Algoritmos y Estructuras de Datos II

Segundo Cuatrimestre de 2016

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

Trabajo Practico 2

Especificacion

Grupo De TP Algo2

Integrante	LU	Correo electrónico
Fernando Castro	627/12	fernandoarielcastro92@gmail.com
Philip Garrett	318/14	garrett.phg@gmail.com
Gabriel Salvo	564/14	gabrielsalvo.cap@gmail.com
Bernardo Tuso	792/14	btuso.95@gmail.com

Reservado para la cátedra

Instancia	Docente	Nota
Primera entrega		
Segunda entrega		

Índice

1. Modulos

Esta es un disenio(no tengo enie, paja) de la especificacion del Trabajo Practico 2 del 2^{do} cuatrimestre del 2016 presentada por la catedra para la realizacion del Trabajo Practico 2. Ver enunciado:
`http://www.dc.uba.ar/materias/aed2/2016/2c/descargas/tps/tp2/view`

2. Módulo Coordenada

Interfaz

Representación

`Coordenada` **se representa con** `estr`

donde `estr` es `tupla`(x: `Nat`, y: `Nat`)

1) True
Abs(e): estre - > Coor Rep(e)
c : Coor tq e.x = latitud(c) y e.y = longitud(c)

3. Módulo Mapa

Interfaz

Representación

`Mapa` se representa con `estr`

donde `estr` es `tupla`(*coordenadas*: `ConjLineal`, *ancho*: `Nat`)

1) El ancho es igual al maximo de las coordenadas X.
1) e.Ancho = Max(TT1(e.Coordenadas))
Abs(e): estre - > Mapa Rep(e)
m : Mapa tq e.coordenadas = coordenadas(m)

4. Módulo Juego

Interfaz

usa: MAPA, COORDENADA.

se explica con: JUEGO.

géneros: `juego`.

CREARJUEGO(**in** m: `mapa`) $\rightarrow$ res : `juego`
Pre $\equiv \{m = m_0\}$
Post $\equiv \{res =_{\text{obs}} \text{crearJuego}(m_0) \wedge \text{mapa}(res) =_{\text{obs}} m_0\}$

AGREGARPOKEMON(**in/out** j: `juego`, **in** c: `coordenada`, **in** p: `pokemon`) $\rightarrow$ res : `pokemon`
Pre $\equiv \{j =_{\text{obs}} j_0\}$
Post $\equiv \{j =_{\text{obs}} agregarPokemon(res, \text{c}, \text{j}_0)\}$
Complejidad:$\Theta(|p| + EC * log(EC))$

AGREGARJUGADOR(**in/out** j: `juego`) $\rightarrow$ res : `jugador`
Pre $\equiv \{j =_{\text{obs}} j_0\}$
Post $\equiv \{j =_{\text{obs}} \text{agregarJugador}(res, j_0) \wedge \neg estaConectado(\text{j, res}) \wedge \neg vacio?(\text{pokemons(j, res)}\}$
Complejidad:$\Theta(J)$

Representación

`Juego` **se representa con** `estr`

donde `estr` es `tupla`(*pokemones*: `diccTrie`, *jugadores*: `conjLineal` , *jugadoresPorPosicion*: `conjHash` , *pokemonesPorPosicion*: `conjHash` , *jugadoresEnRango*: `diccHeap` , *mapa*: `Mapa` , *pT*: `Nat`)

`Jugador` **se representa con** `jug`

donde `jug` es `tupla`(*id*: `Nat`, *posicion*: `Coordenada` , *estaConectado*: `Bool` , *sanciones*: `Nat` , *pokeCapturados*: `ConjLineal`)

`Pokemon` **se representa con** `poke`

donde `poke` es `tupla`(*tipo*: `String`, *contador*: `Nat` , *posicion*: `Coordenada` , *salvaje*: `Bool`)

Rep: estr -> bool
1) La suma de toos los significados de pokemons es igual al PT
2) Todos las posiciones de jugPorPosicion esta contenida en el heap
3) Idem pokePorPosicion
4) Todo jugador que esta conectado y no expulsado, existe en jugPorPosicion
5) Para cada posicion hay un jugador en jugPorPosicion que pertenece a jugadores
6) Para cada pos en pokePorPosicion hay pokemon en pokemones
7) Para cada posicion en jugadoresEnRango, sus jugadores estan contenidos en jugadores
8) Para cada jugador en jugadores: si no esta expulsado, sus pokemones estan contenidos en pokemones del juego y no estan en pokemonesPorPosicion; y si esta conectado, su posicion pertenece al mapa del juego
9) Para cada pokemon en pokemones, si es salvaje: su contador es menor a 10, su posicion pertenece al mapa del juego y pertenece a pokemonEnPosicion

Abs(e): estre - > Jugo Rep(e)
pGo: Juego tq e.mapa = mapa(pGo) y e.jugadores = jugadores(pGo) yluego
(Para todo j : jugador) j pertenece e.jugadores impluego
j.sanciones = sanciones(j, pGo) ((j pertenece expulsados(pGo) y j.sanciones ¿= 10)
oluego (j.pokesCapturados = pokemones(j,pGo) y j.estaConectado = estaConectad(j,pGo)
y j.estaConectado impluego j.pos = posicion(j,pGo))) y
(Para todo p : pokemon) p pertenece c.pokemones impluego (Para todo j : Jugador)
j pertenece e.jugadores yluego p pertenece pokemones(j,pGo) o [(Para todo c : coord)
c pertenece e.mapa.coordenadas yluego p = pokemonEnPos(c,pGo) y cantMovParaCap(c,pGo)

p.contador]

5. Módulo Tabla de Valores(*coordenada*, σ)

El módulo Tabla de Valores provee un diccionario por posiciones en el que se puede definir, borrar, y testear si hay un valor en una posicion en tiempo $O(1)$.

El principal costo de paga al crear la estructura, dado de cuesta tiempo lineal *ancho* por *largo*.

Interfaz

parámetros formales
géneros $coordenada, \sigma$

se explica con: DICCIONARIO(κ, σ),

géneros: `tabla`($coordenada, \sigma$).

Trabajo Practico 2Operaciones básicas de tabla

VACÍO(**in** Nat : `a` ncho, **in** Nat : `l` argo) $\rightarrow$ res : `tabla`($coordenada, \sigma$)
Pre $\equiv$ {ancho $> 0 \wedge$ largo > 0}
Post $\equiv$ {$res =_{\text{obs}}$ vacio}
Complejidad: Θ(*ancho*largo*)
Descripción: genera una tabla vacía.

DEFINIR(**in/out** t : `tabla(`$coordenada, \sigma$`)`, **in** c : $coordenada$, **in** s : σ)
Pre $\equiv$ {$t =_{\text{obs}} t_0$}
Post $\equiv$ {$t =_{\text{obs}}$ definir(t, c, s)}
Complejidad: $\Theta(1)$
Descripción: define el significado s en la tabla, en la posicion representada por c.
Aliasing: Hay alising, pero no se como explicarlo TODO

DEFINIDO?(**in** t : `tabla(`$coordenada, \sigma$`)`, **in** c : $coordenada$) $\rightarrow$ res : `bool`
Pre $\equiv$ {true}
Post $\equiv$ {$res =_{\text{obs}}$ def?(t, c)}
Complejidad: $\Theta(1)$
Descripción: devuelve `true` si y sólo c tiene un valor en la tabla.

SIGNIFICADO(**in** t : `tabla(`$coordenada, \sigma$`)`, **in** c : $coordenada$) $\rightarrow$ res : σ
Pre $\equiv$ {def?(t, c)}
Post $\equiv$ {alias($res =_{\text{obs}}$ Significado(t, c))}
Complejidad: $\Theta(1)$
Descripción: devuelve el valor en la posicion c de t.

BORRAR(**in/out** t : `tabla(`$coordenada, \sigma$`)`, **in** c : $coordenada$)
Pre $\equiv$ {$t = t_0 \wedge$ def?(t, c)}
Post $\equiv$ {$t =_{\text{obs}}$ borrar(t_0, c)}
Complejidad: $\Theta(1)$
Descripción: elimina el valor en la posicion c en t.

6. Módulo Lista Enlazada Acotada(puntero(α))

El módulo Lista Enlazada Acotada provee una lista por posiciones en el que se puede definir, borrar, y testear si hay un valor en una posicion en tiempo $O(1)$.

Interfaz

parámetros formales
géneros α
función COPIAR(**in** a: α) $\rightarrow$ res : α
Pre $\equiv$ {true}
Post $\equiv$ {$res =_{\text{obs}} a$}
Complejidad: $\Theta(copy(a))$
Descripción: función de copia de α's

se explica con: SECUENCIA(α), ITERADOR BIDIRECCIONAL(α).

géneros: `listaAc`(α), `itListaAc`(α).

Trabajo Practico 2Operaciones básicas de lista

VACÍA(**in** lon: `Nat`, **in** an: `Nat`) $\rightarrow$ res : `listaAc`(α)
Pre $\equiv$ {$an > 0 \wedge lon > 0$}
Post $\equiv$ {$long(res) = lon$}
Complejidad: $\Theta(lon)$
Descripción: Genera una lista de largo lon, donde cada nodo es un puntero a null.

INSERTAR(**in/out** l: `listaAc`(α), **in** d: α, **in** x: `Nat`, **in** y: `Nat`) $\rightarrow$ res : `itListaAc`(α)
Pre $\equiv$ {$f(x,y) < long(l) \wedge l =_{\text{obs}} l_0 \wedge \neg$est?($l_0$, d)}
Post $\equiv$ {est?(l_0, d)}
Complejidad: $\Theta(1)$
Descripción: agrega el elemento d en la posicion i-esima de la lista. Retorna un iterador a l, de forma tal que Siguiente devuelva d.

LONGITUD(**in** l: `listaAc`(α)) $\rightarrow$ res : `Nat`
Pre $\equiv$ {true}
Post $\equiv$ {$res =_{\text{obs}} long(l)$}
Complejidad: $\Theta(1)$
Descripción: devuelve la longitud de la lista l

HAYALGUIENAHI?(**in** l: `listaAc`(α), **in** x: `Nat`, **in** y: `Nat`) $\rightarrow$ res : `bool`
Pre $\equiv$ {$f(x,y) < long(l)$}
Post $\equiv$ {res $=_{\text{obs}}$ est?(l, f(x,y))}
Complejidad: $\Theta(1)$
Descripción: devuelve `true` si y sólo si hay algo un elemento que no apunta a null en esa posicion

IESIMO(**in/out** l: `listaAc`(α), **in** x: `Nat`, **in** y: `Nat`) $\rightarrow$ res : α
Pre $\equiv$ {$f(x,y) < long(l)$}
Post $\equiv$ {alias($res =_{\text{obs}} iesimo(l, f(x,y))$)}
Complejidad: $\Theta(1)$
Descripción: devuelve el elemento en la posicion f(x,y)

BORRARIESIMO(**in/out** l: `listaAc`(α), **in** x: `Nat`, **in** y: `Nat`)
Pre $\equiv$ {$f(x,y) < long(l) \wedge l =_{\text{obs}} l_0$}
Post $\equiv$ {alias($secuSuby(l_0, l)$)}
Complejidad: $\Theta(1)$
Descripción: elimina el elemento en la posicion f(x,y)