Programowanie Funkcyjne 2018

Lista zadań nr 2

17 października 2018

Uwaga: W rozwiązaniach poniższych zadań *nie wolno* używać funkcji bibliotecznych z modułu `List`.

Zadanie 1 (2p). Zdefiniuj funkcję `sublists` znajdującą wszystkie podlisty (rozumiane jako podciągi, niekoniecznie kolejnych elementów) listy zadanej jako argument. Zadbaj o to by Twoja funkcja nie generowała nieużytków.

Zadanie 2 (3p). Zdefiniuj funkcję `cycle: 'a list -> int -> 'a list`, która w cykliczny sposób przesuwa listę o zadaną liczbę pozycji `n`, przy czym zakładamy, że długość listy jest nie mniejsza niż `n`. Na przykład:

```
cycle [1; 2; 3; 4] 3 = [2; 3; 4; 1]
```

Zadanie 3 (6p). Sortowanie przez scalanie.

1. Zdefiniuj funkcję `merge`, która łączy dwie listy posortowane rosnąco w pewnym porządku $\leq$ tak, by wynik działania funkcji był także listą posortowaną rosnąco w tym samym porządku. Argumentami funkcji `merge` powinny być: funkcja `cmp: 'a -> 'a -> bool` (przy czym zakładamy, że `cmp a b = true` wtw `a` $\leq$ `b`) i dwie listy elementów typu `'a`. Na przykład

   ```
   merge (<=) [1; 2; 5] [3; 4; 5] = [1; 2; 3; 4; 5; 5]
   ```

2. Zapisz tę samą funkcję używając rekursji ogonowej, a następnie porównaj działanie obu funkcji na odpowiednich przykładach.

3. Wykorzystaj funkcję `merge` w wersji ogonowej do napisania funkcji `mergesort` sortującej listę przez scalanie.

4. Zaproponuj alternatywną implementację algorytmu sortowania przez scalanie, w której funkcja `merge` jest ogonowa, ale nie wykonuje odwracania list.[1] Nie przejmuj się, jeżeli otrzymasz algorytm sortowania, który nie jest stabilny. Porównaj szybkość działania tej implementacji z definicją z poprzedniego punktu.

Zadanie 4 (4p). Sortowanie szybkie.

1. Zdefiniuj funkcję `partition: ('a -> bool) -> 'a list -> 'a list * 'a list`, która dzieli listę na dwie listy, zawierające odpowiednio elementy spełniające zadany predykat i niespełniające go. Czy Twoja definicja zachowuje kolejność elementów, jest ogonowa, generuje nieużytki?

2. Używając funkcji `partition` zdefiniuj procedurę `quicksort` implementującą sortowanie szybkie względem zadanego porządku (reprezentowanego funkcją typu `'a -> 'a -> bool`).

Zadanie 5 (3p). Zdefiniuj funkcję zwracającą listę wszystkich permutacji zadanej listy.

Zadanie 6 (2p). Zdefiniuj funkcję generującą wszystkie sufiksy danej listy. Na przykład dla listy `[1; 2; 3]` Twoja funkcja powinna zwrócić listę `[[1; 2; 3]; [2; 3]; [3]; []]`. Następnie, zdefiniuj funkcję generującą wszystkie prefiksy danej listy. Na przykład dla listy `[1; 2; 3]` Twoja funkcja powinna zwrócić listę `[[]; [1]; [1; 2]; [1; 2; 3]]`.

[1] Taka funkcja `merge` oblicza inny wynik: zacznij od *wyspecyfikowania* działania tej funkcji, a następnie zastanów się jak przy jej pomocy posortować listę.