Algortimos y Estructuras de Datos

2do Cuatrimestre 2024

Complejidades de Estructuras

Rep/Abs

Elección de Estructuras

COMPLEJIDADES

CONSTRUÍR UNA TABLA COMPARATIVA CON LAS COMPLEJIDADES EN PEOR CASO DE LAS OPERACIONES COMUNES PARA CONJUNTOS SOBRE LAS SIGUIENTES ESTRUCTURAS

	Pertenece	Insertar	Borrar	Buscar Minimo	Borrar Minimo
Array					
Lista Enlazada					
ABB					
AVL					
Heap					
Trie					

COMPLEJIDADES

	Pertenece	Insertar	Borrar	Buscar Minimo	Borrar Minimo
Array	O(n)	O(1)	O(1)	O(n)	O(n)
Lista Enlazada	O(n)	O(1)	O(n)	O(n)	O(n)
ABB	O(n)	O(n)	O(n)	O(n)	O(n)
AVL	O(log n)	O(log n)	O(log n)	O(log n)	O(log n)
Heap	O(n)	O(log n)	O(n)	O(1)	O(log n)
Trie	O(|k|)	O(|k|)	O(|k|)	O(|k|)*	

EJERCICIO 2 GUIA 8

CONSIDERE LA SIGUIENTE ESPECIFICACION DE UNA RELACION UNO/MUCHOS ENTRE ALARMAS Y SENSORES DE UNA PLANTA INDUSTRIAL: UN SENSOR PUEDE ESTAR ASOCIADO A MUCHAS ALARMAS, Y UNA ALARMA PUEDE TENER MUCHOS SENSORES ASOCIADOS

```
TAD Planta {
obs alarmas: conj⟨Alarma⟩
obs sensores: conj⟨⟨Sensor, Alarma⟩⟩
proc nuevaPlanta () : Planta {
    asegura {res.alarmas = {}}
    asegura {res.sensores = {}}
}
```

```
proc agregarAlarma (inout p : Planta, in a : Alarma) {
    requiere {p = P_0}
    requiere {a ∉ p.alarmas}
    asegura {p.alarmas = P_0.alarmas ∪ {a}}
    asegura {p.sensores = P_0.sensores}
}

    proc agregarSensor (inout p : Planta, in a : Alarma, in s : Sensor) {
        requiere {p = P_0}
        requiere {a ∈ p.alarmas}
        requiere {⟨s, a⟩ ∉ p.sensores}
        asegura {p.alarmas = P_0.alarmas}
        asegura {p.sensores = P_0.sensores + {⟨s, a⟩}}
    }
}
```

SE DECIDIO UTILIZAR LA SIGUIENTE ESTRUCTURA COMO REPRESENTACION, QUE PERMITE CONSULTAR FACILMENTE TANTO EN UNA DIRECCION (SENSORES DE UNA ALARMA) COMO EN LA CONTRARIA (ALARMAS DE UN SENSOR).

```
Módulo PlantaImpl implementa Planta <
    var alarmas: Diccionario<Alarma, Conjunto<Sensor>>
    var sensores: Diccionario<Sensor, Conjunto<Alarma>>
>
```

SE PIDE:

- **ESCRIBIR FORMALMENTE Y EN CASTELLANO EL INVARIANTE DE REPRESENTACION.**
- **ESCRIBIR LA FUNCION DE ABSTRACCION.**

EJERCICIO 9 GUIA 8

LA MADERERA SAN BLAS VENDE, ENTRE OTRAS COSAS, LISTONES DE MADERA.LOS COMPRA EN ASERRADEROS DE LA ZONA, LOS CEPILLA Y ACONDICIONA, Y LOS VENDE POR MENOR DEL LARGO QUE EL CLIENTE NECESITE.
TIENEN UN SISTEMA UN POCO PARTICULAR Y CIERTAMENTE NO MUY EFICIENTE: CUANDO INGRESA UN PEDIDO, BUSCAN EL LISTON MAS LARGO QUE TIENE EN EL DEPOSITO, REALIZAN EL CORTE DEL TAMAÑO QUE EL CLIENTE PIDIO, Y DEVUELVEN EL TROZO QUE QUEDA AL DEPOSITO.
POR OTRA PARTE, IDENTIFICAN A CADA CLIENTE CON UN CODIGO ALFANUMERICO DE 10 DIGITOS Y CUENTAN CON UN FICHERO EN EL QUE REGISTRAN TODAS LAS COMPRAS QUE HIZO CADA CLIENTE (CON LA FECHA DE LA COMPRA Y EL TAMAÑO DEL LISTON VENDIDO). ESTE SERIA EL TAD SIMPLIFICADO DEL SISTEMA:

```
Cliente es string
TAD Maderera {

proc comprarUnListon (inout m: Maderera, in tamaño: ℤ) {
    // comprar en el aserradero un listón de un determinado tamaño
}

proc venderUnListon (inout m: Maderera, in tamaño: ℤ, in cli: Cliente, in f: Fecha) {
    // vender un listón de un determinado tamaño a un cliente particular en una fecha determinada
}

proc ventasACliente (in m: Maderera, in cli: Cliente) {
    // devolver el conjunto de todas las ventas que se le hicieron a un cliente
    // (para cada venta, se quiere saber la fecha y el tamaño del listón)
}
}
```

SE PIDE:
ESCRIBA UNA ESTRUCTURA QUE PERMITA REALIZAR LAS OPERACIONES INDICADAS CON LAS SIGUIENTES COMPLEJIDADES:

- comprarUnListon en O(log(m)) - venderUnListon en O(log(m)) - ventasACliente en O(1)

DONDE M ES LA CANTIDAD DE PEDAZOS DE LISTON QUE HAY EN EL DEPOSITO

ESCRIBA EL ALGORITMO PARA LA OPERACION **venderUnListon**

Terminamos!!

Momento de Consultas