Programowanie I R

Projekt I: Automat komórkowy.

Bartłomiej Zglinicki

1. Wprowadzenie

Automat komórkowy to, w nieformalnym ujęciu, siatka komórek – prostych obiektów charakteryzowanych przez stan, którego możliwe wartości tworzą zbiór skończony. Komórki rozmieszczone są jedna obok drugiej w przestrzeni ustalonego wymiaru, każdej z nich można więc przypisać skończony zbiór jej sąsiadów, definiowany na różne sposoby (np. jako zbiór komórek bezpośrednio graniczących z daną komórką). Komórki zmieniają swój stan regularnie, w określonych odstępach czasu, wszystkie jednocześnie. Nowy stan każdej z komórek zależy wyłącznie od jej dotychczasowego stanu oraz dotychczasowego stanu jej sąsiadów i jest obliczany przez z góry określoną funkcję, taką samą dla każdej komórki.

Te niezwykle proste modele znalazły szerokie zastosowanie w wielu dziedzinach nauki, w tym w fizyce – wykorzystuje się je m.in. w fizyce statystycznej i fizyce materii skondensowanej, do badania dynamiki płynów i przejść fazowych. Automaty komórkowe odgrywają również ogromną rolę w wielu gałęziach informatyki. Wystarczy tylko wspomnieć, że niektóre z nich, w szczególności chyba najsłynniejszy automat komórkowy – zaprojektowana przez angielskiego matematyka Johna Conwaya „Gra w życie", są zupełne w sensie Turinga, co oznacza, że mogą wykonać każdy algorytm i są *de facto* pełnoprawnymi komputerami.

Rozważmy automat komórkowy nazywany elementarnym. Określa się w ten sposób automat jednowymiarowy, złożony ze skończonej liczby komórek, z których każda może przyjmować jeden z dwóch stanów. Sąsiedztwo każdej z komórek tworzą dwie komórki znajdujące się tuż obok niej; przyjmiemy też, że – mówiąc nieformalnie – automat tworzy okrąg, co znaczy, że pierwsza i ostatnia z jego komórek są uważane za sąsiadujące. Ilustracja przykładowego automatu tego typu, zbudowanego z dziewięciu komórek, znajduje się na poniższym rysunku. Stan komórki oznaczono na nim kolorem: jednemu z możliwych stanów komórki przyporządkowano barwę białą, drugiemu zaś – czarną.

Stan automatu jest określony przez stany poszczególnych jego komórek. Jeśli jednemu z możliwych stanów komórki przyporządkujemy cyfrę 0, zaś drugiemu – cyfrę 1, stan automatu możemy opisać jako ciąg zer i jedynek. Dla przykładu, automat z rysunku jest w stanie, który można opisać jako 000101110.

Potrafimy już określić stan automatu, zajmiemy się więc teraz opisem jego ewolucji. Nowy stan każdej z komórek zależy od jej dotychczasowego stanu oraz od dotychczasowego stanu dwóch komórek znajdujących się tuż obok niej, z jej lewej i prawej strony (tak właśnie zdefiniowaliśmy sąsiedztwo komórki). Niech C będzie stanem analizowanej komórki, L – stanem komórki znajdującej się z jej lewej strony, zaś R – stanem komórki znajdującej się z jej prawej strony. Nowy, osiągany w kolejnym kroku ewolucji stan analizowanej komórki to wówczas $f(L, C, R)$, gdzie funkcja ewolucji

$$\{0, 1\}^3 \ni (L, C, R) \longmapsto f(L, C, R) \in \{0, 1\}.$$

Skoro $L, C, R \in \{0, 1\}$, mamy tylko $2^3 = 8$ możliwych zestawów wartości argumentów tej funkcji:

$$(L, C, R) \in \{0, 1\}^3 = \{(0, 0, 0), (0, 0, 1), (0, 1, 0), (0, 1, 1), (1, 0, 0), (1, 0, 1), (1, 1, 0), (1, 1, 1)\}.$$

Każdemu z tych zestawów funkcja ewolucji przyporządkowuje liczbę ze zbioru dwuelementowego $\{0, 1\}$, istnieje zatem łącznie $2^8 = 256$ funkcji ewolucji.

Brytyjski matematyk Stephen Wolfram wprowadził bardzo wygodną konwencję nazewnictwa funkcji ewolucji elementarnego automatu komórkowego, nazywaną od jego nazwiska *kodem Wolframa*. Zgodnie z tą konwencją każdej funkcji ewolucji przyporządkowana jest liczba całkowita z przedziału $[0, 255]$. Zobaczmy dla przykładu, jak wygląda funkcja o numerze 30. W pierwszym kroku zapisujemy liczbę 30 w postaci binarnej:

$$30 = 11110_2.$$

Następnie dopisujemy na początku tyle zer, by otrzymać łącznie osiem cyfr (istnieje bowiem osiem możliwych zestawów wartości argumentów funkcji):

$$11110 \quad \to \quad 00011110.$$

Teraz wypisujemy możliwe zestawy wartości argumentów w kolejności malejącej, interpretując je jako liczby binarne (np. $110_2 = 6$, a $101_2 = 5$, zatem $110 > 101$, ponieważ $6 > 5$):

$$111, \quad 110, \quad 101, \quad 100, \quad 011, \quad 010, \quad 001, \quad 000.$$

Na koniec przyporządkowujemy kolejnym elementom powyższego ciągu kolejne cyfry binarnej reprezentacji liczby 30 (z dopisanymi na początku zerami):

$$\begin{array}{cccc} 111 \mapsto 0, & 110 \mapsto 0, & 101 \mapsto 0, & 100 \mapsto 1, \\ 011 \mapsto 1, & 010 \mapsto 1, & 001 \mapsto 1, & 000 \mapsto 0. \end{array}$$

Właśnie to przyporządkowanie określa funkcję ewolucji o numerze 30.

Cała procedura obliczania ewolucji elementarnego automatu komórkowego, w szczególności odszyfrowywania kodu Wolframa, wygląda na dosyć skomplikowaną, jest jednak niezwykle prosta do zaimplementowania. Przykład kompletnej implementacji tego automatu w języku Python wraz z jego wizualizacją można znaleźć na stronie

`https://matplotlib.org/matplotblog/posts/elementary-cellular-automata/`

Ten prosty automat komórkowy można skomplikować zasadniczo na dwa sposoby: zmieniając definicję sąsiedztwa komórki (tak, by zawierało nie dwie, a na przykład cztery najbliższe jej komórki) i zwiększając liczebność zbioru możliwych stanów komórki. Dla przykładu, niech liczba możliwych stanów komórki wynosi nie dwa, a trzy. Stan komórki możemy wówczas określić, podając jedną z trzech cyfr, np. 0, 1 lub 2, zaś stan automatu – podając ciąg zbudowany z tych trzech cyfr. Liczba możliwych zestawów wartości argumentów funkcji ewolucji wynosi tym razem $3^3 = 27$, zaś liczba funkcji ewolucji to $3^{27} = 7625597484987$. Kod Wolframa w naturalny sposób uogólnia się na przypadek większej liczby stanów. Ustalenie postaci funkcji ewolucji na podstawie jej kodu Wolframa byłoby dla człowieka dosyć karkołomnym zadaniem już w przypadku trzech stanów, dla komputera zaś nie stanowi to najmniejszego problemu.

2. Opis projektu

Celem projektu jest napisanie programu `cells`, implementującego jednowymiarowy trójwartościowy automat komórkowy, a więc automat różniący się od tego opisanego we Wprowadzeniu jedynie tym, że liczebność zbioru możliwych stanów komórki wynosi trzy, a nie dwa (niektóre własności tego automatu podaliśmy pod koniec Wprowadzenia).

Program powinien przyjmować trzy argumenty wywołania:

- liczbę całkowitą z przedziału $[0, 7625597484986]$, określającą funkcję ewolucji automatu zgodnie z trójwartościowym kodem Wolframa,

- łańcuch tekstowy zbudowany wyłącznie ze znaków `0`, `1` i `2`, określający początkowy stan automatu: kolejne cyfry definiują początkowy stan kolejnych komórek automatu; długość łańcucha określa oczywiście ilość komórek tworzących automat,
- liczbę naturalną określającą ilość kroków ewolucji do wykonania.

Wynikiem działania programu powinna być grafika prezentująca stan automatu w kolejnych etapach jego ewolucji. Każdemu z trzech możliwych stanów komórki automatu powinien być przyporządkowany inny kolor. Grafika powinna zostać wykonana z wykorzystaniem pakietu Matplotlib i wyświetlona na ekranie.

Trzeba tu podkreślić, że wynikiem działania programu ma być dwuwymiarowy, nieruchomy obraz, a nie animacja. Automat jest jednowymiarowy, zaś obraz – dwuwymiarowy, zatem poziomą oś obrazu można wybrać jako biegnącą wzdłuż automatu, zaś pionową – jako oś czasu, i w ten sposób przedstawić ewoluujący układ na nieruchomej grafice. Analogiczny sposób wizualizacji ewolucji automatu przyjęto w przykładzie implementacji automatu elementarnego, do którego odnośnik znajduje się we Wprowadzeniu.

Projekt może wykorzystywać wyłącznie moduł `math` oraz pakiety Matplotlib i NumPy, korzystanie z innych modułów i pakietów jest zabronione.