Projektowanie algorytmów (w języku Python) – narzędzia programistyczne i grafy

Maciej Hojda

1 Instalacja oprogramowania (instrukcje w dalszej części)

1. Maszyna wirutalna Xubuntu (opcjonalne).
2. Python w wersji np. 3.8.5
3. Środowisko zintegrowane PyCharm.
4. Podstawowe biblioteki.

   Uwaga: Korzystanie z wbudowanych i dodatkowych biblotek Python-a będzie ograniczane (np. odpada skorzystanie z wbudowanej funkcji sortującej jeśli zadanie polega na implementacji algorytmu sortowania).

2 Grafy – przypomnienie

Graf skierowany $\mathbb{G}$ to dwójka $(\mathbf{V}, \mathbf{E})$, gdzie $\mathbf{V} = \{v_1, v_2, \ldots, v_V\}$ to zbiór wierzchołków v_i, a $\mathbf{E} = \{e_1, e_2, \ldots, e_E\}$ to zbiór krawędzi e_i. Krawędzią grafu skierowanego jest ukierunkowane połączenie między parą wierzchołków, i symbolicznie jest to oznaczane jako dwójka. Np. $e_1 = (v_3, v_1)$ oznacza krawędź skierowaną z wierzchołka v_3 do wierzchołka v_1 [Rys. 1a]. O krawędzi e_1 mówimy, że jest wychodząca z v_3 i wchodząca do v_1.

Uwaga: krawędzie można też definiować za pomocą relacji – tutaj nie będziemy tak robić.

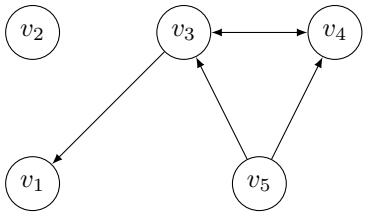
Graf nieskierowany $\mathbb{G}$ to dwójka $(\mathbf{V}, \mathbf{E})$ (analogicznie jak w przypadku grafu skierowanego), gdzie krawędź jest nieukierunkowanym połączeniem między parą wierzchołków symbolicznie oznaczanym przez dwójkę, np. $e_1 = (v_3, v_1)$, co określa taką samą krawędź jak (v_1, v_3), lub przez dwuelementowy zbiór $\{v_1, v_3\}$ – zależnie od przyjętej konwencji [Rys. 1b]. O krawędzi e_1 mówimy, że jest incydentna z v_3 i z v_1.

O wierzchołkach połączonych krawędzią mówimy, że są sąsiednie.

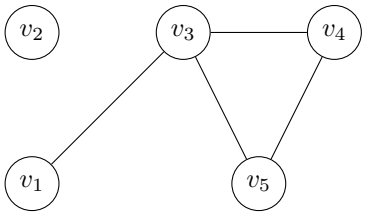
Grafy wygodnie jest przedstawiać w formie graficznej, czego przykłady pokazano na Rys. 2.

(a) dla grafu skierowanego, krawędź (v_3, v_1) (b) dla grafu nieskierowanego, krawędź $\{v_3, v_1\}$

Rysunek 1: Przykład krawędzi



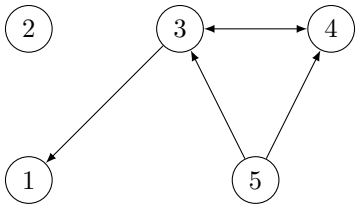
(a) skierowanego
$\mathbf{V} = \{v_1, v_2, v_3, v_4, v_5\}$
$\mathbf{E} = \{(v_3, v_1), (v_3, v_4), (v_4, v_3), (v_5, v_3), (v_5, v_4)\}$



(b) nieskierowanego
$\mathbf{V} = \{v_1, v_2, v_3, v_4, v_5\}$
$\mathbf{E} = \{\{v_3, v_1\}, \{v_3, v_4\}, \{v_5, v_3\}, \{v_5, v_4\}\}$

Rysunek 2: Przykład grafu

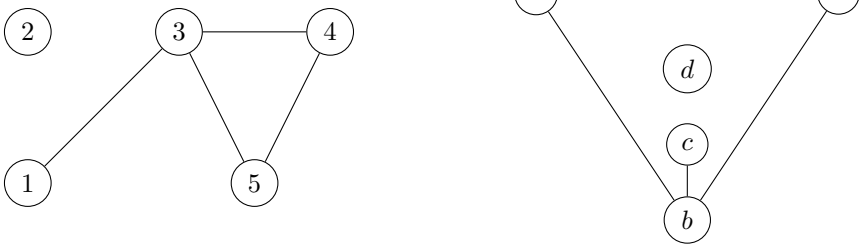
Zapis zbioru wierzchołków może przyjmować inną postać, np. $\mathbf{V} = \{1, 2, 3, \ldots, V\}$, co upraszcza też zapis zbioru krawędzi $\mathbf{E} = \{(3, 1), (3, 4), \ldots, (5, 4)\}$ i zapis graficzny [Rys. 3] (graf nieskierowany analogicznie).



Rysunek 3: Inny zapis grafu skierowanego

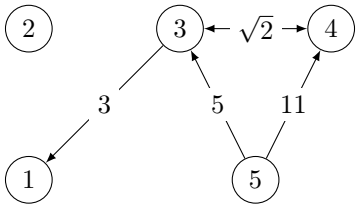
W ogólności, oznaczenia (nazwy, symbole) wierzchołków mogą być bardzo zróżnicowane (litery, teksty itd.), o ile są unikatowe (dwa wierzchołki nie mogą mieć tego samego oznaczenia – co jest jasne, bo wiemy, że $\mathbf{V}$ jest zbiorem). Dodatkowo, symbole wierzchołków nie muszą być tożsame z ich etykietami w reprezentacji graficznej (chociaż do w poprzednich przykładach właśnie tak było). Żeby utożsamiać grafy o identycznej strukturze (ale o różnych oznaczeniach) wprowadzamy pojęcie izomorfizmu.

Dwa grafy $(\mathbf{V}_1, \mathbf{E}_1)$ i $(\mathbf{V}_1, \mathbf{E}_1)$ są **izomorficzne** jeśli istnieje przekształcenie różnowartościowe i „na” (bijekcja) $f : \mathbf{V}_1 \to \mathbf{V}_2$ takie, że $(v_1, v_2) \in \mathbf{E}_1 \Leftrightarrow (f(v_1), f(v_2)) \in \mathbf{E}_2$. W reprezentacji graficznej należy to rozumieć tak, że wierzchołki pierwszego grafu można tak poprzestawiać i tak zmienić im oznaczenia, że otrzymamy w efekcie drugi graf [Rys. 4].



Rysunek 4: Grafy izomorficzne – przykład

Grafem ważonym (z ważonymi krawędziami) nazywamy taki graf skierowany lub nieskierowany, do którego krawędzi przypisano wartości liczbowe [Rys.5]. Interpretacja tych wartości zależy od rozpatrywanego problemu, do którego modelowania wykorzystano graf. Może to być odległość między punktami, koszt przejazdu, czas przejazdu, przepustowość itd. Wagi na krawędziach są szczególnym typem etykiet (mogą być inne, np. tekstowe).



Rysunek 5: Graf ważony (skierowany)

Ścieżką w grafie $(\mathbf{V}, \mathbf{E})$ nazywamy ciąg wierzchołków $(u_1, u_2, \ldots, u_N)$, gdzie $\forall i \in \{1, 2, \ldots, N\} : u_i \in \mathbf{V}$, oraz $\forall i \in \{1, 2, \ldots, N-1\} : (u_i, u_{i+1}) \in \mathbf{E}$. Długość ścieżki (rozumianej jako liczba wierzchołków) to $N-1$, czyli liczba krawędzi w ścieżce. Przykładowo, na [Rys.5] mamy ścieżki $(5, 3, 1)$ i $(5, 4, 3)$, ale nie ścieżkę $(1, 3, 4)$. Ścieżkę której wszystkie wierzchołki są różne nazywamy ścieżką prostą (jest to np. $(5, 3, 1)$).

3 Zadania

3.1 Zadanie nr 1

Uruchom (i przeanalizuj) następujący program

```
import networkx as nx
import matplotlib.pyplot as plt

G = nx.Graph()
G.add_edge('A', 'B', weight=4)
G.add_edge('B', 'D', weight=2)
G.add_edge('A', 'C', weight=3)
G.add_edge('C', 'D', weight=4)

pos = nx.spring_layout(G)
nx.draw_networkx_nodes(G, pos, node_size = 500)
nx.draw_networkx_labels(G, pos)
nx.draw_networkx_edges(G, pos)
plt.show()
```

3.2 Zadanie nr 2

Uruchom (i przeanalizuj) następujący program

```
import networkx as nx
import matplotlib.pyplot as plt
import numpy as np

G = nx.Graph()
VV = [1, 2, 3, 4, 5]
WW = [(1, 2), (2, 3), (3, 4), (4, 5), (1, 3), (3, 5)]
Vx = {1:0, 2:1, 3:2, 4:3, 5:4}
Vy = {1:0, 2:1, 3:0, 4:-1, 5:0}

g = nx.Graph();
gpos = {};

for v in VV:
  g.add_node(v);
```

```
  gpos[v] = [Vx[v], Vy[v]]

for v1 in VV:
  for v2 in VV:
    if (v1, v2) in WW:
      label = str(np.sqrt((Vx[v1] - Vx[v2])**2 + (Vy[v1] - Vy[v2])**2))
      g.add_weighted_edges_from([(v1, v2, label)])

nx.draw(g, gpos, with_labels=True, node_color='yellow')
labels = nx.get_edge_attributes(g, 'weight')
nx.draw_networkx_edge_labels(g, gpos, edge_labels=labels)

plt.show()
```

3.3 Zadanie nr 3

Napisz program wyświetlający graf pełny o parametrach:

- liczba wierzchołków zadana parametrycznie (należy zapytać użytkownika, np. z poziomu konsoli),
- wierzchołki rozmieszczone na okręgu, w równych odstępach
- etykiety wierzchołkow są kolejnymi liczbami naturalnymi.

3.4 Zadanie nr 4

Napisz program losujący graf o parametrycznie zadanej liczbie wierzchołków (podawana przez użytkownika). Wierzchołki są generowane na losowej pozycji na płaszczyźnie Oxy. Losowej, czyli losowanej z rozkładów jednostajnych dla osi Ox i Oy na pewnych ustalonych przedziałach.

3.5 Zadanie nr 5

Zmodyfikuj poprzedni program tak, aby wierzchołki stały się środkiem okręgów o jednakowym promieniu. Okręgi, o losowanej pozycji, dodawaj pojedynczo w następujący sposób: losuj pozycję (x, y) środka nowego okręgu, jeśli okrąg zachodzi na istniejące już okręgi, odrzuć go i wylosuj następny. Przerwij procedurę dodawania okręgów, jeśli nie udało się dodać nowego w ciągu 100 iteracji. Wyświetl uzyskany graf – wyświetl ponumerowane okręgi.

4 Maszyna wirtualna (opcjonalne)

Dla wygody korzystania z obowiązującego środowiska i wersji języka Python, można skorzystać z maszyny wirtualnej. Procedura instalacji jest jak następuje.

4.1 Pobieranie

1. Pobierz i zainstaluj oprogramowanie VirtualBox (np. 6.1)
   `https://www.virtualbox.org/wiki/Downloads`
2. Pobierz obraz systemu operacyjnego Xubuntu (np. lts 22.04)
   `https://xubuntu.org/download` → `xubuntu-22.04.x-desktop-amd64.iso`

4.2 Tworzenie maszyny wirtualnej (Xubuntu)

1. Uruchom VirtualBox-a
2. Utwórz maszynę; menu: Machine -> New
   Utwórz maszynę o parametrach
   (a) Name: pwr
   (b) Type: Linux
   (c) Version: Ubuntu (64-bit)
   (d) Memory size: 8 GB (mniej też powinno zadziałać)
   (e) Hard disk: create virtual hard disk now; VDI; dynamically allocated; 40GB
3. Ustaw opcje maszyny; prawy przycisk: Settings
   (a) menu: Storage > Controller IDE > Empty
   Z prawej strony: Attributes > Optical Drive > [ikona CD] > Choose a disk file
   (b) Wybierz wcześniej pobrany obraz systemu.
   (c) Zatwierdź
   (d) menu: Settings > Display > Video Memory > 128 MB
4. Uruchom maszynę
   (a) Wybierz dysk startowy: zamontowany wcześniej obraz
   (b) Install Xubuntu; wybierz język, klawiaturę; odznacz download updates; erase disk and install Xubuntu; wybierz lokalizację, podaj nazwę użytkownika i hasło
5. Zaktualizuj system
   (a) Zrestartuj maszynę (zmień rozdzielczość `menu start > Settings > Display`)
   (b) Uruchom konsolę: Ctrl+Alt+T
   Uwaga: » oznacza komendę do wpisania w konsoli; **«** oznacza spodziewany rezultat wyświetlony w konsoli. Dodatkowo, nawiasy kwadratowe z tekstem `[...]` nie mają być traktowane dosłownie, o ile z opisu nie wynika inaczej.
   (c) `>> sudo apt update`
   (podaj hasło)
   (d) `>> sudo apt upgrade`
6. Ustaw reguły firewalla
   (a) `>> sudo iptables -P INPUT DROP`
   (b) `>> sudo iptables -P FORWARD DROP`
   (c) `>> sudo ip6tables -P INPUT DROP`
   (d) `>> sudo ip6tables -P FORWARD DROP`
   (e) `>> sudo ip6tables -P OUTPUT DROP`
   (f) `>> sudo iptables -A INPUT -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT`
   (g) `>> sudo iptables -A INPUT -i lo -j ACCEPT`
   (h) `>> sudo ip6tables -A INPUT -i lo -j ACCEPT`
   (i) `>> sudo ip6tables -A OUTPUT -o lo -j ACCEPT`
   (j) `>> sudo apt install iptables-persistent`
   (zaakceptuj domyślne)

7. Zainstaluj Dodatki
   (a) menu: Devices > Install Guest Addon CD image
   (b) otwórz płytę z pulpitu (możliwe, że otworzy się automatycznie, jeśli nie; odszukaj ją w `/media/[nazwa użytkownika]`); zapamiętaj ścieżkę
   (c) uruchom konsolę, przejdź do ścieżki
   (d) `>> cd [ścieżka]`
   (e) `>> sudo apt install build-essential module-assistant gcc make perl dkms`
   (f) `>> sudo ./autorun.sh`
   (g) `>> sudo usermod -aG vboxsf [nazwa użytkownika]`
   (h) `>> sudo reboot 0`

5 Instalacja Pythona i PyCharma (przykład dla Xubuntu)

1. Zainstaluj Pythona
   (a) Ze strony `https://www.python.org/downloads/release/python-385/` pobierz `Gzipped source tarball`
   (b) `>> sudo apt install wget build-essential libreadline-dev libncursesw5-dev libssl-dev libsqlite3-dev tk-dev libgdbm-dev libc6-dev libbz2-dev libffi-dev zlib1g-dev`
   (c) `>> cd ~/Down*`
   (d) `>> tar -xvf Python*`
   (e) `>> cd Python*5`
   (f) `>> sudo ./configure --enable-optimizations`
   (g) `>> sudo make altinstall` zweryfikuj instalację
   (h) `>> python3`
2. Zainstaluj PyCharma
   (a) Ze strony `https://www.jetbrains.com/pycharm/download/#section=linux` pobierz PyCharm community edition
   (b) `>> cd ~/Down*`
   (c) `>> tar -xvf pycharm*`
   (d) `>> cd [rozpakowany katalog]/bin`
   (e) `>> chmod 777 pycharm.sh`
   (f) `>> ./pycharm.sh`
   (g) utwórz nowy projekt, wybierz `menu Tools > Create Desktop Entry`, potem `alt+f4`
   (h) `>> sudo apt install python3-tk`
   (i) `>> sudo reboot 0`
3. Zainstaluj Pakiety pomocnicze
   (a) Uruchom PyCharma (jest już w „menu start”)
   (b) Utwórz nowy projekt lub załaduj stary

(c) w konsoli pycharma (na dole: Python Console) wpisz:

(d) `>> import pip`

(e) `>> pip.main(['install','numpy'])`

(f) `>> pip.main(['install','networkx'])`

(g) `>> pip.main(['install','matplotlib'])`