Tema 16: El TAD de las colas de prioridad

Informática (2014–15)

José A. Alonso Jiménez

Grupo de Lógica Computacional
Departamento de Ciencias de la Computación e I.A.
Universidad de Sevilla

Tema 16: El TAD de las colas de prioridad

Tema 16: El TAD de las colas de prioridad

Descripción de las colas de prioridad

- Una **cola de prioridad** es una cola en la que cada elemento tiene asociada una prioridad. La operación de extracción siempre elige el elemento de menor prioridad.
- Ejemplos:
  - La cola de las ciudades ordenadas por su distancia al destino final.
  - Las colas de las tareas pendientes ordenadas por su fecha de terminación.

Signatura de las colas de prioridad

- Signatura:

```
vacia,   :: Ord a => CPrioridad a
inserta, :: Ord a => a -> CPrioridad a -> CPrioridad a
primero, :: Ord a => CPrioridad a -> a
resto,   :: Ord a => CPrioridad a -> CPrioridad a
esVacia, :: Ord a => CPrioridad a -> Bool
valida   :: Ord a => CPrioridad a -> Bool
```

- Descripción de las operaciones:
  - `vacia` es la cola de prioridad vacía.
  - `(inserta x c)` añade el elemento `x` a la cola de prioridad `c`.
  - `(primero c)` es el primer elemento de la cola de prioridad `c`.
  - `(resto c)` es el resto de la cola de prioridad `c`.
  - `(esVacia c)` se verifica si la cola de prioridad `c` es vacía.
  - `(valida c)` se verifica si `c` es una cola de prioridad válida.

Tema 16: El TAD de las colas de prioridad

Propiedades del TAD de las colas de prioridad

1. `inserta x (inserta y c) == inserta y (inserta x c)`
2. `primero (inserta x vacia) == x`
3. Si `x <= y`, entonces
   `primero (inserta y (inserta x c))`
   `== primero (inserta x c)`
4. `resto (inserta x vacia) == vacia`
5. Si `x <= y`, entonces
   `resto (inserta y (inserta x c))`
   `== inserta y (resto (inserta x c))`
6. `esVacia vacia`
7. `not (esVacia (inserta x c))`

Tema 16: El TAD de las colas de prioridad

Las colas de prioridad como listas

- Cabecera del módulo:

```
module ColaDePrioridadConListas
    (CPrioridad,
     vacia,    -- Ord a => CPrioridad a
     inserta,  -- Ord a => a -> CPrioridad a -> CPrioridad
     primero,  -- Ord a => CPrioridad a -> a
     resto,    -- Ord a => CPrioridad a -> CPrioridad a
     esVacia,  -- Ord a => CPrioridad a -> Bool
     valida    -- Ord a => CPrioridad a -> Bool
    ) where
```

- Colas de prioridad mediante listas:

```
newtype CPrioridad a = CP [a]
    deriving (Eq, Show)
```

Las colas de prioridad como listas

- Ejemplo de cola de prioridad: cp1 es la cola de prioridad obtenida añadiéndole a la cola vacía los elementos 3, 1, 7, 2 y 9.

```
cp1  ⇝ CP [1,2,3,7,9]
```

```
cp1 :: CPrioridad Int
cp1 = foldr inserta vacia [3,1,7,2,9]
```

- `(valida c)` se verifica si c es una cola de prioridad válida; es decir, está ordenada crecientemente. Por ejemplo,

```
valida (CP [1,3,5])  ⇝  True
valida (CP [1,5,3])  ⇝  False
```

```
valida :: Ord a => CPrioridad a -> Bool
valida (CP xs) = ordenada xs
    where ordenada (x:y:zs) = x <= y && ordenada (y:zs)
          ordenada _        = True
```

Las colas de prioridad como listas

- Ejemplo de cola de prioridad: cp1 es la cola de prioridad obtenida añadiéndole a la cola vacía los elementos 3, 1, 7, 2 y 9.

```
cp1  ⇝ CP [1,2,3,7,9]
```

```
cp1 :: CPrioridad Int
cp1 = foldr inserta vacia [3,1,7,2,9]
```

- `(valida c)` se verifica si c es una cola de prioridad válida; es decir, está ordenada crecientemente. Por ejemplo,

```
valida (CP [1,3,5])  ⇝  True
valida (CP [1,5,3])  ⇝  False
```

```
valida :: Ord a => CPrioridad a -> Bool
valida (CP xs) = ordenada xs
    where ordenada (x:y:zs) = x <= y && ordenada (y:zs)
          ordenada _        = True
```

Las colas de prioridad como listas

- `vacia` es la cola de prioridad vacía. Por ejemplo,

```
vacia  ⇝  CP []
```

```
vacia :: Ord a => CPrioridad a
vacia = CP []
```

- `(inserta x c)` es la cola obtenida añadiendo el elemento `x` a la cola de prioridad `c`. Por ejemplo,

```
cp1             ⇝  CP [1,2,3,7,9]
inserta 5 cp1   ⇝  CP [1,2,3,5,7,9]
```

```
inserta :: Ord a => a -> CPrioridad a -> CPrioridad a
inserta x (CP q) = CP (ins x q)
    where ins x []                    = [x]
          ins x r@(e:r') | x < e      = x:r
                         | otherwise  = e:ins x r'
```

Las colas de prioridad como listas

- `vacia` es la cola de prioridad vacía. Por ejemplo,

```
vacia  ⇝  CP []
```

```
vacia :: Ord a => CPrioridad a
vacia = CP []
```

- `(inserta x c)` es la cola obtenida añadiendo el elemento `x` a la cola de prioridad `c`. Por ejemplo,

```
cp1              ⇝  CP [1,2,3,7,9]
inserta 5 cp1    ⇝  CP [1,2,3,5,7,9]
```

```
inserta :: Ord a => a -> CPrioridad a -> CPrioridad a
inserta x (CP q) = CP (ins x q)
    where ins x []                    = [x]
          ins x r@(e:r') | x < e      = x:r
                         | otherwise  = e:ins x r'
```

Las colas de prioridad como listas

- `vacia` es la cola de prioridad vacía. Por ejemplo,

```
vacia  ⇝  CP []
```

```
vacia :: Ord a => CPrioridad a
vacia = CP []
```

- `(inserta x c)` es la cola obtenida añadiendo el elemento `x` a la cola de prioridad `c`. Por ejemplo,

```
cp1              ⇝  CP [1,2,3,7,9]
inserta 5 cp1    ⇝  CP [1,2,3,5,7,9]
```

```
inserta :: Ord a => a -> CPrioridad a -> CPrioridad a
inserta x (CP q) = CP (ins x q)
    where ins x []                      = [x]
          ins x r@(e:r') | x < e        = x:r
                         | otherwise    = e:ins x r'
```

Las colas de prioridad como listas

- `(primero c)` es el primer elemento de la cola de prioridad c.

```
cp1           ⇝  CP [1,2,3,7,9]
primero cp1   ⇝  1
```

```
primero :: Ord a => CPrioridad a -> a
primero (CP(x:_)) = x
primero _         = error "primero: cola vacia"
```

- `(resto c)` es la cola de prioridad obtenida eliminando el primer elemento de la cola de prioridad c. Por ejemplo,

```
cp1         ⇝  CP [1,2,3,7,9]
resto cp1   ⇝  CP [2,3,7,9]
```

```
resto :: Ord a => CPrioridad a -> CPrioridad a
resto (CP (_:xs)) = CP xs
resto _           = error "resto: cola vacia"
```

Las colas de prioridad como listas

- `(primero c)` es el primer elemento de la cola de prioridad c.

```
cp1           ⇝  CP [1,2,3,7,9]
primero cp1   ⇝  1
```

```
primero :: Ord a => CPrioridad a -> a
primero (CP(x:_)) = x
primero _         = error "primero: cola vacia"
```

- `(resto c)` es la cola de prioridad obtenida eliminando el primer elemento de la cola de prioridad c. Por ejemplo,

```
cp1          ⇝  CP [1,2,3,7,9]
resto cp1    ⇝  CP [2,3,7,9]
```

```
resto :: Ord a => CPrioridad a -> CPrioridad a
resto (CP (_:xs)) = CP xs
resto _           = error "resto: cola vacia"
```

Las colas de prioridad como listas

- `(primero c)` es el primer elemento de la cola de prioridad c.

```
cp1           ⇝  CP [1,2,3,7,9]
primero cp1   ⇝  1
```

```
primero :: Ord a => CPrioridad a -> a
primero (CP(x:_)) = x
primero _         = error "primero: cola vacia"
```

- `(resto c)` es la cola de prioridad obtenida eliminando el primer elemento de la cola de prioridad c. Por ejemplo,

```
cp1          ⇝  CP [1,2,3,7,9]
resto cp1    ⇝  CP [2,3,7,9]
```

```
resto :: Ord a => CPrioridad a -> CPrioridad a
resto (CP (_:xs)) = CP xs
resto _           = error "resto: cola vacia"
```

Las colas de prioridad como listas

- `(esVacia c)` se verifica si la cola de prioridad c es vacía. Por ejemplo,

```
esVacia cp1    ⇝  False
esVacia vacia  ⇝  True
```

```
esVacia :: Ord a => CPrioridad a -> Bool
esVacia (CP xs) = null xs
```

Las colas de prioridad como listas

- `(esVacia c)` se verifica si la cola de prioridad c es vacía. Por ejemplo,

```
esVacia cp1    ⇝  False
esVacia vacia  ⇝  True
```

```
esVacia :: Ord a => CPrioridad a -> Bool
esVacia (CP xs) = null xs
```

Tema 16: El TAD de las colas de prioridad

Las colas de prioridad como montículos

La implementación de las colas de prioridad como montículos (`ColaDePrioridadConMonticulos.hs`) se encuentra en en el tema 20 (El TAD de los montículos).

Tema 16: El TAD de las colas de prioridad

Importación de librerías en el módulo de comprobación

- Importación de la implementación de colas de prioridad que se desea verificar.

```
import ColaDePrioridadConListas
-- ColaDePrioridadConMonticulos.hs
```

- Importación de las librerías de comprobación

```
import Test.QuickCheck
import Test.Framework
import Test.Framework.Providers.QuickCheck2
```

Tema 16: El TAD de las colas de prioridad

1. Especificación del TAD de las colas de prioridad

2. Implementaciones del TAD de las colas de prioridad

3. Comprobación de las implementaciones con QuickCheck
   - Librerías auxiliares
   - Generador de colas de prioridad
   - Especificación de las propiedades de las colas de prioridad
   - Comprobación de las propiedades

Generador de colas de prioridad

- `genCPrioridad` es un generador de colas de prioridad. Por ejemplo,

```
ghci> sample genCPrioridad
CP [-4]
CP [-2,-1,-1,2,5]
...
```

```
genCPrioridad :: (Arbitrary a, Num a, Ord a)
                 =>  Gen (CPrioridad a)
genCPrioridad = do xs <- listOf arbitrary
                   return (foldr inserta vacia xs)

instance (Arbitrary a, Num a, Ord a)
         => Arbitrary (CPrioridad a) where
    arbitrary = genCPrioridad
```

Corrección del generador de colas de prioridad

- Las colas de prioridad producidas por `genCPrioridad` son válidas.

```
prop_genCPrioridad_correcto :: CPrioridad Int -> Bool
prop_genCPrioridad_correcto c = valida c
```

- Comprobación.

```
ghci> quickCheck prop_genCPrioridad_correcto
+++ OK, passed 100 tests.
```

Corrección del generador de colas de prioridad

- Las colas de prioridad producidas por `genCPrioridad` son válidas.

```
prop_genCPrioridad_correcto :: CPrioridad Int -> Bool
prop_genCPrioridad_correcto c = valida c
```

- Comprobación.

```
ghci> quickCheck prop_genCPrioridad_correcto
+++ OK, passed 100 tests.
```

Tema 16: El TAD de las colas de prioridad

Especificación de las propiedades de colas de prioridad

- Si se añade dos elementos a una cola de prioridad se obtiene la misma cola de prioridad idependientemente del orden en que se añadan los elementos.

```
prop_inserta_conmuta :: Int -> Int -> CPrioridad Int
                     -> Bool
prop_inserta_conmuta x y c =
    inserta x (inserta y c) == inserta y (inserta x c)
```

- La cabeza de la cola de prioridad obtenida anadiendo un elemento `x` a la cola de prioridad vacía es `x`.

```
prop_primero_inserta_vacia :: Int -> CPrioridad Int -> Boo
prop_primero_inserta_vacia x c =
    primero (inserta x vacia) == x
```

Especificación de las propiedades de colas de prioridad

- Si se añade dos elementos a una cola de prioridad se obtiene la misma cola de prioridad idependientemente del orden en que se añadan los elementos.

```
prop_inserta_conmuta :: Int -> Int -> CPrioridad Int
                        -> Bool
prop_inserta_conmuta x y c =
    inserta x (inserta y c) == inserta y (inserta x c)
```

- La cabeza de la cola de prioridad obtenida anadiendo un elemento `x` a la cola de prioridad vacía es `x`.

```
prop_primero_inserta_vacia :: Int -> CPrioridad Int -> Boo
prop_primero_inserta_vacia x c =
    primero (inserta x vacia) == x
```

Especificación de las propiedades de colas de prioridad

- Si se añade dos elementos a una cola de prioridad se obtiene la misma cola de prioridad idependientemente del orden en que se añadan los elementos.

```
prop_inserta_conmuta :: Int -> Int -> CPrioridad Int
                        -> Bool
prop_inserta_conmuta x y c =
    inserta x (inserta y c) == inserta y (inserta x c)
```

- La cabeza de la cola de prioridad obtenida anadiendo un elemento `x` a la cola de prioridad vacía es `x`.

```
prop_primero_inserta_vacia :: Int -> CPrioridad Int -> Boo
prop_primero_inserta_vacia x c =
    primero (inserta x vacia) == x
```

Especificación de las propiedades de colas de prioridad

- El primer elemento de una cola de prioridad c no cambia cuando se le añade un elemento mayor o igual que algún elemento de c.

```
prop_primero_inserta :: Int -> Int -> CPrioridad Int
                     -> Property
prop_primero_inserta x y c =
    x <= y ==>
    primero (inserta y c') == primero c'
    where c' = inserta x c
```

- El resto de añadir un elemento a la cola de prioridad vacía es la cola vacía.

```
prop_resto_inserta_vacia :: Int -> Bool
prop_resto_inserta_vacia x =
    resto (inserta x vacia) == vacia
```

Especificación de las propiedades de colas de prioridad

- El primer elemento de una cola de prioridad c no cambia cuando se le añade un elemento mayor o igual que algún elemento de c.

```
prop_primero_inserta :: Int -> Int -> CPrioridad Int
                        -> Property
prop_primero_inserta x y c =
    x <= y ==>
    primero (inserta y c') == primero c'
    where c' = inserta x c
```

- El resto de añadir un elemento a la cola de prioridad vacía es la cola vacía.

```
prop_resto_inserta_vacia :: Int -> Bool
prop_resto_inserta_vacia x =
    resto (inserta x vacia) == vacia
```

Especificación de las propiedades de colas de prioridad

- El primer elemento de una cola de prioridad c no cambia cuando se le añade un elemento mayor o igual que algún elemento de c.

```
prop_primero_inserta :: Int -> Int -> CPrioridad Int
                        -> Property
prop_primero_inserta x y c =
    x <= y ==>
    primero (inserta y c') == primero c'
    where c' = inserta x c
```

- El resto de añadir un elemento a la cola de prioridad vacía es la cola vacía.

```
prop_resto_inserta_vacia :: Int -> Bool
prop_resto_inserta_vacia x =
    resto (inserta x vacia) == vacia
```

Especificación de las propiedades de colas de prioridad

- El resto de la cola de prioridad obtenida añadiendo un elemento `y` a una cola `c'` (que tiene algún elemento menor o igual que `y`) es la cola que se obtiene añadiendo `y` al resto de `c'`.

```
prop_resto_inserta :: Int -> Int -> CPrioridad Int
                      -> Property
prop_resto_inserta x y c =
    x <= y ==>
    resto (inserta y c') == inserta y (resto c')
    where c' = inserta x c
```

- `vacia` es una cola vacía.

```
prop_vacia_es_vacia :: Bool
prop_vacia_es_vacia = esVacia (vacia :: CPrioridad Int)
```

Especificación de las propiedades de colas de prioridad

- El resto de la cola de prioridad obtenida añadiendo un elemento `y` a una cola `c'` (que tiene algún elemento menor o igual que `y`) es la cola que se obtiene añadiendo `y` al resto de `c'`.

```
prop_resto_inserta :: Int -> Int -> CPrioridad Int
                      -> Property
prop_resto_inserta x y c =
    x <= y ==>
    resto (inserta y c') == inserta y (resto c')
    where c' = inserta x c
```

- `vacia` es una cola vacía.

```
prop_vacia_es_vacia :: Bool
prop_vacia_es_vacia = esVacia (vacia :: CPrioridad Int)
```

Especificación de las propiedades de colas de prioridad

- El resto de la cola de prioridad obtenida añadiendo un elemento `y` a una cola `c'` (que tiene algún elemento menor o igual que `y`) es la cola que se obtiene añadiendo `y` al resto de `c'`.

```
prop_resto_inserta :: Int -> Int -> CPrioridad Int
                      -> Property
prop_resto_inserta x y c =
    x <= y ==>
    resto (inserta y c') == inserta y (resto c')
    where c' = inserta x c
```

- `vacia` es una cola vacía.

```
prop_vacia_es_vacia :: Bool
prop_vacia_es_vacia = esVacia (vacia :: CPrioridad Int)
```

Especificación de las propiedades de colas de prioridad

- Si se añade un elemento a una cola de prioridad se obtiene una cola no vacía.

```
prop_inserta_no_es_vacia :: Int -> CPrioridad Int
                         -> Bool
prop_inserta_no_es_vacia x c =
    not (esVacia (inserta x c))
```

Especificación de las propiedades de colas de prioridad

- Si se añade un elemento a una cola de prioridad se obtiene una cola no vacía.

```
prop_inserta_no_es_vacia :: Int -> CPrioridad Int
                            -> Bool
prop_inserta_no_es_vacia x c =
    not (esVacia (inserta x c))
```

Tema 16: El TAD de las colas de prioridad

Definición del procedimiento de comprobación

- `compruebaPropiedades` comprueba todas las propiedades con la plataforma de verificación.

```
compruebaPropiedades =
    defaultMain
        [testGroup "Corrección del generador"
          [testProperty "P0" prop_genCPrioridad_correcto],
         testGroup "Propiedade de colas de prioriad:"
          [testProperty "P1" prop_inserta_conmuta,
           testProperty "P2" prop_primero_inserta_vacia,
           testProperty "P3" prop_primero_inserta,
           testProperty "P4" prop_resto_inserta_vacia,
           testProperty "P5" prop_resto_inserta,
           testProperty "P6" prop_vacia_es_vacia,
           testProperty "P7" prop_inserta_no_es_vacia]]
```

Comprobación de las propiedades de las colas de prioridad

```
ghci> compruebaPropiedades
Corrección del generador:
  P0: [OK, passed 100 tests]
Propiedades de colas de prioridad:
  P1: [OK, passed 100 tests]
  P2: [OK, passed 100 tests]
  P3: [OK, passed 100 tests]
  P4: [OK, passed 100 tests]
  P5: [OK, passed 100 tests]
  P6: [OK, passed 100 tests]
  P7: [OK, passed 100 tests]

         Properties  Total
 Passed  8           8
 Failed  0           0
 Total   8           8
```