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
Church, Alonso	1/20	alonso@iglesia.com
Lovelace, Ada	10/19	ada_de_los_dientes@tatooine.com
Null, Linda	100/18	null@null.null
Turing, Alan	314/16	halting@problem.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.
- $L_{máx}$ — 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: ??

operaciones :

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

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

UBICAR(**in/out** `j`: `juego`, **in** o : `occurrencia`)
Pre $\equiv \{jugadaVálida(j, o) \land j =_{\text{obs}} J_0\}$
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 juego.
Aliasing: ??
Requiere: ??

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

Requiere: ??

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: ??
Requiere: ??

PUNTAJE(**in** j : `juego`, **in** i : `nat`) $\rightarrow$ res : `nat`
Pre $\equiv$ $\{i < \#jugadores(j)\}$
Post $\equiv$ $\{res =_{\text{obs}} puntaje(j, i)\}$
Complejidad: $O(m \cdot L_{máx})$, 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: ??
Requiere: ??

CELDA(**in** J : `juego`, **in** i : `nat`, **in** j : `nat`,) $\rightarrow$ res : `puntero(letra)`
Pre $\equiv$ $\{enTablero?(tablero(J), i, j)\}$
Post $\equiv$ $\{res =_{\text{obs}}$ **if** $hayLetra?(tablero(J), i, j)$ **then** $\&letra(tablero(J), i, j)$ **else** $NULL$ **fi**$\}$
Complejidad: $O(1)$
Descripción: Obtiene el contenido del tablero en una coordenada (i, j).
Aliasing: ??
Requiere: ??

#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: ??
Requiere: ??

2.2. Implementación

Representación

`foo` se representa con `estr`

donde `estr` es `tupla`(*foo*: `bar` , *foo*: `bar`)

Invariante de Representación

Rep : estr $\longrightarrow$ bool
Rep(e) $\equiv$ true $\Longleftrightarrow$ foo

Función de Abstracción

Abs : estr e $\longrightarrow$ foo {Rep(e)}

Abs(e) $=_{obs}$ p: foo | bar

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)
```

3. Módulo Servidor

3.1. Interfaz

3.2. Implementación

4. Módulos auxiliares

4.1. Módulo Foo

4.1.1. Interfaz

4.1.2. Implementación