Métodos de Ordenação

Wladimir Araújo Tavares

Universidade Federal do Ceará - Quixadá

31 de Março de 2018

Ordenação por seleção

Ordenação por Inserção

Ordenação por merge

Ordenação rápida

Ordenação por bolha

Ordenação por seleção

1. Uma lista vazia já está ordenada.
2. Para ordenar uma lista não vazia, encontre o menor elemento da lista m e coloque o menor elemento na cabeça da lista m e recursivamente ordenamos a cauda da lista removendo o menor elemento.

Ordenação por seleção

```
remove :: a ->[a] -> [a]
remove x [] = []
remove x (y:ys) | x == y    = ys
        | otherwise = y : remove x ys

ordesel :: (Ord a) => [a]-> [a]
ordsel [] = []
ordsel xs = m : ordsel (remove m xs)
        where m = minimum xs
```

Ordenação por inserção

1. a lista vazia já está ordenada;
2. para ordenar uma lista não vazia, recursivamente ordenamos a cauda e inserimos o elemento da cabeça na posição correta.

Ordenação por inserção

```
inserir :: (Ord a) => a -> [a] -> [a]
inserir x xs = takeWhile (<x) xs ++ [x] ++
  dropWhile (<x) xs

ordins :: (Ord a) => [a]-> [a]
ordins []     = []
ordins (x:xs) = inserir x (ordins xs)
```

Ordenação por merge

1. uma lista vazia ou com um só elemento já está ordenada;
2. para ordenar uma lista com dois ou mais elementos, partimos em duas metades, recursivamente ordenamos as duas parte e juntamos os resultados usando merge .

Ordenação por merge

```
merge [] ys = ys
merge xs [] = xs
merge (x:xs) (y:ys) | x <= y = x : merge xs (y:ys)
                    | otherwise = y : merge (x:xs) ys

metades xs = (take m xs,drop m xs)
        where
        m = div (length xs) 2

mergesort [] = []
mergesort [x] = [x]
mergesort xs = merge (mergesort l)( mergesort r)
        where (l,r) = metades xs
```

Ordenação rápida

1. uma lista vazia já está ordenada;
2. para ordenar uma lista não vazia, considere o primeiro elemento da lista como o elemento pivô. Particione a lista inicial em duas listas: a primeir lista contendo os elementos menores que o pivô e a segunda lista contendo os elementos maiores que o pivô. Recursivamente ordene as duas listas e concatene a lista dos elementos menores ordenada, seguida do pivô, seguida da lista dos elementos maiores ordenada.

Ordenação rápida

```
quicksort [] = []
quicksort (x:xs) = quicksort esq ++ [x] ++ quicksort dir
        where
        esq = [y | y <- xs, y < x]
        dir = [y | y <- xs, y >= x]
```

Ordenação por bolha

1. uma lista vazia já está ordenada;
2. para ordenar uma lista não vazia, realize o procedimento de trocas de elementos consecutivos se dois elementos consecutivos então na ordem errada então eles são trocados. Após esse processo, o maior elemento está na posição correta. Recursivamente ordene o ínicio da lista.

Ordenação por bolha

```
trocas [] = []
trocas [x] = [x]
trocas (x:y:xs) | x <= y = x : trocas (y:xs)
                | otherwise = y : trocas (x:xs)
bubblesort [] = []
bubblesort xs = bubblesort (init ys) ++ [last ys]
        where ys = trocas xs
```

Exercício

Implemente a função contaTrocas tal que (contaTrocas xs) devolve o número de trocas realizadas pelo algoritmo de ordenação por bolhas para ordenar a lista xs.

```
*Main> contaTrocas [5,4,3,2,1]
10
*Main> contaTrocas [6,5,4,3,2,1]
15
*Main> contaTrocas [6,5,4,3,2,1,7]
15
*Main> contaTrocas [6,5,4,3,2,1,7,3]
19
```

Exercício

```
trocas2 [] = ([], 0)
trocas2 [x] = ([x], 0)
trocas2 (x:y:xs) | x <= y = (x:l1, z1)
                 | otherwise = (y:l2, z2 + 1)
        where
        (l1, z1) = trocas2 (y:xs)
        (l2, z2) = trocas2 (x:xs)


bubblesort2 [] = ([],0)
bubblesort2 xs  = (zs ++ [last ys], n1+n2)
        where
        (ys, n1) = trocas2 xs
        (zs, n2) = bubblesort2 (init ys)

contaTrocas xs = z
        where (ys, z) = bubblesort2 xs
```

Exercício

Desenvolva função quickselect tal que (quickselect k xs) encontra o k-ésimo menor elemento de uma lista não-ordenada xs. Sua função deve ser inspirada no quicksort.