Lista 1: „Self-avoiding walk”

Introduction to Programming in Java Roberta Sedgewicka i Kevina Wayne ma na stronie 109 następujący program:

```
   program SelfAvoidingWalk;
var
   N, T : Integer;
   DeadEnds : Integer;
   walk, x, y : Integer;
   A : array of array of Boolean;
   ParseCode : Word;
   DeadEnd : Boolean;
   r : Real;
begin
   Val (ParamStr (1), N, ParseCode); if ParseCode <> 0 then exit;
   Val (ParamStr (2), T, ParseCode); if ParseCode <> 0 then exit;
   SetLength (A, N, N);
   DeadEnds := 0;
   for walk := 1 to T do
   begin
      for x := 0 to N-1 do
         for y := 0 to N-1 do
            A[x,y] := false;
      x := N div 2; y := N div 2;
      DeadEnd := false;
      while not DeadEnd
            and (x > 0) and (x < N-1) and (y > 0) and (y < N-1) do
      begin
         A[x,y] := true;
         {Check for dead end and make a random move.}
         if A[x-1,y] and A[x+1,y] and A[x,y-1] and A[x,y+1] then
         begin
            Inc (DeadEnds); DeadEnd := true
         end else begin
            r := random;
            if      r < 0.25 then begin if not A[x+1,y] then Inc(x); end
            else if r < 0.50 then begin if not A[x-1,y] then Dec(x); end
            else if r < 0.75 then begin if not A[x,y+1] then Inc(y); end
            else if r < 1.00 then begin if not A[x,y-1] then Dec(y); end
         end
      end
   end;
   WriteLn ((100*DeadEnds) div T, '% dead ends')
end.
```

Zrób dwa spośród następujących zadań.

Ćwiczenie 1. (Ćw. 1.4.18 w książce.) Zmodyfikuj `SelfAvoidingWalk` tak by liczył i wypisywał średnią długość ścieżek. Podawaj osobno średnią długość ścieżek-ucieczek, i ścieżek do ślepych uliczek.

Ćwiczenie 2. (Ćw. 1.4.19 w książce.) Zmodyfikuj `SelfAvoidingWalk` tak by liczył i wypisywał średnie pole najmniejszego zorientowanego wzdłuż osi prostokąta zawierającego ścieżkę spaceru losowego.

Ćwiczenie 3. Rozszerz `SelfAvoidingWalk` tak by drukował, na standardowym wyjściu, zliczenia odwiedzin pól. Użyj „ASCII art": jeśli komórka, z pominięciem komórki startowej, była odwiedzona przez maksymalnie K spacerów, podziel zakres $0..K$ na kilka przedziałów i zaprezentuj je znakami o rosnącej „gęstości", np. `.,o,x,#`. Pamiętaj aby nie zliczać pojedynczego wejścia na pole wielokrotnie. Zamiast używać instrukcji warunkowych, policz znak kodujący gęstość przez zaindeksowanie „tablicy kodowej".

Ćwiczenie 4. (Ćw. 1.4.31 w książce.) Długość „bezprzecięciowych" spacerów losowych. Załóżmy że nie ma ograniczenia na wielkość planszy. Przeprowadź eksperymenty szacujące średnią długość spaceru. Jeden pomysł to dynamicznie allokować coraz większe plansze gdy mniejsze okazują się za małe. Możesz też zaallokować statycznie tablicę która wydaje się dostatecznie duża, ale uważaj wtedy na tzw. *selection bias*.

Ćwiczenie 5. (Ćw. 1.4.32 w książce.) Trójwymiarowy „bezprzecięciowy" spacer losowy. Przeprowadź eksperymenty żeby zweryfikować że prawdopodobieństwo ślepych uliczek jest 0 dla trójwymiarowych spacerów losowych i żeby policzyć średnią długość spacerów dla różnych rozmiarów N sześcianu.

Ćwiczenie 6. (Ćw. 1.4.33 w książce.) Błądzący spacerowicze. (Zwróć uwagę, że to zadanie nie opiera się na programie z przykładu `SelfAvoidingRandomWalk`.) Weźmy N błądzących spacerowiczów, startujących w środku planszy N-na-N, którzy robią kroki jednocześnie, wybierając sąsiadujące pole na prawo, lewo, powyżej, poniżej z jednakowym prawdopodobieństwem (nie „unikają przecięć"). Napisz program pomagający sformułować i przetestować hipotezę o ilości kroków potrzebnych zanim wszystkie pola zostaną odwiedzone.