

Examen-febrero-2015-Haskell.pdf



Perejy



Estructuras de Datos



2º Grado en Ingeniería Informática



**Escuela Técnica Superior de Ingeniería Informática
Universidad de Málaga**

WUOLAH + BBVA

Te regalamos

15€



1/6

Este número es indicativo del riesgo del producto, siendo 1/6 indicativo de menor riesgo y 6/6 de mayor riesgo.

BBVA está adherido al Fondo de Garantía de Depósitos de Entidades de Crédito de España. La cantidad máxima garantizada es de 100.000 euros por la totalidad de los depósitos constituidos en BBVA por persona.

1

Abre tu Cuenta Online sin comisiones ni condiciones

2

Haz una compra igual o superior a 15€ con tu nueva tarjeta

3

BBVA te devuelve un máximo de 15€



1/6

Este número es indicativo del riesgo del producto, siendo 1/6 indicativo de menor riesgo y 6/6 de mayor riesgo.

BBVA está adherido al Fondo de Garantía de Depósitos de Entidades de Crédito de España. La cantidad máxima garantizada es de 100.000 euros por la totalidad de los depósitos constituidos en BBVA por persona.

1

Abre tu Cuenta Online sin comisiones ni condiciones

2

Haz una compra igual o superior a 15€ con tu nueva tarjeta

3

BBVA te devuelve un máximo de 15€

7/2/22 14:46

TwoListsDoubleEndedQueue.hs

```

1 -----
2 -- Estructuras de Datos. Grado en Informática, IS e IC. UMA.
3 -- Examen de Febrero 2015.
4 --
5 -- Implementación del TAD Deque
6 --
7 -- Apellidos:
8 -- Nombre:
9 -- Grado en Ingeniería ...
10 -- Grupo:
11 -- Número de PC:
12 -----
13
14 module TwoListsDoubleEndedQueue
15   ( DeQueue
16   , empty
17   , isEmpty
18   , first
19   , last
20   , addFirst
21   , addLast
22   , deleteFirst
23   , deleteLast
24   ) where
25
26 import Prelude hiding (last)
27 import Data.List(intercalate)
28 import Test.QuickCheck
29
30 data DeQueue a = DEQ [a] [a]
31
32 -- Complexity: O(1)
33 empty :: DeQueue a
34 empty = DEQ [] []
35
36 -- Complexity: O(1)
37 isEmpty :: DeQueue a -> Bool
38 isEmpty (DEQ [] []) = True
39 isEmpty _           = False
40
41 -- Complexity: O(1)
42 addFirst :: a -> DeQueue a -> DeQueue a
43 --addFirst x empty = DEQ [x] []
44 addFirst a (DEQ xs ys) = DEQ (a:xs) ys
45
46 -- Complexity: O(1)
47 addLast :: a -> DeQueue a -> DeQueue a
48 addLast a (DEQ xs ys) = DEQ xs (a:ys)
49
50 -- Complexity: O(1)
51 first :: DeQueue a -> a
52 first (DEQ (x:xs) ys) = x
53 first (DEQ [] ys) = head(reverse ys)
54
55 -- Complexity: O(1)
56 last :: DeQueue a -> a
57 last (DEQ xs (y:ys)) = y
58 last (DEQ xs []) = head(reverse xs)
59

```

```

60
61 -- Complexity: O(log n)
62 deleteFirst :: DEQueue a -> DEQueue a
63 deleteFirst (DEQ [] []) = empty
64 deleteFirst (DEQ (x:xs) ys) = DEQ xs ys
65 deleteFirst (DEQ [] (y:ys)) = deleteFirst (DEQ xs' ys')
66     where
67         xs' = reverse (snd (splitAt (sz) (y:ys)))
68         ys' = fst (splitAt (sz) (y:ys))
69         sz = div (length (y:ys)) 2
70
71 -- Complexity: O(log n)
72 deleteLast :: DEQueue a -> DEQueue a
73 deleteLast (DEQ [] []) = empty
74 deleteLast (DEQ xs (y:ys)) = DEQ xs ys
75 deleteLast (DEQ (x:xs) []) = deleteLast (DEQ xs' ys')
76     where
77         ys' = reverse (snd (splitAt (sz) (x:xs)))
78         xs' = fst (splitAt (sz) (x:xs))
79         sz = div (length (x:xs)) 2
80
81
82
83 instance (Show a) => Show (DEQueue a) where
84     show q = "TwoListsDoubleEndedQueue(" ++ intercalate "," [show x | x <- toList q]
85     ++ ")"
86
87 toList :: DEQueue a -> [a]
88 toList (DEQ xs ys) = xs ++ reverse ys
89
90 instance (Eq a) => Eq (DEQueue a) where
91     q == q' = toList q == toList q'
92
93 instance (Arbitrary a) => Arbitrary (DEQueue a) where
94     arbitrary = do
95         xs <- listOf arbitrary
96         ops <- listOf (oneof [return addFirst, return addLast])
97         return (foldr id empty (zipWith ($) ops xs))

```