ESI: Podstawy teorii grafów

[Matlab_1.0] Matlab2016b i nowsze

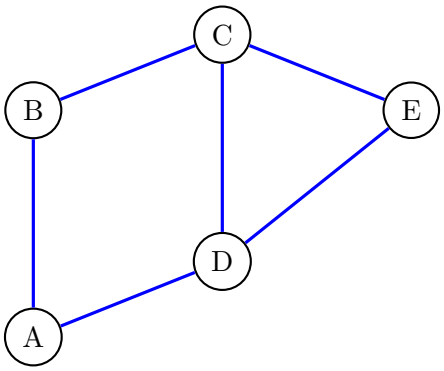
18 marca 2019

1. **Cel ćwiczeń:** Celem ćwiczeń jest zapoznanie się studentów z podstawami teorii grafów. Lista zawiera ćwiczenia związane z przeszukiwaniem i kolorowaniem grafów. Studenci poznają także podstawowe problemy teorii grafów: problem komiwojażera, minimalnego drzewa rozpinającego oraz najkrótszej ścieżki.

2. **Grafy:** Graf jest podstawowym elementem rozważań teorii grafów. Przedstawia on i opisuje relacje (krawędzie) pomiędzy obiektami (wierzchołki). W ujęciu matematycznym graf jest dwójką, taką że

$$G = (V, E), \tag{1}$$

gdzie $V \neq \emptyset$ jest niepustym zbiorem wierzchołków, natomiast $E = \Big\{\{a, b\} : a, b \in V, a \neq b\Big\}$ jest zbiorem krawędzi - rodziną podzbioru wierzchołków. Rys. 1 przedstawia przykładowy graf (nieskierowany). Istnieje także pojęcie grafu skierowanego. Wówczas krawędzie są parą (zbiorem uporząd-



Rysunek 1: Przykładowy graf nieskierowany

kowanym). Oznacza to, iż możliwe jest jedynie przejście w jedną stronę pomiędzy poszczególnymi wierzchołkami grafu połączonymi jedną krawędzią. Matematycznie, graf skierowany jest dwójką

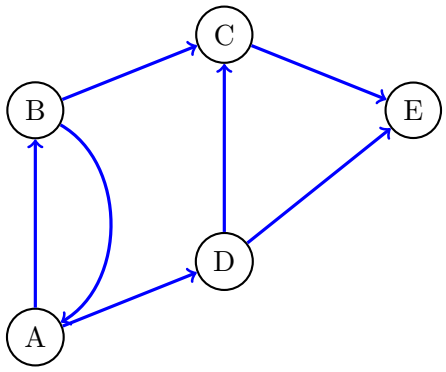
$$G = (V, A), \tag{2}$$

gdzie $V \neq \emptyset$ jest niepustym zbiorem wierzchołków, natomiast $A = \Big\{(a, b) : a, b \in V, a \neq b\Big\}$ jest zbiorem par (dwuelementowych zbiorów uporządkowanych). Rys. 2 przedstawia przykładowy graf

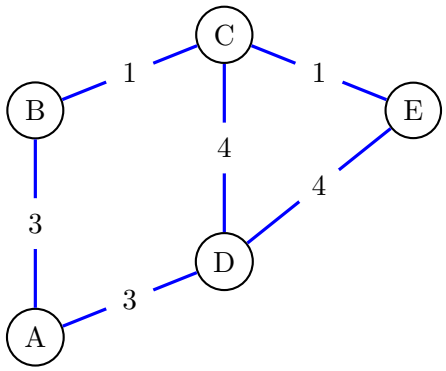
skierowany. Ponadto można wyróżnić grafy ważone (zarówno skierowane jak i nieskierowane). Wówczas, graf jest określony jako

$$G = (V, E, W) \qquad \text{lub} \qquad G = (V, A, W), \tag{3}$$

w zależności czy graf jest skierowany czy nieskierowany, gdzie dodatkowo W jest przekształceniem $W : E \to X$ (lub $W : A \to X$) takim, że każdej krawędzi przyporządkowana jest waga $w(e) = x$. Rysunek 3 przedstawia przykładowy graf ważony (nieskierowany).



Rysunek 2: Przykładowy graf skierowany



Rysunek 3: Przykładowy graf ważony

Ponadto należy wyróżnić następujące pojęcia i definicje:

stopień wierzchołka – liczba opisujące ilość krawędzi dochodzących do danego wierzchołka $deg(v) = x$ ponadto oznacza się wierzchołek o największym stopniu $\Delta(v)$ oraz najmniejszym $\delta(v)$, np w grafie z Rys. 2, $deg(v_C) = 2$,

graf regularny – graf jest regularny jeżeli stopień każdego wierzchołka $deg(v_i) = r, \forall_i$ jest taki sam,

droga – uporządkowany ciąg wierzchołków $v_i, \dots v_n$ taki, że kolejne wierzchołki połączone są krawędzią,

cykl – droga której wierzchołek początkowy i końcowy to ten sam wierzchołek $v_i, \dots v_n, \dots, v_i$,

graf acykliczny – graf nie posiadający cykli,

graf spójny – graf w którym istnieje droga pomiędzy dwoma dowolnie wybranymi wierzchołkami,

izomorfizm – graf G jest izomorficzny z grafem H, jeżeli istnieje bijekcja wierzchołków grafu H w G zachowująca krawędzie,

podgraf – podgraf grafu G, to graf zawierający podzbiór wierzchołków i krawędzi grafu G,

graf planarny – graf, który przedstawiony na płaszczyźnie nie posiada przecinających się krawędzi

grubość grafu – grubość grafu $t(G)$ oznacza minimalną ilość planarnych podgrafów grafu G

spójność grafu – spójność wierzchołkowa k oznacza, iż usunięcie co najmniej k wierzchołków z grafu spowoduje utratę spójności. Spójność krawędziowa jest zdefiniowana analogicznie względem krawędzi,

dwudzielność – graf G jest dwudzielny, jeżeli jego zbiór wierzchołków można podzielić na rozłączne podzbiory X i Y takie, że każda krawędź grafu G ma początek w X a koniec w Y,

drzewo - acykliczny, spójny graf nieskierowany,

liczba chromatyczna – liczba kolorów niezbędna do optymalnego wierzchołkowego pokolorowania grafu

macierz sąsiedztwa – macierz kwadratowa, w której $a_{i,j}$ oznacza wagę krawędzi pomiędzy wierzchołkami i i j,

macierz incydencji – macierz M której liczba kolumn j jest równa liczbie krawędzi a liczba wierszy i liczbie wierzchołków. W grafie nieskierowanym $m_{i,j} = 1$ gdy j-ta krawędź prowadzi do i-tego wierzchołka. W grafie skierowanym $m_{i,j} = -1$ gdy v_i jest początkiem krawędzi j a $m_{i,j} = 1$ gdy v_i jest jej końcem,

cykl Hamiltona – cykl Hamiltona to taki cykl w grafie, w którym każdy wierzchołek grafu odwiedzany jest dokładnie raz [2].

3. **Algorytmy przeszukiwania grafów:** Algorytmy przeszukiwania grafów pozwalają na przechodzenie po wszystkich wierzchołkach grafu w sposób usystematyzowany [3].

   3.1. **Przeszukiwanie w głąb DFS:** ang. Depth-First Search. Wierzchołki odwiedzane są jeden po drugim, zachłannie z odkładaniem do kolejki zgodnie z zasadą pierwszy wchodzi pierwszy wychodzi. Danymi wejściowymi algorytmu są macierz sąsiedztwa grafu oraz wierzchołek początkowy v

   a) oznacz wszystkie wierzchołki jako nieodwiedzone
   b) odwiedź wierzchołek v
   c) utwórz kolejkę FILO sąsiadów wierzchołka v
   d) zdejmij wierzchołek u z kolejki
   e) powtarzaj kroki b)-d) dla wierzchołka u i kolejnych wierzchołków dopóki kolejka nie jest pusta

3.2. **Przeszukiwanie wszerz BFS:** and. Breadth-First Search. Wierzchołki odwiedzane są jeden po drugim, zachłannie z odkładaniem do kolejki zgodnie z zasadą pierwszy wchodzi ostatni wychodzi, oznacza to, iż w pierwszej kolejności odwiedzane są wszystkie wierzchołki sąsiadujące z badanym wierzchołkiem. Danymi wejściowymi algorytmu są macierz sąsiedztwa grafu oraz wierzchołek początkowy v

a) oznacz wszystkie wierzchołki jako nieodwiedzone

b) odwiedź wierzchołek v

c) utwórz kolejkę FIFO sąsiadów wierzchołka v

d) zdejmij wierzchołek u z kolejki

e) powtarzaj kroki b)-d) dla wierzchołka u i kolejnych wierzchołków dopóki kolejka nie jest pusta

3.3. **Algorytm Dijkstry:** Algorytm Dijkstry pozwala na wyznaczenie najkrótszych ścieżek w grafie pomiędzy zadanym wierzchołkiem początkowym v oraz wszystkimi pozostałymi wierzchołkami grafu [1]. Danymi wejściowymi algorytmu są macierz sąsiedztwa grafu oraz wierzchołek początkowy v. $r(u, w)$ oznacza wagę krawędzi pomiędzy wierzchołkami u, w

a) utwórz tablicę odległości od v dla wszystkich wierzchołków. Odległości wszystkich wierzchołków poza początkowym wynosi $d(u) = \infty$, natomiast $d(v) = 0$

b) utwórz kolejkę priorytetową P wszystkich wierzchołków, za wartość priorytetu obierz $d(u)$

c) zdejmij z kolejki wierzchołek u o najniższym priorytecie

d) dla każdego sąsiada w wierzchołka u dokonaj relaksacji poprzez u (jeżeli $d(w) > d(u) + r(u, w)$ to $d(w) := d(u) + r(u, w)$)

e) powtarzaj kroki c) i d) dopóki kolejka nie jest pusta

3.4. **Algorytm A*:** Algorytm A* podobnie jak algorytm Dijkstry wykorzystuje informacje o przebytej drodze, jednak w inny sposób. Ponadto, algorytm A* wymaga podania zarówno wierzchołka startowego jak i docelowego. W każdym kroku algorytm wybiera taki wierzchołek v minimalizując

$$f(v) = g(v) + h(v), \tag{4}$$

gdzie $g(v)$ oznacza drogę pomiędzy wierzchołkiem v a wierzchołkiem początkowym, $h(v)$ jest przewidywaną drogą do wierzchołka docelowego

4. **Algorytmy kolorowania grafów:** W ogólności kolorowanie grafu polega na przypisaniu określonym elementom składowym grafu (najczęściej wierzchołkom) kolorów według zadanych reguł. Kolorowanie wierzchołków grafu jest związane z przypisaniem wszystkim wierzchołkom w grafie danego kolory, tak aby żadne dwa sąsiednie wierzchołki nie miały tego samego koloru. Oznacz to, iż gdy końcom żadnej krawędzi nie przypisano tego samego koloru [2].

4.1. **Algorytm LF – largest first:**

a) uporządkuj wierzchołki grafu malejąco według ich stopni.

b) zaczynając od wierzchołka o największym stopniu koloruj wierzchołki zgodnie z ustaloną wcześniej kolejnością,

4.2. **Algorytm SL – smallest last:**

a) znajdź wierzchołek o minimalnym stopniu, usuń go z grafu i dodaj do kolejki FILO

b) powtarzaj krok a) tak długo, aż graf będzie pusty

c) koloruj wierzchołki zgodnie z ustaloną wcześniej kolejnością

4.3. **Algorytm SLF – saturated largest first:**

a) znajdź wierzchołek o maksymalnym stopniu spośród wierzchołków o maksymalnym stopniu nasycenia (liczbie różnych kolorów sąsiednich z tym wierzchołkiem)

b) pokoloruj znaleziony wierzchołek

5. **Podstawowe problemy w teorii grafów:**

5.1. **Problem komiwojażera:** Problem komiwojażera polega na znalezieniu takiej kolejności dla odwiedzanych przez wędrownego sprzedawcę (komiwojażera) miast, aby koszt podróży był jak najmniejszy przy założeniu, że każde miasto zostanie odwiedzone dokładnie raz. Innymi słowy jest to problem znalezienia minimalnego cyklu Hamiltona w grafie ważonym. Problem ten jest NP-trudny.

5.2. **Problem minimalnego drzewa rozpinającego:** ang. minimum spanning tree - jest to drzewo rozpinające grafu - tj. takie które łączy wszystkie jego wierzchołki - o najmniejszej możliwej sumie wag krawędzi. Minimalne drzewo rozpinające stosuje się w protokołach rozgłaszających sieci komputerowych, analizy klastrów oraz projektowania układów elektronicznych. Rys. 4 przestawia przykładowe minimalne drzewo rozpinające. Jednym z rozwiązań problemu minimalnego drzewa rozpinającego jest algorytm Kruskala [4]

a) utwórz las L z wierzchołków oryginalnego grafu (niepołączone wierzchołki są drzewami)

b) utwórz zbiór S zawierający wszystkie krawędzie oryginalnego grafu

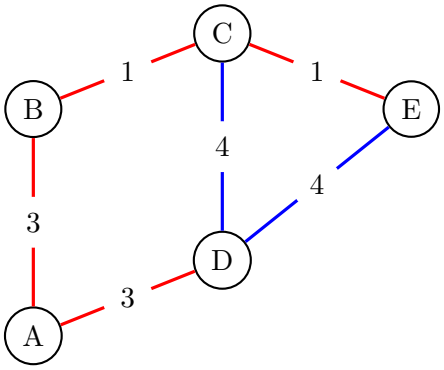
c) wybierz i usuń z S jedną z krawędzi o minimalnej wadze

d) jeśli krawędź ta łączyła dwa różne drzewa, to dodaj ją do lasu L, tak aby połączyła dwa odpowiadające drzewa w jedno, W przeciwnym wypadku odrzuć ją.

e) powtarzaj krok d) dopóki S nie jest pusty oraz L nie jest drzewem

6. **Zadania:**

6.1. Zaimplementuj algorytm przeszukiwania grafu w głąb. Sprawdź działanie algorytmu dla grafu



Rysunek 4: Minimalne drzewo rozpinające (czerwone)

opisanego macierzą sąsiedztwa w postaci

$$A = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{5}$$

6.2. Zaimplementuj algorytm przeszukiwania A*. Sprawdź działanie algorytmu dla grafu zdefiniowanego jak na Listingu 1 Plik *seed.mat* znajduje się w archiwum z listą zadań. należy go zaim-

```
load seed.mat
rng(s);
A = randi(10,[10,10]);
A(A<7)=0;
A=A-diag(A);
A=triu(A);
```

Listing 1: Dane inicjalizacyjne do zad. 1.

.

portować do przestrzeni roboczej, tak aby w oknie **Workspace** dostępna była zmienna *s*. Za wierzchołek początkowy wybierz wierzchołek o nr 2 a końcowy o nr 10.

6.3. Wykorzystując polecenie *dfsearch* sprawdź poprawność działania algorytmu zaimplementowanego w zad. 1.

6.4. Wykorzystując polecenie *bfsearch* przeszukaj graf z zad 2. Porównaj wynik z *dfsearch* i A*.

6.5. Zaimplementuj algorytm LF kolorowania grafu. Sprawdź działanie algorytmu dla grafu z zad 2.

```
G = digraph(A);
plot(G);
d = dfsearch(G)
```

Listing 2: Kod do zadania 3.

.

6.6. Zastosuj polecenie *minspantree* do wyznaczenia minimalnego drzewa rozpinającego grafu z zad. 2. Wykorzystaj polecenie *highlight* w celu graficznej reprezentacji wyniku.

6.7. Użyj polecenia *highlight* w celu podświetlenia ścieżki w grafie z zad. 2.

6.8. Sprawdź rozwiązanie problemu komiwojażera wykorzystując polecenie *openExample('optim/TravellingSalesmanExample')*.
Sprawdź zachowanie algorytmu dla $nStops = 20$; i $nStops = 100$;

Literatura

[1] T H Cormen, T H Cormen, C E Leiserson, Inc Books24x7, M I T Press, Massachusetts Institute of Technology, R L Rivest, McGraw-Hill Publishing Company, and C Stein. *Introduction To Algorithms*. Introduction to Algorithms. MIT Press, 2001.

[2] Reinhard Diestel. *Graph Theory, Electronic*, volume 173. 2000.

[3] Gross Jonathan L.; Yellen Jay. *Handbook of Graph Theory*. 2003.

[4] Joseph B. Kruskal. On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem. *Proceedings of the American Mathematical Society*, 7(1):48, 1956.