Wrocław
University
of Science
and Technology

Struktury danych

Wykład 9

Grafy

dr inż. Jarosław Rudy

HR EXCELLENCE IN RESEARCH

Graf (1)

- Graf jako ADT służy do reprezentacji grafów matematycznych.
- Matematycznie graf składa się z wierzchołków (węzłów, vertices) i łączących je krawędzi (edges).
  - Każda krawędź łączy dwa (domyślnie różne) wierzchołki grafu.
- Ściślej graf G jest parą uporządkowaną (dwójką) zbioru wierzchołków V i zbioru krawędzi E:

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

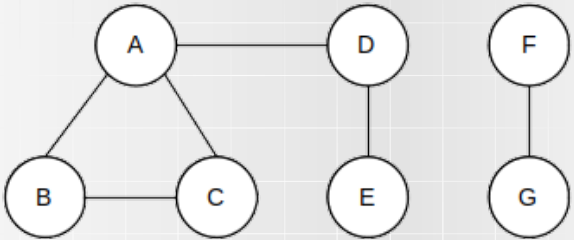
- Krawędzie zwykle definiuje sę jako zbiór E zbiorów dwuelemenowych, tak że E jest podzbiorem zbioru „wszystkich" możliwych krawędzi:

$$E \subseteq \{\{u, v\} : u, v \neq u \in V\}. \tag{2}$$

krawędź $\{u, v\}$ łączy wierzchołki u i v.

Graf (2)

Przykład grafu



$V = \{A, B, C, D, E, F, G\}$
$E = \{\{A, B\}, \{B, C\}, \{A, C\}, \{A, D\}, \{D, E\}, \{G, F\}\}$

Graf (3)

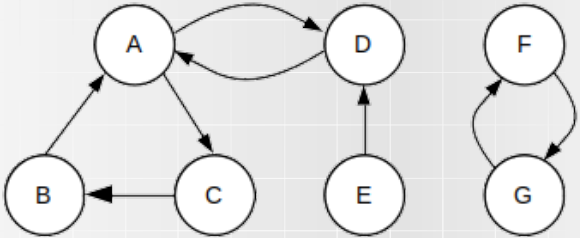
- Poprzednia definicja zakłada, że kierunek krawędzi nie ma znaczenia (graf jest nieskierowany).
- Można jednak zdefiniować zbiór krawędzi A tak by krawędzie miały kierunek tj. każda krawędź jest parą uporządkowaną

$$A \subseteq \{(u, v) : u, v \neq u \in V\}. \tag{3}$$

- Krawędź (u, v) biegnie od u do v i jest czymś innym niż krawędź (v, u).
- Taki graf $G = (V, A)$ nazywamy grafem skierowanym, zaś jego krawędzie nazywamy łukami (arcs).

Graf (4)

Przykład grafu



$V = \{A, B, C, D, E, F, G\}$
$A = \{(A, C), (B, A), (C, B), (A, D), (D, A), (E, D), (F, G), (G, F)\}$

Graf (5)

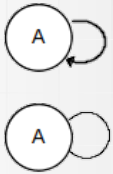
- Często przyjmowana notacja:
  - $n = |V|$ – liczba wierzchołków grafu.
  - $k = |E|$ (lub $k = |A|$) – liczba krawędzi/łuków grafu.
- Według standardowej definicji graf skierowany może mieć od 0 do $n(n-1)$ łuków.
- Analogicznie, graf nieskierowany może mieć od 0 do $\frac{n(n-1)}{2}$ krawędzi.

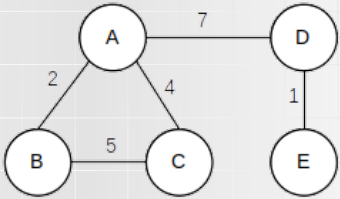
Graf (6)

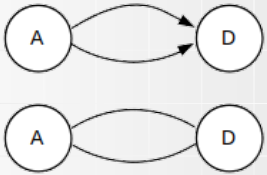
Można wprowadzać modyfikacje definicji grafu

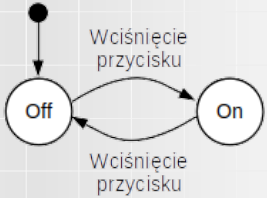
- Dopuszczanie pętli tj. krawędzi/łuków zaczynających się i kończących się w tym samym wierzchołku.
- Multigraf – dopuszczanie wielokrotnych krawędzi pomiędzy tą samą parą wierzchołków.
- Graf z wagami – krawędziom i/lub wierzchołkom można przypisywać wartości liczbowe (wagi).
  - Ogólniej można przepisywać dowolne wartości – etykiety.

Graf (7)


A
A


A
7
D
2
4
1
B
5
C
E


A
D
A
D


Wciśnięcie
przycisku
Off
On
Wciśnięcie
przycisku

Typy grafów i pojęcia grafowe (1)

Wybrane typy grafów i pojęcia grafowe:

- Rząd grafu – liczba wierzchołków.
- Rozmiar grafu – liczba krawędzi.
- Droga – ciąg następujących po sobie krawędzi (trasa). Niekiedy w skład drogi wchodzą też znajdujące się na niej wierzchołki.
  - Droga prosta – droga na której nie powtarzają się krawędzie.
  - Cykl – droga kończąca się w tym samym wierzchołku, w którym się zaczęła.

Typy grafów i pojęcia grafowe (2)

- Stopień wierzchołka (degree, deg):
  - deg(v) – w grafie nieskierowanym liczba krawędzi wchodzących do lub wychodzących z v.
  - indeg(v) – w grafie skierowanym liczba krawędzi wchodzących do v.
  - outdeg(v) – w grafie skierowanym liczba krawędzi wychodzących z v.
- Wierzchołek izolowany – wierzchołek o stopniu 0 (z którego nie wychodzą i do którego nie wchodzą krawędzie).

Typy grafów i pojęcia grafowe (3)

- Graf pełny – liczba krawędzi jest maksymalna.
- Graf gęsty – duża liczba krawędzi (k) w stosunku do liczby wierzchołków (v). Różne praktyczne definicje np.:
  - $k \notin O(n)$.
  - $k \in \Theta(n^2)$.
  - $k \gg v$.
- Graf rzadki – nieduża liczba krawędzi np.:
  - $k \in O(n)$.
  - $k \leqslant n$.
- Graf pusty – graf bez wierzchołków lub graf bez krawędzi.

Typy grafów i pojęcia grafowe (4)

- Graf spójny (connected graph) – graf, w którym istnieje droga pomiędzy każdą parą wierzchołkdów $u, v \neq u$.
- Spójna składowa grafu (component) – maksymalny (tj. taki którego nie można już powiększyć) spójny podgraf grafu. Graf może mieć wiele spójnych składowych.
- Drzewo – graf nieskierowany, acykliczny (brak cykli) i spójny.
  - Dokładnie 1 droga pomiędzy każdą parą wierzchołków.

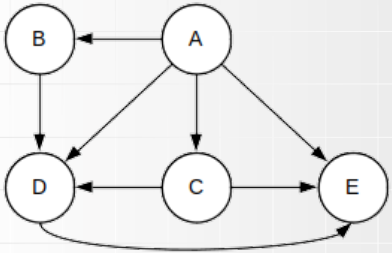
Typy grafów i pojęcia grafowe (5)

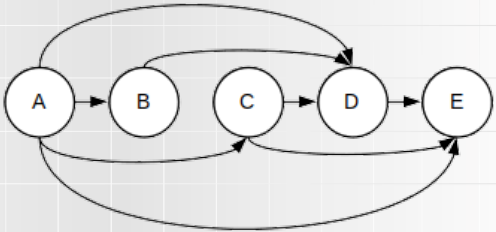
Skierowany graf acykliczny (directed acyclic graph, DAG)

- Graf skierowany, w którym nie ma skierowanych cykli.
- W DAG podążanie krawędziami (zgodnie z ich kierunkiem) nigdy nie doprowadzi do cyklu.
  - Jeśli jest krawędź (u, v) to nie ma krawędzi (v, u).
- Dla każdego DAG można wyznaczyć kolejność topologiczną, pewne wierzchołki są przed innymi („brak możliwości cofania się").
- DAG są przydatne do modelowania pewnych procesów np. w produkcji.

Typy grafów i pojęcia grafowe (6)

Przykładowy DAG i jego przykładowa kolejność topologiczna

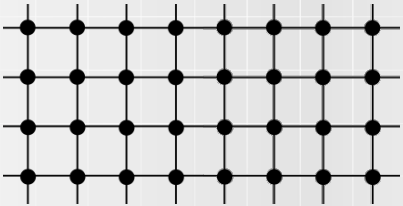




Typy grafów i pojęcia grafowe (8)

Graf typu krata/siatka (grid/lattice graph):

- Graf który po narysowaniu w Euklidesowej przetrzeni 2D ma strukturę regularnych kafelków.
- Kafelki mogą być kwadratowe, trójkątne, sześcienne itd.
- Przydatne do modelowania pewnych procesów np. w produkcji.



Reprezentacje grafu (1)

Różne sposobu przechowywania (struktury) grafu w pamięci komputera:

- Macierz sąsiedztwa (adjacency matrix).
- Lista sąsiedztwa (adjacency list).
- Lista krawędzi (edge list).
- Macierz incydencji (incidence matix).

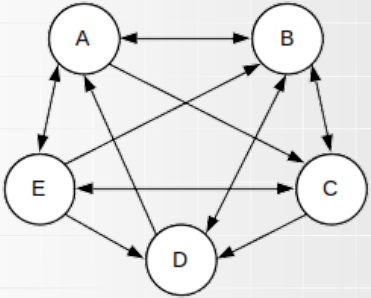
Istotne operacje:

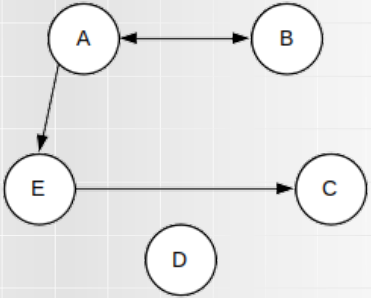
- Dostęp do łuku (u, v) (lub krawędzi $\{u, v\}$).
- Znalezienie (lub przegląd) sąsiadów wierzchołka v.

Reprezentacje grafu (2)

Jak przykłady użyjemy dwóch grafów skierowanych:

- Gęstego $v = 5$, $k = 15$ (75% krawędzi).
- Rzadkiego $v = 5$, $k = 4$ (20% krawędzi).





Macierz sąsiedztwa (1)

- Macierz dwuwymiarowa rozmiaru $n \times n$.
- W indeksie i, j macierzy przechowywana jest informacja o krawędzi (i, j) (lub $\{i, j\}$).
  - Dla grafów bez wag/etykiet przechowujemy jedynie czy krawędź istnieje czy nie.
  - W innym przypadku przechowujemy wagę, etykietę lub wręcz wskaźnik/referencję na obiekt krawędzi (dbając o odpowiednie wartości jeśli krawędź nie istnieje).
- Dla grafów nieskierowanych $\{u, v\} = \{v, u\}$, więc przechowujemy tylko połowę macierzy (np. macierz górnotrójkątna).
- Realizowana tablicą dwuwymiarową, tablicami jednowymiarowymi, tablicami dynamicznymi itd.

Macierz sąsiedztwa (2)

Dla przykładowych grafów:

	A	B	C	D	E
A	0	1	1	0	1
B	1	0	1	1	0
C	0	1	0	1	1
D	1	1	0	0	0
E	1	1	1	1	0

	A	B	C	D	E
A	0	1	0	0	1
B	1	0	0	0	0
C	0	0	0	0	0
D	0	0	0	0	0
E	0	0	1	0	0

Macierz sąsiedztwa (3)

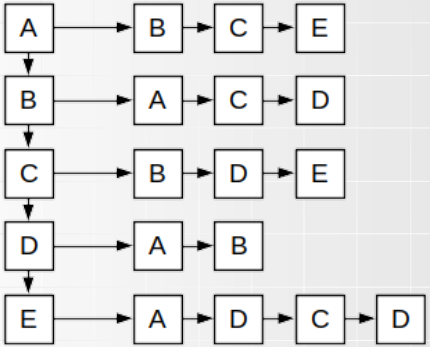
- Dostęp do krawędzi w czasie $O(1)$.
- Przegląd sąsiadów wierzchołka v w czasie $n \in \Theta(n)$.
  - Niezależne od tego ile wynosi $N(v)$, czyli liczba sąsiadów v!
- Potrzebujemy co najmniej n^2 (czyli $O(n^2)$) pamięci.
  - Niezależne od k!
  - Wartości na przekątnej są bezużyteczne, chyba że graf dopuszcza pętle.
- Dość wydajna (pamięciowo i czasowo) dla grafów gęstych.

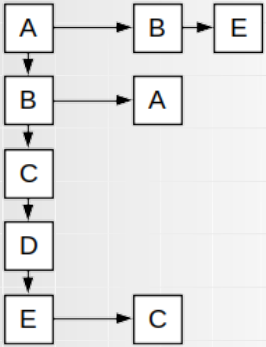
Lista sąsiedztwa (1)

- Lista rozmiaru n.
- Elementami listy są kolejne listy.
- Lista i-ta przechowuje krawędzie zaczynające się w wierzchołku i-tym.
  - Najprościej przechować numer wierzchołka którym krawędź się kończy.
- Ponieważ przechowujemy tylko faktycznych sąsiadów, to każda podlista może mieć różną długość.

Lista sąsiedztwa (2)

Dla przykładowych grafów:





Lista sąsiedztwa (3)

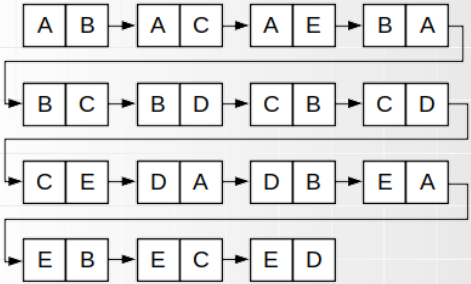
- Lista wierzchołka v ma długość $N(v)$, zamiast n.
  - Przegląd sąsiadów v w czasie $O(N(v))$.
- Zwykle mniejsze zużycie pamięci $O(n + k)$.
  - Wydajna dla grafów rzadkich.
  - Czasami może wymagać więcej pamięci (np. graf pełny plus dopuszczanie pętli).
- Dostęp do krawędzi wierzchołka v w czasie $O(N(v)) \in O(n)$.

Lista krawędzi (1)

- Lista rozmiaru k.
- Elementami listy są krawędzie.
  - Najprościej przechować krawędź jako parę: wierzchołek początkowy i końcowy.
- Zwykle krawędzie przechowywane są w losowej kolejności.
  - Można przyjąć pewien porządek, ale nie ma to większego wpływu na efekt.

Lista krawędzi (2)

Dla przykładowych grafów:



Lista krawędzi (3)

- Lista krawędzi zajmuje $O(k)$ pamięci.
  - Każdy element przechowuję parę, więc łączna pamięć jest zwykle większa niż dla listy sąsiedztwa.
- Dostępu do krawędzi wierzchołka v w czasie $O(k)$.
- Czas przeglądniecia sąsiadów wierzchołka v w czasie $O(k)$.
- Zwykle skrajnie niewydajna dla grafów gęstych.
- Może być przydatna dla niektórych algorytmów (np. zmodyfikowana w kolejkę priorytetową).

Macierz indydencji (1)

- Macierz rozmiaru nk.
- Indeks i, j przechowuje informacje czy j-ta krawędź jest incydentna (łączy się) z wierzchołkiem i. Typowe wartości:
  - 1 – krawędź zaczyna się w tym wierzchołku.
  - -1 – krawędź kończy się w tym wierzchołku.
  - 0 – krawedź nie jest incydentna z wierzchołkiem.
- Zwykle przechowuje tylko informacje o incydencji, ale można też dołączyć inne informacje (waga, etykieta).

Macierz incydencji (2)

Dla przykładowych grafów:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
A	1	1	1	-1	0	0	0	0	0	-1	0	-1	0	0	0
B	-1	0	0	1	1	1	-1	0	0	0	-1	0	-1	0	0
C	0	-1	0	0	-1	0	1	1	1	0	0	0	0	-1	0
D	0	0	0	0	0	-1	0	-1	0	1	1	0	0	0	-1
E	0	0	-1	0	0	0	0	0	-1	0	0	1	1	1	1

	1	2	3	4
A	1	1	-1	0
B	-1	0	1	0
C	0	0	0	-1
D	0	0	0	0
E	0	-1	0	1

Zakładamy kolejność krawędzi taką jak pokazano w liście krawędzi.

Macierz incydencji (3)

- Zajmuje $O(nk)$ pamięci.
  - Czyli rzadko mniej niż $O(n^2)$, nawet dla grafów rzadkich.
  - $\frac{n-2}{n}$ tablicy zawiera zera.
- Dostęp do krawędzi (u, v) w czasie $O(k)$.
- Przeglądnięcie wszystkich sąsiadów w czasie $O(k + nN(v))$.
- Przydatna gdyby krawędź miała więcej niż 2 końce.

Reprezentacje grafu (3)

Reprezentacja	Pamięć	Dostęp do krawędzi	Przegląd sąsiadów
Macierz sąsiedztwa	$O(n^2)$	$O(1)$	$O(n)$
Lista sąsiedztwa	$O(n+k)$	$O(N(v))$	$O(N(v))$
Lista krawędzi	$O(k)$	$O(k)$	$O(k)$
Macierz incydencji	$O(nk)$	$O(k)$	$O(k+nN(v))$

Przeszukiwanie grafu (1)

- Analogicznie jak w przypadku drzew istnieją algorytmy do przeglądu grafu tj. do odwiedzenia wszystkich wierzchołków.
- Podstawowy algorytm:
  - Dodajemy pewien wierzchołek początkowy s do kolejki Q.
  - Dopóki Q nie jest puste:
    - Pobieramy (usuwamy) jeden wierzchołek v z Q.
    - Odwiedzamy v.
    - Dodajemy nieodwiedzonych jeszcze sąsiadów v do Q.
- Węzły mogą być odkrywane wielokrotnie.
- Problem z grafami niespójnymi.

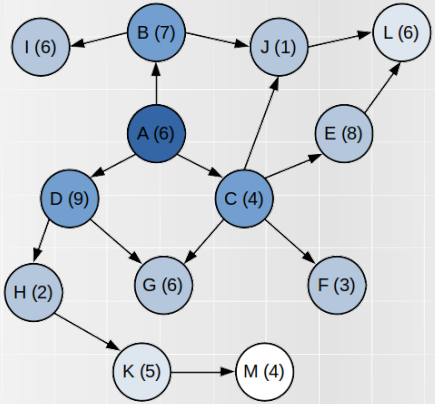
Przeszukiwanie grafu (2)

Podstawowe typy przeglądu grafu:

- Przegląd wgłąb (depth-first) – ostatnio dodany wierzchołek odwiedzany jest najpierw (Q jest stosem).
- Przegląd wszerz (breadth-first) – ostatnio dodany wierzchołek odwiedzany jest na końcu (Q jest kolejką FIFO).
- Przegląd pierwszy najlepszy (best-first) – wierzchołki odwiedzamy wg priorytetu (Q jest kolejką priorytetową).
  - Dla grafów ważonych.
  - Priorytetem jest waga wierzchołka (na podobnej zasadzie działa algorytm Dijkstry) i/lub waga prowadzącej do niego krawędzi (np. algorytm najbliższego sąsiada dla problemu komiwojażera).

Przeszukiwanie grafu (3)

Przykładowy skierowany graf ważony:



Przeszukiwanie grafu (4)

Przykładowy przegląd dla przedstawionego grafu:

- Depth-first: A, D, H, K, M, G, C, J, L, F, E, B, I.
- Breadth-first: A, B, C, D, I, J, E, F, G, H, L, K, M
- Best-first: A, C, J, F, G, L, B, I, E, D, H, K, M.

Zastosowania grafów (1)

- Transport, komunikacja (problem komiwojażera, sieci przepływowe).
- Analiza procesów (maszyny stanów, sieci Petriego).
- Modelowanie relacji/połączeń (social media, rozprzestrzenianie się chorób np. COVID-19).
- Poszukiwanie trasy.
- Optymalizacja produkcji (szeregowanie zadań).
- Inne.