Projektowanie algorytmów (w języku Python)
narzędzia programistyczne i grafy

Maciej Hojda

1 Instalacja oprogramowania

1. Python 3 `https://www.python.org/downloads/`.
2. PyCharm (community edition) `https://www.jetbrains.com/pycharm/download/`.
3. Podstawowe biblioteki.
   (a) Uruchom PyCharm-a.
   (b) Utwórz nowy projekt lub załaduj stary.
   (c) zaimportuj biblioteki: `numpy`, `networkx`, `matplotlib` .
   Jak? Np. w konsoli (Python Console) wpisz:

```
import pip
pip.main(['install','numpy'])
pip.main(['install','networkx'])
pip.main(['install','matplotlib'])
```

Uwaga: Korzystanie z wbudowanych i dodatkowych biblotek Python-a jest ograniczone zależnie od wykonywanego zadania (np. jest niewskazane wykorzystanie wbudowanej funkcji sortującej jeśli zadanie polega na implementacji algorytmu sortowania). W razie wątpliwości – zweryfikuj u prowadzącego.

2 Grafy – przypomnienie

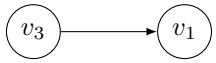
Graf skierowany $\mathbb{G}$ to dwójka $(\mathbf{V}, \mathbf{E})$, gdzie $\mathbf{V} = \{v_1, v_2, \ldots, v_V\}$ to zbiór wierzchołków v_i, a $\mathbf{E} = \{e_1, e_2, \ldots, e_E\}$ to zbiór krawędzi e_i. Krawędzią grafu skierowanego jest ukierunkowane połączenie między parą wierzchołków, i symbolicznie 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 tego robić.

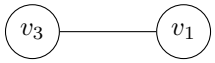
Graf nieskierowany $\mathbb{G}$ to dwójka $(\mathbf{V}, \mathbf{E})$ (analogicznie jak w przypadku grafu skierowanego), gdzie krawędź jest nieukierunkowanym połączeniem między parą wierzchołków symbolicznie oznaczanym przez dwójkę, np. $e_1 = (v_3, v_1)$, co określa taką samą krawędź jak (v_1, v_3), 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.

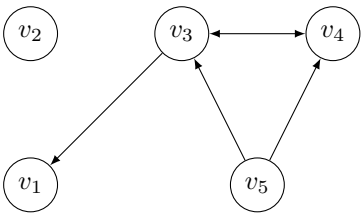


(a) dla grafu skierowanego, krawędź (v_3, v_1)

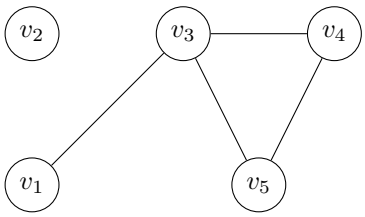


(b) dla grafu nieskierowanego, krawędź $\{v_3, v_1\}$

Rysunek 1: Przykład krawędzi



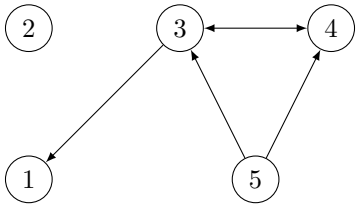
(a) skierowanego
$\mathbf{V} = \{v_1, v_2, v_3, v_4, v_5\}$
$\mathbf{E} = \{(v_3, v_1), (v_3, v_4), (v_4, v_3), (v_5, v_3), (v_5, v_4)\}$



(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

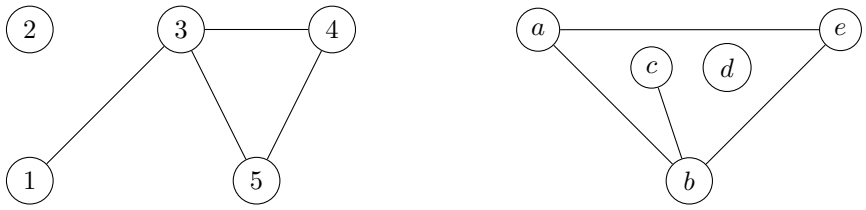
Zapis zbioru wierzchołków może przyjmować inną postać, np. $\mathbf{V} = \{1, 2, 3, \ldots, V\}$, co upraszcza też zapis zbioru krawędzi $\mathbf{E} = \{(3, 1), (3, 4), \ldots, (5, 4)\}$ i zapis graficzny [Rys. 3] (graf nieskierowany analogicznie).



Rysunek 3: Inny zapis grafu skierowanego

W ogólności, oznaczenia (nazwy, symbole, etykiety) 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].

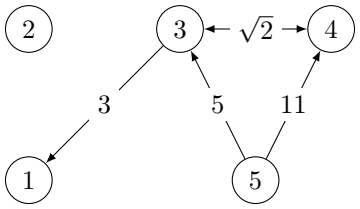


Rysunek 4: Grafy izomorficzne – przykład

Graf pełny to taki graf, który posiada połączenie (krawędź) między każdą parą wierzchołków.

Graf pusty nie posiada krawędzi.

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()
```

Wyjaśnij działanie programu. W razie potrzeby odszukaj i wykorzystaj dokumentację użytych bibliotek.

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()
```

Wyjaśnij działanie programu. W razie potrzeby odszukaj i wykorzystaj dokumentację użytych bibliotek.

Uwaga: kopiowanie kodu Pythona z PDF-a to też jakaś umiejętność.

3.3 Zadanie nr 3

Napisz program wyświetlający graf pełny o parametrach:

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

Procedurę umieszczania wierzchołków na okręgu przygotuj samodzielnie. Wykorzystaj odpowiednie funkcje trygonometryczne.

Wejście: liczba wierzchołków.
Wyjście: wyświetlony graf.

3.4 Zadanie nr 4

Napisz program wyświetlający graf pusty 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 (podanych przez użytkownika). Wyświetl uzyskany graf.

Wejście: liczba wierzchołków, przedział na osi Ox, przedział na osi Oy.
Wyjście: wyświetlony graf.

3.5 Zadanie nr 5

Zmodyfikuj poprzedni program tak, aby wierzchołki stały się środkiem kół o jednakowym promieniu (zadanym przez użytkownika). Wierzchołki dodawaj do okręgu pojedynczo, w następujący sposób.

1. Losuj pozycję (x, y) nowego wierzchołka.
2. Wyświetl wierzchołek (dodaj do wyświetlanego grafu odpowiednie koło).
3. Sprawdź, czy koło o środku (x, y) zachodzi na dowolne inne koło.
4. Jeśli zachodzi, to: odrzuć wierzchołek, usuń wyświetlone wcześniej koło (możesz przerysować cały graf) i wróć do 1. Przerwij procedurę dodawania wierzchołków, jeśli nie udało się dodać nowego w ciągu 10 iteracji.

Wejście: liczba wierzchołków, przedział na osi Ox, przedział na osi Oy, promień koła.
Wyjście: graf wyświetlany wierzchołek po wierzchołku (np. kolejny wierzchołek po naciśnięciu spacji).

Uwaga: jeśli jest mowa o modyfikacji poprzedniego/wcześniejszego programu, to nadal należy zachować wersję z poprzedniego/wcześniejszego zadania.