.tablero[f_0][c].first ≠ ' ' ∧
22:                            self.tablero[f_0][c].second ≤ turno; c -= 1 do      ▷ O(L_max)
23:            skip                                                                ▷ O(1)
24:        end for
25:        for ;c < self.Variante.TamañoTablero ∧ self.tablero[f_0][c].first ≠ ' ' ∧
26:                            self.tablero[f_0][c].second ≤ turno; c++ do         ▷ O(L_max)
27:            J.Puntaje += obtener(x[2], self.Variante.LetrasPuntaje)             ▷ O(1)
28:        end for
29:    else
30:        analogo a horizontal
31:    end if
32: end while
33: return J.TotalPuntajes
```

TableroIJ(nat i, nat j)

```
1: return self.Tablero[i][j].0
```

CantFichasPorJugador(nat id, tupla<Letra, nat> ficha)

```
1: dicc≪Letra, nat>, nat> dicc = self.FichasDeJugadores[id]
2: return dicc.obtener(ficha)
```

actualizarHistorial(ocurrencia o, nat id)

```
1: self.jugadores[id].historialJugadas.encolar(o, self.turno)
```

reponer(nat cantidadLetras, nat id)

```
1: i ← 0
2: while i < cantidadLetras do
3:     l ← self.mazoDeFichas.desencolar()
4:     self.jugadores[id].mano.agregar(l)
5:     i ← i + 1
6: end while
```

3. Servidor

3.1. Interfaz

Interfaz

se explica con: SERVIDOR

géneros: `Servidor`.

SERVIDOR(**in** n : `nat`, **in** k : `nat`, **in** f : `nat`, **in** L : `conj(string)`, `in` $self$: `Servidor`, **in** $Mazo$: `cola(tupla(Letra, nat))`, **in** $fichas$: `dicc(Letra, nat)`, **in** e : `nat`) $\rightarrow$ res : `servidor`
Pre $\equiv$ {true}
Post $\equiv$ {$res =_{\text{obs}}$ gen}
Complejidad: $O(N^2 + \Sigma K + FK)$
Descripción: Crea un nuevo servidor

CONECTARCLIENTE(**in** $self$: `Servidor`,**in** s : `Servidor`) $\rightarrow$ res : `nat`
Pre $\equiv$ {$C_0 = \#conectados(Servidor)$}
Post$\equiv$ {res = $C_0 + 1$}
Complejidad:O(1)

CONSULTAR(**in** $self$: `Servidor`,**in** id : `nat`) $\rightarrow$ res : `cola(Notificaciones)`
Pre $\equiv$ {true}
Post $\equiv$ {Servidor $=_{\text{obs}}$ notificaciones(Servidor)[id]}
Complejidad: $\mathcal{O}(n)$

RECIBIRMENSAJE(**in** $self$: `Servidor`,**in** o : `ocurrencia`, **in** id : `nat`)
Pre $\equiv$ {true}
Post $\equiv$ {Servidor $=_{\text{obs}}$ recibirMensaje(o, id)}
Complejidad: $3\mathcal{O}(|o|)$

ESPERADOS(**in** $self$: `Servidor`) $\rightarrow$ res : `nat`
Pre $\equiv$ {True}
Post $\equiv$ {res = #esperados(servidor)}
Complejidad: $\mathcal{O}(1)$

CONECTADOS(**in** $self$: `Servidor`) $\rightarrow$ res : `nat`
Pre $\equiv$ {True}
Post $\equiv$ {res = #conectados(servidor)}
Complejidad: $\mathcal{O}(1)$

JUEGO(**in** $self$: `Servidor`) $\rightarrow$ res : `&Juego`
Pre $\equiv$ {True}
Post $\equiv$ {res = juego(Servidor)}
Complejidad: $\mathcal{O}(1)$
Aliasing: el resultado es una referencia al juego

3.2. Representación

Esperados: nat
Conectados: nat

`tipoNotificacion` **se representa con** `tipoNotificacion_estr`

donde `tipoNotificacion_estr` es `tupla`(IdCliente, Empezar, TurnoDe, Ubicar, Reponer, SumaPuntos, Mal)

`notification` **se representa con** `notificacion_estr`

donde `notificacion_estr` es `tupla`(*tipoNotif*: `TipoNotificacion` , *id*: `nat` , *puntosQueSeSuman*: `nat` , *mazoDeFichas*: `cola(Ficha)` , *ocurrencia*: `ocurrencia`)

Rep : notificacion_estr $\longrightarrow$ bool
Rep(n) $\equiv$ true $\iff$ (n = <"IdCliente", id, 0, 0, vacio, vacio > $\vee$
n = <"Empezar", 0, 0, nat, vacio, vacio> $\vee$
n = <"TurnoDe", id, 0, 0, vacio, vacio> $\vee$
n = <"Ubicar", id, 0, 0, vacio, ocurrencia> $\vee$
n = <"Reponer", 0, 0, 0, cola(letra), vacio> $\vee$
n = <"SumaPuntos", id, nat, 0,vacio, vacio> $\vee$
n = <"Mal", 0, 0, 0, vacio, vacio>)

Abs : estr e $\longrightarrow$ notificacion {Rep(e)}
Abs(e) $=_{\text{obs}}$ n: notificacion | e.tipoNotif = π_0 (datos(n)) $\wedge$
e.id = π_1 (datos(n)) $\wedge$
e.puntosQueSeSuman = π_2 (datos(n)) $\wedge$
e.mazoDeFichas = π_3 (datos(n)) $\wedge$
e.ocurrencia = π_4 (datos(n))

`Servidor` **se representa con** `Servidor`

donde `Servidor` es `tupla`(*Juego*: `Juego`, *Notificaciones*: `Array(cola(Notificacion))`, *InfoDelServer*: `tupla(Esperados, Conectados)`)

Rep : Servidor $\longrightarrow$ bool

Rep(s) $\equiv$ true $\iff$ (

1. π_0(s.InfoDelServer) = tamaño(Juego.Puntajes) = tamaño(s.Notificaciones)
2. π_0(s.InfoDelServer) $\geq$ π_1(s.InfoDelServer)
3. Para toda notificacion en Notificaciones vale que π_1(Notificacion) $\leq$ π_0(Servel.InfoDelServer)
4. Si en la cola de notificaciones está la notificación EMPEZAR debe ser luego de que se hayan conectado la cantidad esperada de jugadores.
5. Si en la cola de notificaciones está la notificación UBICAR:
   a) A continuación, en la cola de Notificaciones, deben estar las notificaciones SUMAPUNTOS (asignada a todos los jugadores), REPONER (asignada al mismo jugador) y TURNODE.
   b) El turno enviado por la notificación TURNODE debe ser (id de ubicar + 1) mod cantidad de jugadores.
   c) El tablero debe actualizarse, ubicando las fichas en las posiciones definidas por la ocurrencia de UBICAR.
   d) En FichasDeJugadores debe modificarse el array correspondiente al id de UBICAR. Disminuye el valor de las posiciones correspondientes a las letras dadas en la ocurrencia de UBICAR.
   e) Se modifica Puntajes. En la posición correspondiente al id de UBICAR debe incrementarse el valor de acuerdo a los puntos que le otorgue la ocurrencia de la notificación UBICAR.
6. Si en la cola de notificaciones está la notificación IDCLIENTE:
   a) El id de la notificación debe ser menor o igual a la cantidad de jugadores esperados.
   b) Si en la cola de Notificaciones le siguen las notificaciones EMPEZAR y TURNODE, el id de IDCLIENTE debe ser igual a la cantidad de jugadores esperados.
7. Si en la cola de notificaciones está la notificación MAL no deben realizarse cambios en el juego ni el servidor, salvo enviar la notificación MAL a todos los jugadores.

)

Abs : estr e $\longrightarrow$ Servidor {Rep(e)}
Abs(e) $=_{\text{obs}}$ s: Servidor | e.InfoDelServer[0] = #esperados(s) $\wedge$
e.InfoDelServer[1] = #conectados(s) $\wedge$
tupla(s.Juego.variante, s.Juego.MazoDeFichas) = configuracion(s) $\wedge$
e.Juego = juego(s) $\wedge$
($\forall$*cid*: nat)(cid < e.InfoDelServer[1] $\Rightarrow$
e.Notificaciones[cid] = notificaciones(s, cid)

3.3. Pseudocodigo

Servidor(in self: Servidor, *int* N, *int* K, *int* F, conj<String> &L, cola<pair<Letra, nat≫ &Mazo, dicc<Letra, nat> &Fichas, int E)

```
1: self.Juego = Juego(N, K, F, L, Mazo, Fichas)
2: self.Notificaciones = { }
3: self.Esperados = E
4: self.Conectados = 0
```

ConectarCliente(inout self: Servidor)

```
1: self.Conectados++
```

Consultar(int id, inout self: Servidor)

```
1: cola(Notificaciones) res = self.Notificaciones[id]
2: self.Notificaciones[id] = { }
3: return res
```

RecibirMensaje(in self: Servidor,ocurrencia o, nat id)

```
1: if self.juego.jugadaValida?(o) and id = (self.juego.turno  mód  self.jugadores.long()) then
2:     self.juego.ubicar(o)
3:     self.juego.actualizarHistorial(o, id)
4:     self.juego.reponer(id, o.long())
5:     self.juego.avanzarTurno(id)
6: else
7:     self.notificaciones.encolar(Notificación("Mal", 0, 0, 0, vacío, vacío))
8: end if
```

Esperados(inout self: Servidor)

```
1: return self.Esperados
```

Conectados(inout self: Servidor)

```
1: return self.Conectados
```

Juego(inout self: Servidor)

```
1: return &self.Juego
```