Leniwa ewaluacja i memoizacja

Memoizacja

Terminem *memoizacja* określa się technikę optymalizacji polegającą na przechowywaniu wyników kosztownych obliczeniowo wywołań funkcji i zwrócenie przechowywanych, wcześniej obliczonych wartości, gdy nastąpi ponowne wywołanie funkcji z tym samym wejściem. Do przechowywania wyników można wykorzystać np. **tablice haszujące (OCaml, Scala).**

Zadania 1 i 2 należy wykonać w wybranym języku programowania: OCaml lub Scala.

Zadanie 1

a) Napisz rekurencyjną, binarną funkcję `stirling` obliczającą liczby Stirlinga drugiego rodzaju. Liczby te określają liczbę podziałów n-elementowego zbioru na m niepustych zbiorów. Funkcję tę można wyrazić następującym rekurencyjnym wzorem:

$$S(n, m) = S(n-1, m-1) + m \cdot S(n-1, m)$$

Dodatkowo, należy pamiętać, że S(0,0) = 1, S(n,0)=0 oraz dla dowolnego n $S(n, 1) = 1$ oraz $S(n, m) = 1$ dla $n = m$.

b) Następnie, skonstruuj funkcję `memoized_stirling`, która wykorzystuje technikę memoizacji do zoptymalizowania rekurencyjnych wywołań funkcji. W tym celu zadbaj, aby dla dowolnej pary argumentów (n,m) funkcja wywołała się tylko raz, a wyniki zostały zapamiętane i w przypadku kolejnego wywołania pozyskane z pamięci.

Zadanie 2

Zdefiniuj funkcję `make_memoize`. Niech funkcja ta:

- przyjmuje dowolną (czystą) funkcję fun jako argument,
- zwraca funkcję, która będzie działać dokładnie tak samo jak fun, z tą różnicą, że będzie wywoływać oryginalną funkcję fun jeden raz dla danego argumentu,

przechowa wewnątrz wynik, a następnie będzie zwracać przechowany wynik za każdym razem, kiedy zostanie wywołana z tym samym argumentem.

Przetestuj funkcję na dowolnej, napisanej przez siebie funkcji o dużej złożoności obliczeniowej.

Leniwa ewaluacja

Leniwa ewaluacja (*call by need*) jest strategią, w ramach której wyrażenie nie jest ewaluowane do momentu pierwszego użycia (tzn. jest postponowane do momentu, kiedy nastąpi potrzeba zażądania wartości). W językach Scala i OCaml domyślnym sposobem jest zachłanna ewaluacja, ale przy użyciu słowa kluczowego *lazy* można ją odroczyć.

Zadanie 3.

Pokaż, jak działa leniwa ewaluacja w Scala lub OCaml. W przypadku OCaml, wykorzystaj moduł Lazy. W tym celu:

- w środowisku REPL utwórz leniwą zmienną i przypisz do niej wartość poprzez wywołanie kosztownej funkcji (np. niezmemoizowaną wersję funkcji `stirling`),
- wypisz wartość zmiennej na ekran i pokaż, że ewaluacja następuje w sposób odroczony.