Introducción a la Computación (Matemática)

Primer Cuatrimestre de 2019

Tipos Abstractos de Datos

Repaso: Tipos Abstractos de Datos

- Parte pública: Disponible para el usuario externo.
  - Nombre y tipos paramétricos (ej: Fecha, Lista(ELEM)).
  - Operaciones, sus especificaciones y posiblemente sus órdenes de complejidad temporal.
- Parte privada: Sólo accesible desde dentro del TAD. ¡El usuario externo **nunca** debe ver ni meterse en esto!
  - **Estructura de representación** del TAD.
  - **Invariante de representación**: qué propiedades debe cumplir la estructura elegida para que tenga sentido como la representación de una instancia del TAD.
  - **Algoritmos de las operaciones** para la estructura de representación elegida.

TAD Pila(ELEM)

Operaciones:

- CrearPila() $\rightarrow$ Pila(ELEM): Crea una pila vacía.
- P.Apilar(x): Inserta el elem. x sobre el tope de la pila P.
- P.Vacía?(P) $\rightarrow \mathbb{B}$: Dice si la pila P está vacía.
- P.Tope() $\rightarrow$ *ELEM*: Devuelve el elemento del tope de P.
  Precondición: $\neg P$.Vacía?().
- P.Desapilar(): Borra el elemento del tope de P.
  Precondición: $\neg P$.Vacía?().

donde P : Pila(ELEM), x : ELEM (entero, char, etc.).

Las pilas tienen estrategia LIFO (*last in, first out*).

Problema: Paréntesis balanceados

Dado un string, determinar si los caracteres { }, [ ], ( ) están balanceados correctamente.

Ejemplos:

- "{a(b)x[()]}" está balanceado.
- "}", "a(b))" y "[[}" no están balanceados.

¿Se les ocurre una solución? Sugerencia: usar el tipo Pila.

TAD Pila(ELEM) - Posible implementación

Estructura de representación del TAD Pila(ELEM):

$$\text{Pila(ELEM)} == \langle \textit{milista} : \text{Lista(ELEM)} \rangle$$

Invariante de representación de esta estructura:
En este caso no hay nada que decir. Según la estructura de representación elegida, cualquier lista es una representación válida de una pila.

TAD Pila(ELEM) - Posible implementación

Algoritmos de las operaciones del TAD Pila(ELEM) para la estructura de representación elegida:

CrearPila() $\rightarrow$ Pila(ELEM):
    $RV.milista \leftarrow CrearLista()$

P.Apilar(x):
    $P.milista$.Agregar(x)

P.Vacía?() $\rightarrow \mathbb{B}$:
    $RV \leftarrow (P.milista.\text{Longitud}() = 0)$

P.Tope() $\rightarrow ELEM$: (Pre: $\neg P$.Vacía?())
    $RV \leftarrow P.milista$.Ésimo($P.milista$.Longitud() $- 1$)

P.Desapilar(): (Pre: $\neg P$.Vacía?())
    $P.milista$.BorrarÉsimo($P.milista$.Longitud() $- 1$)

TAD Pila(ELEM) - Posible implementación

La complejidad temporal de los algoritmos vistos para esta estructura de representación dependen de la complejidad de las operaciones del TAD Lista. Suponiendo la implementación del TAD Lista que vimos, los órdenes son:

- CrearPila() $\rightarrow$ Pila(ELEM) $O(1)$
- P.Apilar(x) $O(n)$
- P.Vacía?() $\rightarrow \mathbb{B}$ $O(1)$
- P.Tope() $\rightarrow$ *ELEM* $O(n)$
- P.Desapilar() $O(n)$

¿Se les ocurre otra estructura de representación que nos permita bajar todos estos a O(1)?

TAD Cola(ELEM)

Operaciones:

- CrearCola() $\rightarrow$ Cola(ELEM): Crea una cola vacía.
- C.Encolar(x): Agrega el elemento x al final de la cola C.
- C.Vacía?() $\rightarrow \mathbb{B}$: Dice si la cola C está vacía.
- C.Primero() $\rightarrow$ *ELEM*: Devuelve el primer elemento de C.
  Precondición: $\neg C$.Vacía?().
- C.Desencolar(): Borra el primer elemento de C.
  Precondición: $\neg C$.Vacía?().

donde C : Cola(ELEM), x : ELEM (entero, char, etc.).

Las colas tienen estrategia FIFO (*first in, first out*).

Implementación: Muy parecida a la de Pilas. Ejercicio.

Otro ejemplo

Queremos representar conjuntos de especies animales. Podemos hacerlo con listas, por ejemplo:

- Felinos = [león, gato, tigre, guepardo, pantera, puma]
- Cánidos = [lobo, coyote, chacal, dingo, zorro]
- Cetáceos = [delfín, ballena, orca, narval, cachalote]

¿Las listas son una buena forma de representar conjuntos?

- Orden: [lobo, coyote] = [coyote, lobo] ?
- Repetidos: [delfín, delfín] ?

Mejor usar un TAD Conjunto que nos evite estos problemas.

TAD Conjunto(ELEM)

Operaciones:

- CrearConjunto() $\rightarrow$ Conjunto(ELEM): Crea un conjunto vacío.
- C.Agregar(x): Agrega el elemento x al conjunto C.
- C.Pertenece?(x) $\rightarrow \mathbb{B}$: Dice si el elemento x está en C.
- C.Eliminar(x): Elimina el elemento x de C.
- C.Tamaño() $\rightarrow \mathbb{Z}$: Devuelve la cantidad de elementos de C.
- C_1.Igual?(C_2) $\rightarrow \mathbb{B}$: Dice si los dos conjuntos son iguales.

donde C, C_1, C_2 : Conjunto(ELEM), x : ELEM.

TAD Conjunto(ELEM)

Operaciones (cont.):

- C_1.Unión(C_2) $\rightarrow$ Conjunto(ELEM): Devuelve un nuevo conjunto con $C_1 \cup C_2$.
- C_1.Intersección(C_2) $\rightarrow$ Conjunto(ELEM): Devuelve un nuevo conjunto con $C_1 \cap C_2$.
- C_1.Diferencia(C_2) $\rightarrow$ Conjunto(ELEM): Devuelve un nuevo conjunto con $C_1 \setminus C_2$.
- C.ListarElementos() $\rightarrow$ Lista(ELEM): Devuelve una lista de los elementos de C, en cualquier orden.
- C.AgregarTodos(L): Agrega todos los elementos de L en C.

donde C, C_1, C_2: Conjunto(ELEM), x: ELEM, L: Lista(ELEM).

TAD Conjunto(ELEM) - Posible implementación

Estructura de representación del TAD Conjunto(ELEM):

$$\text{Conjunto(ELEM)} == \langle ls : \text{Lista(ELEM)} \rangle$$

MOMENTO. ¿No dijimos que las listas no son buenas para representar conjuntos?

Correcto, pero acá *encapsulamos* las dificultades de representar conjuntos con listas, y el usuario del TAD Conjunto no se entera de las mismas.

Invariante de representación de esta estructura:
La lista *ls* no puede tener elementos repetidos.

TAD Conjunto(ELEM) - Posible implementación

Algoritmos de las operaciones del TAD Conjunto(ELEM) para la estructura de representación elegida:

CrearConjunto() $\rightarrow$ Conjunto(ELEM):
    $RV.ls \leftarrow$ CrearLista()

C.Pertenece?(x) $\rightarrow \mathbb{B}$:
    $RV \leftarrow (C.ls.\text{Cantidad}(x) > 0)$

C.Agregar(x):
    if $(\neg C.\text{Pertenece}(x))$ $C.ls.\text{Agregar}(x)$

C.Eliminar(x):
    if $(C.\text{Pertenece}(x))$ $C.ls.\text{Borrarlésimo}(C.ls.\text{Indice}(x))$

C.Tamaño() $\rightarrow \mathbb{Z}$:
    $RV \leftarrow C.ls.\text{Longitud}()$

TAD Conjunto(ELEM) - Posible implementación

C.ListarElementos() $\rightarrow$ Lista(ELEM):

```
RV ← CrearLista()
i ← 0
while (i < C.ls.Longitud()) {
        RV.Agregar(C.ls.Iésimo(i))
        i ← i + 1
}
```

C.AgregarTodos(L):

```
i ← 0
while (i < L.Longitud()) {
        C.Agregar(L.Iésimo(i))
        i ← i + 1
}
```

TAD Conjunto(ELEM) - Posible implementación

C_1.Unión(C_2) $\rightarrow$ Conjunto(ELEM):

$RV \leftarrow$ CrearConjunto()
RV.AgregarTodos(C_1.ListarElementos())
RV.AgregarTodos(C_2.ListarElementos())

C_1.Igual?(C_2) $\rightarrow \mathbb{B}$:

if (C_1.Tamaño() = C_2.Tamaño()) {
    $RV \leftarrow$ TRUE
    $L \leftarrow C_1$.ListarElementos()
    $i \leftarrow 0$
    while ($i < L$.Longitud()) {
        $RV \leftarrow RV$ AND C_2.Pertenece?(L.Iésimo(i))
        $i \leftarrow i + 1$
    }
} else {
    $RV \leftarrow$ FALSE
}

TAD Conjunto(ELEM) - Otra implementación

Estructura de representación del TAD Conjunto(ELEM):

$$\text{Conjunto(ELEM)} == \langle ls : \text{Lista(ELEM)} \rangle$$

Invariante de representación: True. (No ponemos ninguna restricción sobre la lista *ls*.)

¿Cómo cambian los algoritmos?

- C.Agregar(x)
- C.Pertenece?(x) $\rightarrow \mathbb{B}$
- C.Eliminar(x)
- C.Tamaño() $\rightarrow \mathbb{Z}$

Otro posible invariante: *ls* no tiene elementos repetidos y está ordenada (sup. ELEM tiene una relación de orden total).

Repaso

- Tipos Abstractos de Datos.
  - Parte pública: nombre + especificación de operaciones.
  - Parte privada: estructura, invariante, algoritmos.
- TADs Fecha, Lista(ELEM), Conjunto(ELEM), etc.

Próximos temas

- Técnicas algorítmicas:
  - Backtracking. 8 reinas.
  - (Heurísticas. Algoritmos aproximados.)