

Paradygmaty programowania - ćwiczenia

Lista 4

- Podaj (i wyjaśnij!) typy poniższych funkcji (samodzielnie, bez pomocy kompilatora OCaml!):
 a) `let f1 x y z = x y z;;` b) `let f2 x y = function z -> x::y;;`

- (OCaml) Napisz dowolną funkcję `f: 'a -> 'b`.

W poniższych zadaniach funkcje należy napisać w obu językach: OCaml i Scala (wykorzystując mechanizm dopasowania do wzorców!).

Na wykładzie zostały zdefiniowane drzewa binarne i grafy.

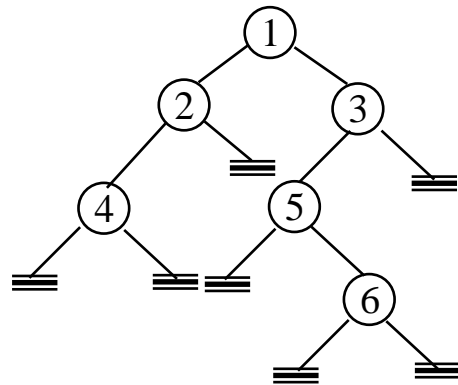
OCaml: `type 'a bt = Empty | Node of 'a * 'a bt * 'a bt`
`type 'a graph = Graph of ('a -> 'a list)`

- Dla drzew binarnych, zdefiniowanych na wykładzie, napisz funkcję `breadthBT: 'a bt -> 'a list` obchodzącą drzewo wszerz i zwracającą zawartość wszystkich węzłów drzewa w postaci listy.

Np. dla poniższego drzewa `tt`

`breadthBT tt => [1; 2; 3; 4; 5; 6]`

```
let tt = Node(1,
  Node(2,
    Node(4,
      Empty,
      Empty
    ),
    Empty
  ),
  Node(3,
    Node(5,
      Empty,
      Node(6,
        Empty,
        Empty
      )
    ),
    Empty
  )
);;
```



- W *regularnym drzewie binarnym* każdy z węzłów jest bądź liściem, bądź ma stopień dwa (patrz Cormen i in. §5.5.3). Zauważ, że drzewa `'a bt` są drzewami regularnymi – traktujemy konstruktor `Empty` jako liść.

Długość ścieżki wewnętrznej i regularnego drzewa binarnego jest sumą, po wszystkich węzłach wewnętrznych drzewa, głębokości każdego węzła. Długość ścieżki zewnętrznej e jest sumą, po wszystkich liściach drzewa, głębokości każdego liścia. Głębokość węzła definiujemy jako liczbę krawędzi od korzenia do tego węzła.

Napisz dwie funkcje, obliczające odpowiednio

- długość ścieżki wewnętrznej
- długość ścieżki zewnętrznej

zadanego regularnego drzewa binarnego.

Zauważ, że dla regularnych drzew binarnych o n węzłach wewnętrznych zachodzi $e = i + 2n$, np. dla powyższego drzewa `tt` $n = 6, i = 9, e = 21$. Czy potrafisz to udowodnić?

- Dla grafów, zdefiniowanych na wykładzie, napisz funkcję `depthSearch: 'a graph -> 'a -> 'a list`, obchodzącą graf w głąb zaczynając od zadanego wierzchołka i zwracającą zawartość zwizytowanych węzłów w postaci listy. Np. dla grafu `g` z wykładu:

`depthSearch g 4 => [4; 0; 3; 2; 1]`