Algorytmy i struktury danych

Temat: Wybrane algorytmy grafowe

Podstawowe pojęcia

- Grafem nazywamy strukturę ***G*** = (***V***, ***E***) złożoną z niepustego zbioru wierzchołków ***V***, zwanych także węzłami oraz zbioru krawędzi ***E***, zwanych inaczej łukami.
- Rozróżnia się grafy skierowane (ang. *directed graph*), zwane też grafami zorientowanymi lub krócej digrafami i grafy nieskierowane (ang. *undirected graph*), zwane również grafami niezorientowanymi.
- W grafie skierowanym krawędzie można opisać jako uporządkowane pary wierzchołków (*u*, *v*); w grafie nieskierowanym jako zbiory {u, v}
- Wierzchołek *u* nazywamy początkiem krawędzi, a *v jej* końcem.
- Mówimy, że krawędź (*u*, *v*) biegnie od wierzchołka *u* do wierzchołka *v*, a także że wierzchołki *u* i *v* są sąsiednimi, sąsiadującymi lub incydentnymi.
- Krawędź, która rozpoczyna się i kończy w tym samym wierzchołku, nazywamy pętlą.
- Krawędź (*u*, *v*) jest często zapisywana jako $u \rightarrow v$ i rysowana w postaci zakończonego strzałką odcinka lub łuku łączącego oba wierzchołki.
- Oznaczenia: |**V**| = n, |**E**| = m

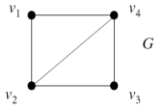
Rodzaje grafów

- Grafy proste
- Multigrafy
- Grafy skierowane (digrafy)
- Grafy ważone
- Hipergrafy

Określenie grafu

- *Graf prosty* $G=(V,E)$ jest uporządkowaną parą dwóch zbiorów: zbioru wierzchołków V oraz zbioru krawędzi $E \subseteq V \times V$.
- Graf prosty nie zawiera krawędzi postaci $\{u,u\}$ oraz pomiędzy każdą parą wierzchołków istnieje co najwyżej jedna krawędź.
- Wierzchołki *u* i *v* są *sąsiednie*, gdy $\{u,v\} \in E$.
- Wierzchołek *u* oraz krawędź *e* są *incydentne*, gdy $u \in e$.

Przykład:

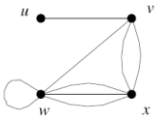


$G = (\{ v_1, v_2, v_3, v_4 \},$
$\{ \{v_1, v_2\}, \{v_2, v_3\}, \{v_3, v_4\}, \{v_4, v_1\}, \{v_2, v_4\} \})$

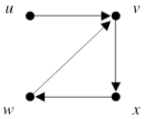
Określenie grafu

- *Multigraf* to graf, w którym pomiędzy dowolną parą wierzchołków może wystąpić więcej niż jedna krawędź oraz dopuszczalne są *pętle*, tzn. krawędzie postaci $\{v,v\}$, gdzie $v \in V$.
- Graf nazywamy *digrafem* (*grafem skierowanym*), gdy krawędź łącząca *u* oraz *v* jest parą uporządkowaną postaci (u,v).

Przykład:



$G = (\{u, v, w, x\},$
$\{\{u,v\}, \{w,v\}, \{w,w\}, \{w,x\},$
$\{w,x\}, \{w,x\}, \{x,v\}, \{x,v\}\})$



$D = (\{u, v, w, x\},$
$\{(u,v), (w,v), (v,x), (x,w)\})$

Grafy ważone

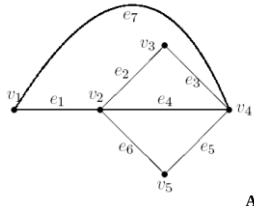
- Niekiedy krawędziom grafu przypisuje się pewne wartości, zwane wagami lub etykietami, a wtedy graf nazywamy ważonym lub etykietowanym.
- Przykładem rzeczywistego grafu ważonego jest graf przedstawiający sieć połączeń drogowych lub lotniczych, w którym wierzchołkami są miejscowości, a wagami odległości między nimi, czas przelotu lub koszty podróży.

Reprezentacje grafów

Macierz przyległości (sąsiedztwa)

- Dany jest graf G = (V, E), przy czym V = {v_1, v_2, . . . , v_n} jest zbiorem wierzchołków.
- Wówczas macierz A(G) = $[a_{ij}]_{n\times n}$, gdzie a_{ij} jest liczbą krawędzi łączących wierzchołki v_i oraz v_j nazywa się macierzą przyległości grafu G.
- W przypadku grafu skierowanego a_{ij} jest liczbą łuków z wierzchołka v_i do v_j .

Macierz przyległości (sąsiedztwa)



$$\mathbf{A(G)} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{pmatrix} \end{array}$$

Własności macierzy przyległości

- Macierz przyległości (sąsiedztwa) A(G) jest dla grafu prostego macierzą binarną
- Macierz przyległości wymaga $|V|^2 = n^2$ bitów pamięci
- Jeżeli w jest długością słowa maszynowego, to każdy wiersz macierzy przyległości można zapisać jako ciąg n bitów zajmujących $\lceil n/w \rceil$ słów maszynowych, gdzie $\lceil x \rceil$ oznacza najmniejszą liczbę całkowitą nie mniejszą niż x
- Implementacja macierzy przyległości wymaga zatem $n\lceil n/w \rceil$ słów
- Dla grafu prostego: macierz A(G) jest symetryczna i wymaga $n(n-1)/2$ bitów
- Reprezentacja macierzowa jest korzystna dla tzw. grafów gęstych tj. takich, dla których $|E| >> |V|^2$

Własności macierzy przyległości

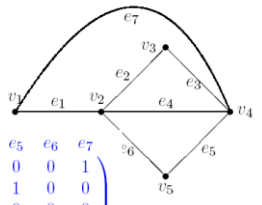
- Reprezentacja macierzowa grafu jest wygodna w obliczeniach (łatwo jest sprawdzić, czy pomiędzy dwoma dowolnymi wierzchołkami grafu istnieje krawędź).
- Czas dostępu do elementów macierzy jest stały, niezależnie od liczby wierzchołków i krawędzi.
- Podstawową wadą macierzy sąsiedztwa jest wysoka złożoność pamięciowa oraz niedogodność reprezentowania grafu o zmiennej liczbie wierzchołków.
- Zapis grafu o *n* wierzchołkach wymaga n^2 komórek pamięci i jest szczególnie niekorzystny w przypadku grafów rzadkich, w których liczba krawędzi *m* jest mała w porównaniu z wartością n^2.
- Elementy macierzy można pamiętać po 8 w bajcie, co pozwala istotnie zredukować zapotrzebowanie na pamięć , ale nieco utrudnia dostęp do nich.

Macierz incydencji

- Dany jest graf G = (V, E), przy czym V = {v_1, v_2, . . . , v_n} jest zbiorem wierzchołków, natomiast E = {e_1, e_2, . . . , e_m} jest zbiorem krawędzi grafu.
- Wówczas macierz M(G) = $[m_{ij}]_{n\times m}$, gdzie liczba $m_{ij} \in \{0, 1, 2\}$ oznacza ile razy wierzchołek v_i oraz krawędź e_j są incydentne (2 występuje w przypadku pętli) jest macierzą incydencji grafu G.

Macierz incydencji

Przykład



$$M(G) = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{array} \begin{array}{c} \begin{array}{ccccccc} e_1 & e_2 & e_3 & e_4 & e_5 & e_6 & e_7 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 \end{pmatrix} \end{array}$$

Własności macierzy incydencji

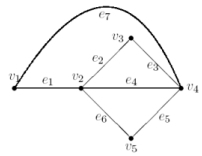
- Macierz incydencji wymaga n·m bitów pamięci, co może być liczbą większą niż n^2 bitów zajmowanych przez macierz przyległości, ponieważ liczba krawędzi m=|E| jest często większa niż liczba wierzchołków n= |V|
- W niektórych jednak przypadkach może być korzystniejsze użycie macierzy incydencji niż macierzy przyległości, pomimo zwiększonej zajętości pamięci.
- Macierze incydencji są szczególnie dogodne przy modelowaniu obwodów elektrycznych i układów przełączających.

Lista krawędzi

- Innym, często stosowanym sposobem reprezentacji grafu jest wypisanie wszystkich jego krawędzi jako par wierzchołków.
- Przykład:

  {v1, v2}, {v1, v4}, {v2, v3}, {v2, v4}, {v2, v5}, {v3, v4}, {v4, v5}.

- Dla grafu skierowanego, byłyby to uporządkowane pary wierzchołków odpowiadające łukom.



Własności listy krawędzi

- Liczba bitów potrzebnych do zaetykietowania (etykietami od *1* do *n*) wierzchołków grafu G jest równa **b**, gdzie

  $2^{b-1} < n \leq 2^b$, czyli $b = \lfloor \log_2 n \rfloor + 1$

  np. dla n=30 b=5 (11110)
- Całkowita zajętość pamięci jest równa 2bm bitów
- Ten sposób reprezentacji jest bardziej ekonomiczny niż macierz przyległości, jeżeli 2bm < n^2
- Reprezentacja "listowa" jest korzystniejsza dla grafów rzadkich tj. takich, dla których |E| << |V| ^{2}
- Przechowywanie i przekształcanie grafu w komputerze jest trudniejsze (na przykład przy badaniu spójności grafu)

Dwie tablice liniowe

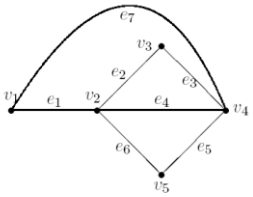
- Modyfikacją listy krawędzi jest przedstawienie grafu za pomocą dwóch tablic jednowymiarowych (początków i końców krawędzi):

  $F = (f_1, f_2, \ldots, f_m)$

  $H = (h_1, h_2, \ldots, h_m)$
- Elementy obu tablic: etykiety wierzchołków, m = |**E**|
- **G** graf skierowany: łuk e_i prowadzi od wierzchołka f_i, do wierzchołka h_i
- **G** graf nieskierowany: krawędź e_i łączy f_i i h_i
- Dogodna reprezentacja do sortowania w grafach ważonych

Dwie tablice liniowe

Przykład



F = (v1, v2, v2, v3, v2, v4, v1),
H = (v2, v3, v5, v4, v4, v5, v4).

Lista wierzchołków sąsiednich (następników)

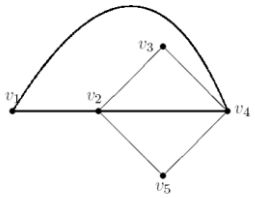
- Efektywna metoda reprezentacji grafów, stosowana w przypadku gdy stosunek |E| / |V| nie jest duży
- Dla każdego wierzchołka **v** tworzymy listę (tablicę jednowymiarową), której pierwszym elementem jest **v**, a pozostałymi elementami są:
  - wierzchołki będące sąsiadami wierzchołka **v** (w przypadku grafu nieskierowanego)
  - bezpośredni następnicy wierzchołka **v**, tzn. wierzchołki, do których istnieje łuk z wierzchołka **v** (w przypadku grafu skierowanego)

Lista wierzchołków sąsiednich (następników)

- Lista sąsiedztwa jest strukturą danych, w której występuje n elementowy wektor N zwany nagłówkiem taki, że $N[i]$ jest wskaźnikiem na listę zawierającą sąsiadów wierzchołka v_i.
- Zapotrzebowanie na pamięć list sąsiedztwa jest proporcjonalne do sumy liczby wierzchołków i liczby krawędzi, wynosi więc $O(m+n)$.
- Jednak sprawdzenie, czy istnieje krawędź (u, v), wymaga przeglądania listy sąsiedztwa wierzchołka u w poszukiwaniu wierzchołka v, toteż jego pesymistyczna złożoność czasowa wynosi $O(n)$.
- W przypadku macierzy sąsiedztwa takie sprawdzenie wykonuje się natychmiastowo, tj. w czasie $O(1)$, gdyż wymaga jedynie dostępu do wartości $A[u, v]$.

Lista wierzchołków sąsiednich (następników)

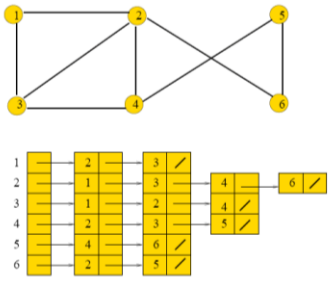
Przykład 1



v1 : v2, v4
v2 : v1, v3, v4, v5
v3 : v2, v4
v4 : v1, v2, v3, v5
v5 : v2, v4

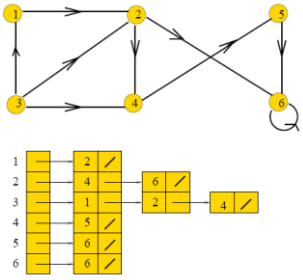
Lista wierzchołków sąsiednich (następników)

Przykład 2



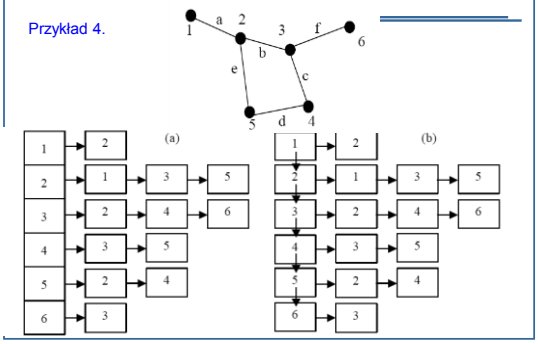
Lista wierzchołków sąsiednich (następników)

Przykład 3



Lista wierzchołków sąsiednich (następników)

Przykład 4.

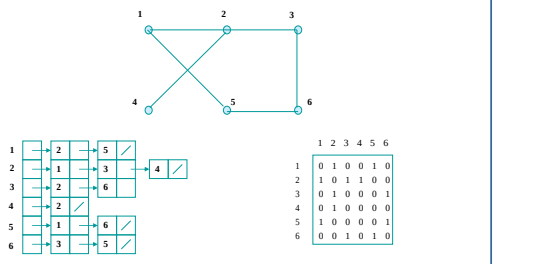


Reprezentacje grafów

W praktyce do implementacji grafów najczęściej wykorzystuje się:

- macierz przyległości (sąsiedztwa),
- listę wierzchołków sąsiednich (następników).

Reprezentacja w pamięci grafów nieskierowanych



a) Listy wierzchołków sąsiednich **b) Macierz sąsiedztwa grafu**

Algorytmy grafowe

Z grafami związanych jest wiele algorytmów, np.:

- Przeszukiwanie grafu (odwiedzania wszystkich wierzchołków)
  - algorytm BFS (przeszukiwanie wszerz)
  - algorytm DFS (przeszukiwanie w głąb))
- Wyznaczanie minimalnego drzewa rozpinającego
  - algorytm Kruskala
  - algorytm Prima
- Wyszukiwanie najkrótszych dróg z jednym źródłem
  - algorytm Dijkstry
  - algorytm Bellmana-Forda
- Wyszukiwanie najkrótszych dróg pomiędzy parą wierzchołków
  - algorytm Floyda-Warshalla
  - algorytm Johnsona (dla grafów rzadkich)