Rekurencja

11 października 2016

1 Rekurencja

Rekurencja polega na rozbiciu problemu na coraz mniejsze podproblemu, aż do momentu napotkania problemu który może być rozwiązany w sposób trywialny (przypadek bazowy). Zazwyczaj w takim wypadku korzysta się z funkcji wywołującej samą siebie. Rozwiązania rekurencyjne okazują mogą być prostsze do zaimplementowania od tych oferowanych przez inne metody. Przykład obliczanie silni: Algorytmy korzystające z rekurencji muszą spełniać trzy prawa:

Algorithm 1 Silnia

```
1: function FACTORIAL(n)
2:     if n==1 then
3:         return 1
4:     end if
5:     return n*FACTORIAL(n-1)
```

1. Algorytm musi mieć przypadek bazowy (dla silni przypadek taki ma miejsce gdy n=1)
2. Algorytm musi zmieniać swój stan i poruszać się w kierunku przypadku podstawowego (dla silni w każdej instancji następuje przemnożenie przez n)
3. Algorytm musi wywoływać samego siebie (dla silni w każdej instancji następuje wywołanie algorytmy dla n-1)

2 Liczby Lucasa

Liczby Lucasa są zdefiniowane rekurencyjnie jako

$$L_0 := 2 \ (n = 0) \tag{1}$$
$$L_1 := 1 \ (n = 1) \tag{2}$$
$$L_n := L_{n-1} + L_{n-2} \ (n > 1) \tag{3}$$

3 Labirynt

Rekurencją można posłużyć się również w celu odnalezienia wyjścia z labiryntu. Załóżmy że labirynt jest zadany przez 2-wymiarową tablicę (x,y). Poruszający się po labiryncie ma 4 możliwości:

- pójść w górę (x+1)
- pość w dół (x-1)
- pójść w lewo (y+1)
- pość w prawo (y-1)

Poruszając się po labiryncie nie można wchodzić na przeszkody ani wychodzić poza jego obszar. Rekurencyjny algorytm znajdujący wyjście można schematycznie zapisać następująco (zobacz pseudokod algorytmu 2):
Zadania

1. Zaimplementować rekurencyjną procedurę obliczającą liczby Lucasa. Za poprawną implementację można otrzymać 3 punkty.

Algorithm 2 Ścieżka

```
1: function ŚCIEŻKA(x,y)
2:     if x,y poza obszarem labiryntu then return false
3:     end if
4:     if x,y to współrzędne przeszkody lub ścieżka była eksplorowana then return false
5:     end if
6:     if x,y to współrzędne wyjścia then return true
7:     end if
8:     Oznacz pole x,y jako sprawdzone
9:     if Ścieżka(x+1,y) == true then droga[x,y] = true
        return true
10:    end if
11:    if Ścieżka(x-1,y) == true then droga[x,y] = true
        return true
12:    end if
13:    if Ścieżka(x,y-1) == true then droga[x,y] = true
        return true
14:    end if
15:    if Ścieżka(x,y+1) == true then droga[x,y] = true
        return true
16:    end ifreturn False
```

2. Zaimplementować rekurencyjny algorytm odnajdujący drogę w labiryncie. Wejściem powinna być tablica dwuwymiarowa, określająca przeszkody w labiryncie. Można przyjąć, że wyjście z labiryntu zawsze znajduje się w komórce [0,0]. Na koniec program powinien wyświetlić tablicę zawierającą zbadaną przez algorytm scieżkę w labiryncie. Za poprawną implementację można otrzymać 7 punktów.