Árvores de pesquisa

Nos exercícios seguintes considere a definição do tipo de árvores de pesquisa apresentada na aulas teórica:

$$data\ Arv\ a = Vazia\ \mid\ No\ a\ (Arv\ a)\ (Arv\ a)$$

4.1 Escreva uma definição recursiva da função $sumArv :: Num\ a \Rightarrow Arv\ a \rightarrow a$ que soma todos os valores duma árvore binária de números.

4.2 Baseado-se na função $listar :: Arv\ a \rightarrow [a]$ apresentada na aula teórica, escreva a definição duma função para listar os elementos duma árvore de pesquisa por *ordem decrescente*.

4.3 Escreva uma definição da função $nivel :: Int \rightarrow Arv\ a \rightarrow [a]$ tal que $nivel\ n\ arv$ é a lista ordenada dos valores da árvore no nível n, isto é, a uma altura n (considerando que a raiz tem altura 0).

4.4 Experimente usar o interpretador de Haskell para calcular a altura de árvores de pesquisa com n valores.

(a) usando o método de partições binárias, e.g. *construir* [1..n];

(b) usando inserções simples, e.g. *foldr inserir Vazia* [1..n];

Experimente com $n = 10$, 100 e 1000 e compare a altura obtida com o minorante teórico: uma árvore binária com n nós tem altura $\geq \log_2 n$.

4.5 Escreva uma definição da função de ordem superior $mapArv :: (a \rightarrow b) \rightarrow Arv\ a \rightarrow Arv\ b$ tal que '$mapArv\ f\ t$' aplica uma função f a cada valor duma árvore t.

4.6 Neste exercício pretende-se implementar uma variante da remoção de um valor duma árvore de pesquisa simples.

(a) Baseando-se na função $mais\_esq :: Arv\ a \rightarrow a$ apresentada na aula teórica, escreva uma definição da função $mais\_dir :: Arv\ a \rightarrow a$ que obtém o valor mais à direita numa árvore (i.e., o maior valor).

(b) Usando a função da alínea anterior, escreva uma definição alternativa da função $remover :: Ord\ a \Rightarrow a \rightarrow Arv\ a \rightarrow Arv\ a$ que use o valor mais à direita da sub-árvore esquerda no caso de um nó com dois descentes não-vazios.

```
------
? a
-a-a-a
? e
Não ocorre
-a-a-a
? b
ba-a-a
? n
banana
Adivinhou em 4 tentativas
```

Figura 1: Exemplo de interacção do jogo de adivinha.

Programas interactivos

4.7 Escreva um programa completo que lê linhas de texto da entrada-padrão e imprime cada linha invertida.

4.8 Escreva uma função interactiva *adivinha* :: *String* → *IO* () que implemente um jogo de advinha duma palavra secreta dada como argumento pelo utilizador; um outro jogador vai tentar adivinhá-la.

O programa deve mostrar a palavra, substituindo as letras desconhecidas por traços e pedir uma nova letra; todas as ocorrências dessa letra na palavra devem então ser reveladas. O jogo termina quando o jogador adivinha a palavra; o programa deve então imprimir o número de tentativas (ver Figura 1).

4.9 Escreva uma função *elefantes* :: *Int* → () tal que, por exemplo, *elefantes* 5 imprime os seguintes versos:

> Se 2 elefantes incomodam muita gente,
> 3 elefantes incomodam muito mais!
> Se 3 elefantes incomodam muita gente,
> 4 elefantes incomodam muito mais!
> Se 4 elefantes incomodam muita gente,
> 5 elefantes incomodam muito mais!

Sugestão: utilize a função *show* :: *Show* $a \Rightarrow a \rightarrow$ *String* para converter um inteiro numa cadeia de caracteres; pode ainda re-utilizar a função *sequence_* :: $[a] \rightarrow$ () para executar uma lista de ações.

4.10 O jogo *Nim* desenrola-se com cinco filas de peças idênticas (representadas por estrelas), cujo estado inicial é o seguinte:

```
1 :  * * * * *
2 :  * * **
3 :  * * *
4 :  **
5 :  *
```

Dois jogadores vão alternadamente retirar uma ou mais estrelas de uma das filas; ganha o jogador que remover a última estrela ou grupo de estrelas.

Implemente este jogo um programa em Haskell que pergunte as jogadas de cada jogador e actualize o tabuleiro. Sugestão: represente estado do jogo como uma lista com o número de estrelas em cada fila; o estado inicial será então $[5, 4, 3, 2, 1]$.

4.11 Escreva um programa completo que codifique a entrada-padrão usando a cifra de César de 13 posições (ver https://en.wikipedia.org/wiki/ROT13 e http://www.rot13.com). Exemplo:

```
$ echo "a maria tinha um cordeirinho" | ./rot13
n znevn gvaun hz pbeqrvevaub
```