Tours

Débora Zupeli Bossois
João Olavo Baião de Vasconcelos

March 2, 2005

1 Testes de correção do módulo Binpq

Operadores: findmin delmin merge insertBinpq

1.1 Operador *findmin*

Para o teste do operador *findmin*, utilizamos um conjunto de 5 árvores de prioridade. Foram elas:
(NodePQ 2 (NodePQ 3 LeafPQ LeafPQ) (NodePQ 4 LeafPQ LeafPQ))
(NodePQ 2 (NodePQ 3 LeafPQ LeafPQ) LeafPQ)
(NodePQ 2 LeafPQ (NodePQ 4 LeafPQ LeafPQ))
(NodePQ 2 LeafPQ LeafPQ)
LeafPQ

Eis aqui os resultados para cada dado:

```
Binpq> findmin (NodePQ 2 (NodePQ 3 LeafPQ LeafPQ) (NodePQ 4 LeafPQ LeafPQ))
Just 2
Binpq> findmin (NodePQ 2 (NodePQ 3 LeafPQ LeafPQ) LeafPQ)
Just 2
Binpq> findmin (NodePQ 2 LeafPQ (NodePQ 4 LeafPQ LeafPQ))
Just 2
Binpq> findmin (NodePQ 2 LeafPQ LeafPQ)
Just 2
Binpq> findmin LeafPQ
Nothing
```

Essas árvores cobrem todas as possibilidades de parâmetros que a função pode receber, e todas as respostas obtidas foram as esperadas.

1.2 Operador *delmin*

As mesmas opções de parâmetros utilizadas no operador acima também cobrem todas as possibilidades de parâmetros do operador *delmin*. Vamos aos testes:

```
*Binpq> delmin (NodePQ 2 (NodePQ 3 LeafPQ LeafPQ) (NodePQ 4 LeafPQ LeafPQ))
NodePQ 3 (NodePQ 4 LeafPQ LeafPQ) LeafPQ
*Binpq> delmin (NodePQ 2 (NodePQ 3 LeafPQ LeafPQ) LeafPQ)
NodePQ 3 LeafPQ LeafPQ
*Binpq> delmin (NodePQ 2 LeafPQ (NodePQ 4 LeafPQ LeafPQ))
NodePQ 4 LeafPQ LeafPQ
*Binpq> delmin (NodePQ 2 LeafPQ LeafPQ)
LeafPQ
*Binpq> delmin LeafPQ
LeafPQ
```

Obtemos os resultados corretos.

1.3 Operador *merge*

Aplicamos as diversas variações possíves para *merge* e, como visto no resultado abaixo, as operações foram concluídas com êxito.

```
*Binpq> merge LeafPQ LeafPQ
LeafPQ
*Binpq> merge LeafPQ (NodePQ 2 LeafPQ LeafPQ)
NodePQ 2 LeafPQ LeafPQ
*Binpq> merge (NodePQ 2 LeafPQ LeafPQ) LeafPQ
NodePQ 2 LeafPQ LeafPQ
*Binpq> merge (NodePQ 2 LeafPQ LeafPQ) (NodePQ 10 LeafPQ LeafPQ)
NodePQ 2 (NodePQ 10 LeafPQ LeafPQ) LeafPQ
*Binpq> merge (NodePQ 10 LeafPQ LeafPQ) (NodePQ 2 LeafPQ LeafPQ)
NodePQ 2 (NodePQ 10 LeafPQ LeafPQ) LeafPQ
*Binpq> merge (NodePQ 2 (NodePQ 3 LeafPQ LeafPQ) LeafPQ) (NodePQ 10 LeafPQ LeafPQ)
NodePQ 2 (NodePQ 10 LeafPQ LeafPQ) (NodePQ 3 LeafPQ LeafPQ)
*Binpq> merge (NodePQ 2 LeafPQ (NodePQ 3 LeafPQ LeafPQ)) (NodePQ 10 LeafPQ LeafPQ)
NodePQ 2 (NodePQ 3 (NodePQ 10 LeafPQ LeafPQ) LeafPQ) LeafPQ
*Binpq> merge (NodePQ 2 (NodePQ 3 LeafPQ LeafPQ) (NodePQ 4 LeafPQ LeafPQ)) (NodePQ 10 LeafPQ LeafPQ)
NodePQ 2 (NodePQ 4 (NodePQ 10 LeafPQ LeafPQ) LeafPQ) (NodePQ 3 LeafPQ LeafPQ)
*Binpq> merge (NodePQ 10 (NodePQ 13 LeafPQ LeafPQ) LeafPQ) (NodePQ 2 LeafPQ LeafPQ)
NodePQ 2 (NodePQ 10 (NodePQ 13 LeafPQ LeafPQ) LeafPQ) LeafPQ
*Binpq> merge (NodePQ 10 LeafPQ (NodePQ 13 LeafPQ LeafPQ)) (NodePQ 2 LeafPQ LeafPQ)
NodePQ 2 (NodePQ 10 LeafPQ (NodePQ 13 LeafPQ LeafPQ)) LeafPQ
*Binpq> merge (NodePQ 10 (NodePQ 13 LeafPQ LeafPQ) (NodePQ 14 LeafPQ LeafPQ)) (NodePQ 2 LeafPQ LeafPQ)
NodePQ 2 (NodePQ 10 (NodePQ 13 LeafPQ LeafPQ) (NodePQ 14 LeafPQ LeafPQ)) LeafPQ
*Binpq> merge (NodePQ 2 LeafPQ LeafPQ) (NodePQ 10 (NodePQ 3 LeafPQ LeafPQ) LeafPQ)
NodePQ 2 (NodePQ 10 (NodePQ 3 LeafPQ LeafPQ) LeafPQ) LeafPQ
*Binpq> merge (NodePQ 2 LeafPQ LeafPQ) (NodePQ 10 LeafPQ (NodePQ 3 LeafPQ LeafPQ))
NodePQ 2 (NodePQ 10 LeafPQ (NodePQ 3 LeafPQ LeafPQ)) LeafPQ
*Binpq> merge (NodePQ 2 LeafPQ LeafPQ) (NodePQ 10 (NodePQ 3 LeafPQ LeafPQ) (NodePQ 4 LeafPQ LeafPQ))
NodePQ 2 (NodePQ 10 (NodePQ 3 LeafPQ LeafPQ) (NodePQ 4 LeafPQ LeafPQ)) LeafPQ
*Binpq> merge (NodePQ 10 LeafPQ LeafPQ) (NodePQ 2 (NodePQ 3 LeafPQ LeafPQ) LeafPQ)
```

```
NodePQ 2 (NodePQ 10 LeafPQ LeafPQ) (NodePQ 3 LeafPQ LeafPQ)
*Binpq> merge (NodePQ 10 LeafPQ LeafPQ) (NodePQ 2 LeafPQ (NodePQ 3 LeafPQ LeafPQ))
NodePQ 2 (NodePQ 3 (NodePQ 10 LeafPQ LeafPQ) LeafPQ) LeafPQ
*Binpq> merge (NodePQ 10 LeafPQ LeafPQ) (NodePQ 2 (NodePQ 3 LeafPQ LeafPQ) (NodePQ 4 LeafPQ LeafPQ))
NodePQ 2 (NodePQ 4 (NodePQ 10 LeafPQ LeafPQ) LeafPQ) (NodePQ 3 LeafPQ LeafPQ)
*Binpq> merge (NodePQ 2 (NodePQ 3 LeafPQ LeafPQ) (NodePQ 4 LeafPQ LeafPQ)) (NodePQ 10 (NodePQ 11 LeafPQ LeafPQ) (NodePQ 12 LeafPQ LeafPQ))
NodePQ 2 (NodePQ 4 (NodePQ 10 (NodePQ 11 LeafPQ LeafPQ) (NodePQ 12 LeafPQ LeafPQ)) LeafPQ) (NodePQ 3 LeafPQ LeafPQ)
*Binpq> merge (NodePQ 10 (NodePQ 13 LeafPQ LeafPQ) (NodePQ 14 LeafPQ LeafPQ)) (NodePQ 2 (NodePQ 11 LeafPQ LeafPQ) (NodePQ 12 LeafPQ LeafPQ))
NodePQ 2 (NodePQ 10 (NodePQ 12 (NodePQ 14 LeafPQ LeafPQ) LeafPQ) (NodePQ 13 LeafPQ LeafPQ)) (NodePQ 11 LeafPQ LeafPQ)
```

1.4 Operador *insertBinpq*

Este operador insere um elemento em uma árvore de prioridade. Seguindo o algoritmo, um exemplo as possibilidades de parâmetros são:

LeafPQ 3 (NodePQ 3 LeafPQ LeafPQ) 10 (NodePQ 3 (NodePQ 4 LeafPQ LeafPQ) LeafPQ) 10 (NodePQ 3 LeafPQ (NodePQ 4 LeafPQ LeafPQ) 10 (NodePQ 3 (NodePQ 4 LeafPQ LeafPQ) (NodePQ 5 LeafPQ LeafPQ)) 10

Veja os resultados obtidos a partir dessas entradas:

```
*Binpq> insertBinpq LeafPQ  3
NodePQ 3 LeafPQ LeafPQ
*Binpq> insertBinpq (NodePQ 3 LeafPQ LeafPQ) 10
NodePQ 3 (NodePQ 10 LeafPQ LeafPQ) LeafPQ
*Binpq> insertBinpq (NodePQ 3 (NodePQ 4 LeafPQ LeafPQ) LeafPQ) 10
NodePQ 3 (NodePQ 10 LeafPQ LeafPQ) (NodePQ 4 LeafPQ LeafPQ)
*Binpq> insertBinpq (NodePQ 3 LeafPQ (NodePQ 4 LeafPQ LeafPQ)) 10
NodePQ 3 (NodePQ 4 (NodePQ 10 LeafPQ LeafPQ) LeafPQ) LeafPQ
*Binpq> insertBinpq (NodePQ 3 (NodePQ 4 LeafPQ LeafPQ) (NodePQ 5 LeafPQ LeafPQ)) 10
NodePQ 3 (NodePQ 5 (NodePQ 10 LeafPQ LeafPQ) LeafPQ) (NodePQ 4 LeafPQ LeafPQ)
```

Como já foi testado anteriormente a correção do *merge*, a utilização desses parâmetros cerca todas as possibilidades de variação das entradas. E como os resultados obtidos foram os esperados, podemos presumir que o algoritmo está correto.