UNIWERSYTET PEDAGOGICZNY
im. Komisji Edukacji Narodowej w Krakowie

INSTYTUT INFORMATYKI

Kierunek: INFORMATYKA

specjalność: nazwa specjalności

Imię i nazwisko dyplomanta

Nr albumu: XXXXXX

TYTUŁ PRACY

Praca magisterska/inżynierska
napisana pod kierunkiem
tytuł/stopień naukowy, imię i nazwisko promotora

Kraków 2020

Streszczenie

Streszczenie powinno zawierać krótkie i skondensowane omówienie zagadnień poruszanych w pracy dyplomowej. Abstract jest streszczeniem pracy dyplomowej napisanym w języku angielskim, które zawiera te same treści, co streszczenie w języku polskim. Objętość każdego z tych elementów z osobna nie powinna przekraczać około połowy strony.

Abstract

The summary should include a brief and condensed discussion of the issues raised in the thesis. Abstract is a written in English a summary of the diploma thesis, which contains the same elements as the summary in Polish. The volume of each of these elements individually should not exceed about half of the page.

Spis treści

Wstęp

Niniejszą pracę należy traktować tylko i wyłącznie jako pewien szablon własnej pracy dyplomowej, a treści w niej zwarte mają jedynie charakter poglądowy oraz ilustrujący pewne praktyki jakie należy stosować podczas pisania właściwej pracy.

Rozdział 1

Preliminaria

Zanim przejdziemy do opisów algorytmów grafowych w niniejszym rozdziale wprowadzimy podstawowe pojęcia teorii grafów, które będą stosowane w dalszej części pracy. Te informacje jak i ich uzupełnienie można znaleźć między innymi w [1, 2, 3].

1.1 Podstawowe definicje i twierdzenia teorii grafów

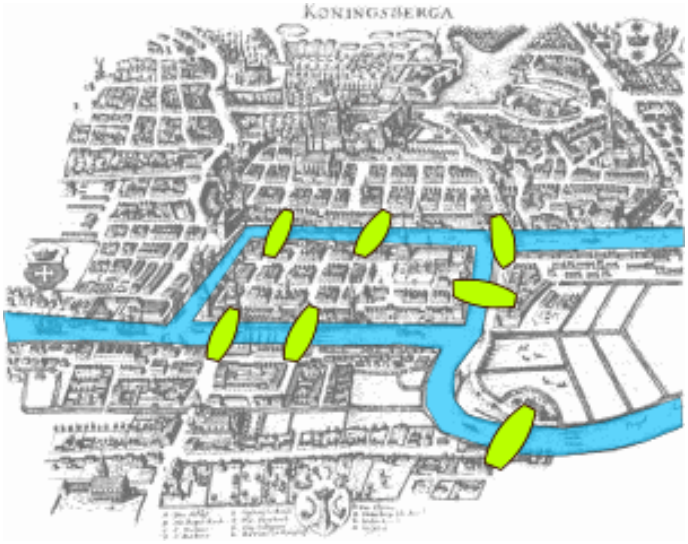
Pojęcie *grafu* jest dość intuicyjnym i stąd można spotkać w literaturze nieco różniące się od siebie definicje. Można powiedzieć, że graf prezentuje pewien zbiór punktów oraz połączenia między nimi. Graf to struktura matematyczna, która służy do opisu oraz badania relacji między pewnymi obiektami. Za pioniera w teorii i badaniu grafów uważa się Leonarda Eulera[1)], który wykorzystując grafy rozwiązał tzw. *problem mostów królewieckich*. Zganienie to dotyczyło siedmiu mostów (zob. rysunek 1.1) przerzuconych nad rzeką Pregoła przepływającej przez Królewiec (obecna, rosyjska nazwa to Kaliningrad). W rozwidleniu tej rzeki znajdowały się dwie wyspy. Jeden z mostów łączył obie wyspy, a pozostałe mosty łączyły wyspy z brzegami rzeki. Zagadnienie, którym zainteresował się Euler, było następujące: *czy można przejść kolejno przez wszystkie mosty tak, żeby każdy przekroczyć tylko raz?*

Definicja 1.1. *Grafem nieskierowanym* (*niezorientowanym*) G nazywamy następującą parę (V, E) składającą się z niepustego i skończonego zbioru *wierzchołków* $V(G)$ oraz skończonego zbioru

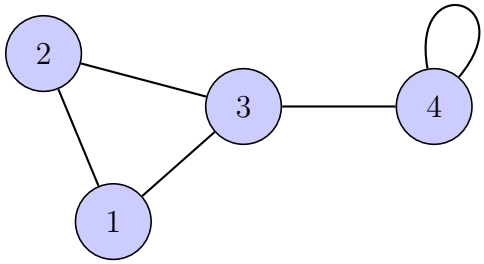
$$E(G) = \Big\{\{v_i, v_j\} \colon v_i, v_j \in V\Big\}.$$

[1)] Leonhard Euler (1707-1783) – szwajcarski matematyk i fizyk. Był prekursorem rozwoju w wielu obszarach obu tych nauk. Większą część swojego życia spędził w Rosji oraz Prusach. Jest uważany za jednego z najbardziej produktywnych matematyków w historii.

Zbiór dwuelementowy $\{u, v\} \in E$ nazywamy *krawędzią*, a wierzchołki u i v nazywamy *końcami* tej krawędzi. Jeżeli $u = v$ to krawędź $\{u, v\} \in E$ nazywamy *pętlą własną*. Mówimy, że krawędź grafu $e \in E$ jest *incydentna* do wierzchołka u, jeśli ma ona w tym wierzchołku swój koniec lub początek (wychodzi lub wchodzi do tego wierzchołka). Do oznaczenia krawędzi grafu $\{u, v\}$ będziemy stosować notację (u, v) - w tym przypadku zapis (u, v) i (v, u) oznacza tę samą krawędź (zob. przykład na rysunku 1.2).



Rysunek 1.1: Mapa przedstawiająca mosty królewieckie za życia Leonharda Eulera. Źródło: [4]



Rysunek 1.2: Graf nieskierowany $G = (V, E)$, gdzie $V = \{1, 2, 3, 4\}$ i $E = \big\{(1, 2), (1, 3), (2, 3), (3, 4), (4, 4)\big\}$. Krawędź $(4, 4)$ jest pętlą własną.

Definicja 1.2. Jeżeli przyjmiemy, że w definicji grafu $G = (V, E)$ zbiór krawędzi zawiera powtarzające się krawędzie lub pętle (tzn. gdy E jest *multizbiorem*[2)]) to takie

[2)] *Multizbiór* jest uogólnieniem pojęcia zbioru, w którym jeden element (w naszym wypadku krawędź grafu) może występować więcej niż jeden raz.

powtarzające się krawędzie lub pętle nazywamy *równoległymi* lub *wielokrotnymi*, sam graf nazywamy *multigrafem*. Graf bez krawędzi wielokrotnych i pętli własnych nazywamy *grafem prostym*.

Definicja 1.3. Jeżeli $G = (V, E)$ jest grafem niezorientowanym, to stopniem $d(u)$ wierzchołka $u \in V$ nazywamy liczbę krawędzi incydentnych z wierzchołkiem u. Wierzchołek $u \in V$ taki, że $d(u) = 0$ nazywamy *izolowanym*, a wierzchołek o $d(u) = 1$ nazywamy *wiszącym* (krawędź z nim incydentną nazywamy *wiszącą*).

Twierdzenie 1.1. *Niech q oznacza liczbę krawędzi w grafie nieskierowanym $G = (V, E)$. Suma stopni wierzchołków w takim grafie jest równa podwojonej liczbie krawędzi*

$$\sum_{u \in V} d(u) = 2q. \tag{1.1}$$

Dowód. Każda krawędź zwiększa o jeden stopień wierzchołka z któryi jest incydentna (w przypadku pętli własnych oznacza to zwiększenie stopnia wierzchołka o dwa). Stąd mamy równość (1.1). □

1.2 Reprezentacja grafów

Istnieje kilka metod reprezentacji grafu $G = (V, E)$, jednymi z najbardziej standardowych są *listy sąsiedztwa* i *macierze sąsiedztwa*…

W tabeli 1.1 zaprezentowano złożoności pamięciowe najpopularniejszych reprezentacji grafów…

Tablica 1.1: Złożoność pamięciowa wybranych reprezentacji grafów.

Reprezentacja grafu	**Złożoność pamięciowa**
macierz sąsiedztwa	$\mathcal{O}\Big(\|V\|^2\Big)$
lista sąsiedztwa	$\mathcal{O}\Big(\|E\|\Big)$
pęki wyjściowe	$\mathcal{O}\Big(\|E\| + \|V\|\Big)$

Rozdział 2

Implementacja wybranych algorytmów grafowych

2.1 Przeszukiwanie wszerz

Przeszukiwanie wszerz to jeden z podstawowych i najprostszych algorytmów przeszukiwania grafów. Wiele algorytmów korzysta z podobnych rozwiązań jak w algorytmie przeszukiwania wszerz.

W przeszukiwaniu grafu wszerz rozważamy graf $G = (V, E)$ wraz z wyróżnionym wierzchołkiem s, który nazywamy *wierzchołkiem źródłowym*. Podczas przeszukiwania grafu wszerz systematycznie badamy krawędzie grafu G w celu odwiedzenia wszystkich wierzchołków osiągalnych z wierzchołka s. Jednocześnie wyliczmy odległości liczone najmniejszą liczbą krawędzi od wierzchołka źródłowego do wszystkich osiągalnych wierzchołków. W wyniku działania algorytmy otrzymujemy drzewo przeszukiwania wszerz o korzeniu s, które zawiera wszystkie osiągalne wierzchołki. Dla każdego wierzchołka v osiągalnego z s droga prosta w tym drzewie od s do v odpowiada najkrótszej drodze od s do v w grafie G, czyli tej zawierającej najmniejszą liczbę krawędzi. Procedura BSF jest tak samo dobra dla grafów skierowanych jak i nieskierowanych.

W procedurze przeszukiwania wszerz (BFS) zakładamy, że graf wejściowy $G = (V, E)$ jest reprezentowany przez listy sąsiedztwa. Z każdym wierzchołkiem grafu związane są dodatkowe zmienne. Kolor każdego wierzchołka $u \in V$ jest przechowywany w atrybucie $u.color$, a poprzednik wierzchołka u jest przechowywany w atrybucie $u.\pi$. Jeżeli dany wierzchołek u nie posiada poprzednika to $u.\pi = \text{NIL}$. Obliczana odległość od źródła s do wierzchołka u jest pamiętana w atrybucie $u.d$. W algorytmie korzystamy z kolejki Q typu FIFO, w której przechowujemy wierzchołki o kolorze

```
1  Function BFS(G, s):
2      for all u ∈ G.V − {s} do
3          u.color = WHITE;
4          u.d = ∞;
5          u.π = NIL;
6      s.color = GRAY;
7      s.d = 0;
8      s.π = NIL;
9      Q = ∅;
10     ENQUEUE(Q, s);
11     while Q ≠ ∅ do
12         u = DEQUEUE(Q);
13         for all v ∈ G.Adj[u] do
14             if v.color == WHITE then
15                 v.color = GRAY;
16                 v.d = d(u) + 1;
17                 v.π = u;
18                 ENQUEUE(Q, v);
19         u.color = BLACK;
```

Rysunek 2.1: Pseudokod procedury BSF.

szarym. Pseudokod procedury BSF prezentuje rysunek 2.1 natomiast implementacja tej procedury w języku Python jest przedstawiona na rysunku 2.2.

```
1  def BSF(Graph, start):
2      for vertex in Graph:
3          if start is vertex:
4              vertex.v_colour = 'szary'
5              vertex.v_d = 0
6              vertex.v_previous = float('inf')
7          else:
8              vertex.v_colour = 'biały'
9              vertex.v_d = float('inf')
10             vertex.v_previous = None
11     Q = collections.deque()
12     Q.append(start)
```

```
13      while len(Q) > 0:
14          print(Q)
15          vv =  Q.popleft()
16          for vertex in Graph[vv]:
17              if vertex.v_colour == 'biały':
18                  vertex.v_colour = 'szary'
19                  vertex.v_d = vv.v_d + 1
20                  vertex.v_previous = vv
21                  Q.append(vertex)
22          vv.v_colour = 'czarny'
23
24      return Graph
```

Rys. 2.2: Kod źródłowy procedury BSF zaimplementowanej w języku Python.

Zakończenie

Rozdział zawierający podsumowanie całej pracy.

Literatura

[1] T. Cormen, C. Leiserson, R. Rivest, C. Stein. *Wprowadzenie do algorytmów*. Wydawnictwo Naukowe PWN, Warszawa, 2012.

[2] W. Kordecki, A. Łyczkowska Hańćkowiak. *Matematyka dyskretna dla informatyków*. Wydawnictwo Helion, Gliwice, 2018.

[3] J. Wojciechowski, K. Pieńkowski. *Grafy i sieci*. Wydawnictwo Naukowe PWN, Gliwice, 2013.

[4] Zagadnienie mostów królewieckich. https://pl.wikipedia.org/wiki/Zagadnienie_mostów_królewieckich. Dostęp: 2020-06-07.

Imię i Nazwisko Kraków, dnia .....

O Ś W I A D C Z E N I E

Świadomy(a) odpowiedzialności oświadczam, że przedłożona praca pt.:

Tu wpisać tytuł pracy

została napisana przeze mnie samodzielnie.

Jednocześnie oświadczam że w/w praca nie narusza praw autorskich w rozumieniu Ustawy z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (Dz.U. z 2006 nr 90, poz. 631 z późn. zmianami) oraz dóbr osobistych chronionych prawem cywilnym.

Przedłożona praca nie zawiera danych empirycznych ani też informacji, które uzyskałem(am) w sposób niedozwolony. Stwierdzam, iż przedstawiona praca w całości ani też w części nie była wcześniej podstawą żadnej innej urzędowej procedury związanej z uzyskaniem dyplomu ani też nadania tytułów zawodowych.

.....

(podpis)