

PODSTAWY INFORMATYKI

25/11/2019

WFAiS UJ, Informatyka Stosowana
I rok studiów, I stopień

Wykład 8

2

Modele
danych: grafy

- **Podstawowe pojęcia**
- **Grafy wywołań**
- **Grafy skierowane i nieskierowane**
- **Grafy planarne, kolorowanie grafów**
- **Implementacja grafów**
 - ▣ **Listy sąsiedztwa**
 - ▣ **Macierze sąsiedztwa**
- **Składowe spójne, algorytm Kruskala**
- **Sortowanie topologiczne**
- **Najkrótsze drogi: algorytm Dikstry i Floyda-Warshalla**

Graf

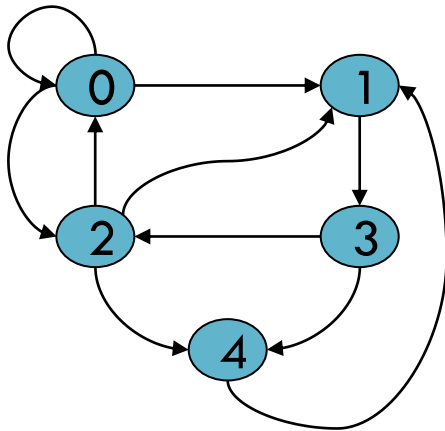
3

- **Graf to jest relacja binarna.**
- **Dla grafów mamy możliwość wizualizacji jako zbiór punktów (zwanymi wierzchołkami) połączonych liniami lub strzałkami (nazwanymi krawędziami). Pod tym względem graf stanowi uogólnienie drzewiastego modelu danych.**
- **Podobnie jak drzewa, grafy występują w różnych postaciach: grafów skierowanych i nieskierowanych lub etykietowanych i nieetykietowanych.**
- **Grafy są przydatne do analizy szerokiego zakresu problemów: obliczanie odległości, znajdowanie cykliczności w relacjach, reprezentacji struktury programów, reprezentacji relacji binarnych, reprezentacji automatów i układów elektronicznych.**

Podstawowe pojęcia

4

- **Graf skierowany** (ang. **directed graph**)
Składa się z następujących elementów:
 - Zbioru **V** wierzchołków (ang. *nodes, vertices*)
 - Relacji binarnej **E** na zbiorze **V**. Relacje **E** nazywa się zbiorem krawędzi (ang. *edges*) grafu skierowanego. Krawędzie stanowią zatem pary wierzchołków (**u,v**).



$$V = \{0,1,2,3,4\}$$

$$E = \{ (0,0), (0,1), (0,2), (1,3), (2,0), (2,1), (2,4), (3,2), (3,4), (4,1) \}$$

Podstawowe pojęcia

5

□ Etykiety:

- Podobnie jak dla drzew, dla grafów istnieje możliwość przypisania do każdego wierzchołka etykiety (ang. label).
- Nie należy mylić nazwy wierzchołka z jego etykietą. Nazwy wierzchołków muszą być niepowtarzalne, ale kilka wierzchołków może być oznaczonych tą samą etykietą.



□ Drogi:

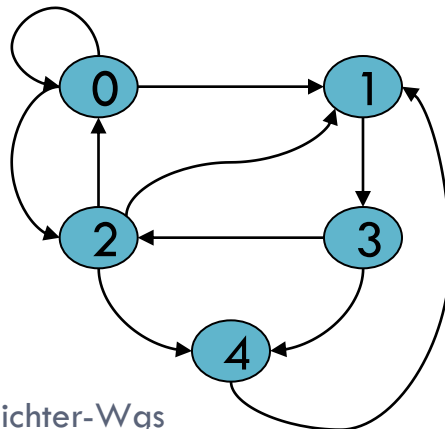
- Droga (ang. path) w grafie skierowanym stanowi listę wierzchołków, (n_1, n_2, \dots, n_k) taką, że występuje krawędź łącząca każdy wierzchołek z następnym, to znaczy $(n_i, n_{i+1}) \in E$ dla $i=1, 2, \dots, k$. Długość (ang. length) drogi wynosi $k-1$, co stanowi liczbę krawędzi należących do tej samej drogi.

Podstawowe pojęcia

6

□ Grafy **cykliczne** i **acykliczne**:

- **Cykl** (ang. cycle) w grafie skierowanym jest drogą o długości 1 lub więcej, która zaczyna się i kończy w tym samym wierzchołku.
- **Długość cyklu** jest długością drogi. **Cykl jest prosty** (ang. simple) jeżeli żaden wierzchołek (oprócz pierwszego) nie pojawia się na nim więcej niż raz.
- Jeżeli istnieje cykl nieprosty zawierający wierzchołek n , to można znaleźć prosty cykl który zawiera n . Jeżeli graf posiada jeden lub więcej cykli to mówimy że jest grafem **cyklicznym** (ang. cyclic). Jeśli cykle nie występują to, graf określa się mianem **acyklicznego** (ang. acyclic).



Przykłady cykli prostych:

$(0,0)$, $(0,2,0)$, $(1,3,2,1)$, $(1,3,2,4,1)$

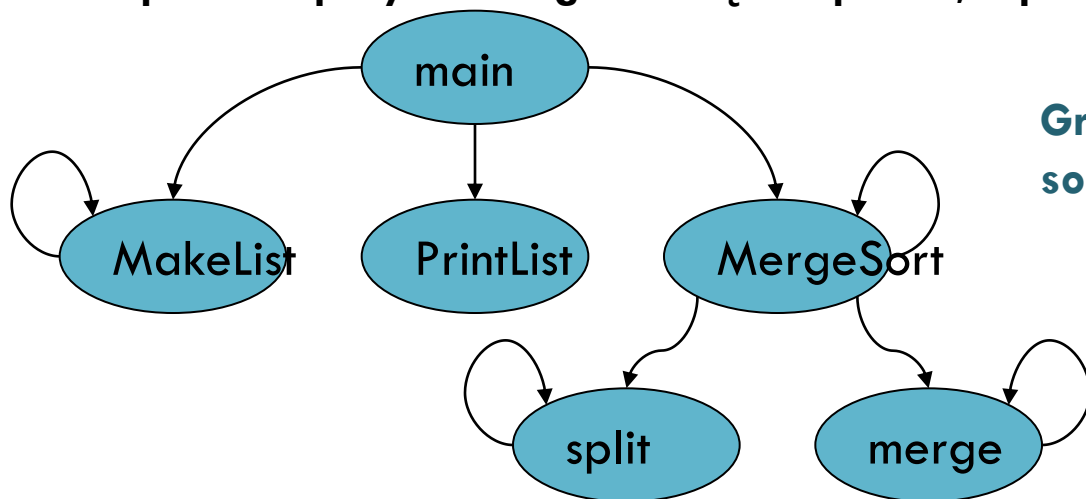
Przykład cyklu nieprostego:

$(0,2,1,3,2,0)$

Grafy wywołań

7

- Wywołania dokonywane przez zestaw funkcji można reprezentować za pomocą grafu skierowanego, zwanego grafem wywołań. Jego wierzchołki stanowią funkcje, a krawędź (P, Q) istnieje wówczas, gdy funkcja P wywołuje funkcję Q .
- Istnienie cyklu w grafie implikuje występowanie w algorytmie rekurencji.
- Rekurencja w której funkcja wywołuje samą siebie nazywamy bezpośrednią (ang. direct).
- Czasem mamy do czynienia z rekurencją pośrednią (ang. indirect) która reprezentuje cykl o długości większej niż 1, np. (P, Q, R, P) .



Graf wywołań dla algorytmu sortowania przez scalanie

Rekurencja bezpośrednia

Grafy nieskierowane

8

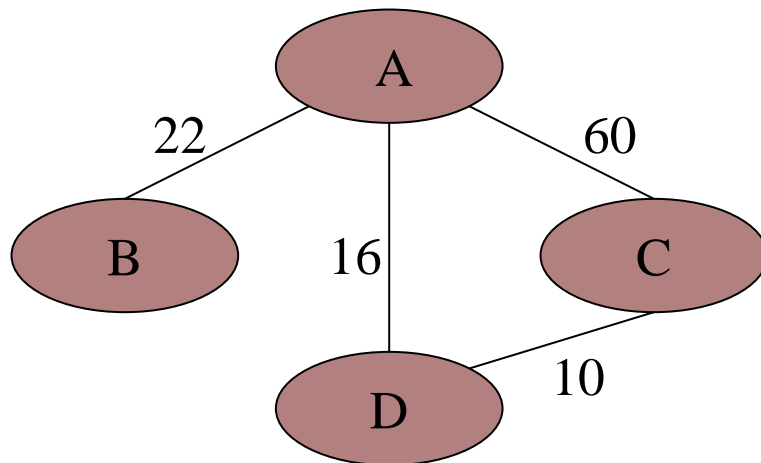
- Czasem zasadne jest połączenie wierzchołków krawędziami, które nie posiadają zaznaczonego kierunku. Z formalnego punktu widzenia taka krawędź jest zbiorem dwóch wierzchołków.
- Zapis $\{u,v\}$ mówi ze wierzchołki u oraz v są połączone w dwóch kierunkach. Jeśli $\{u,v\}$ jest krawędzią nieskierowaną, wierzchołki u i v określa się jako **sąsiednie** (ang. *adjacent*) lub mianem **sąsiadów** (ang. *neighbors*).
- Graf zawierający krawędzie nieskierowane, czyli graf z relacją symetryczności krawędzi, nosi nazwę grafu **nieskierowanego** (ang. *undirected graph*).

Grafy nieskierowane

9

- **Droga to lista wierzchołków. Nieco trudniej jest sprecyzować co to jest cykl, tak aby nie była to każda lista**

$(v_1, v_2, \dots, v_{k-1}, v_k, v_{k-1}, \dots, v_2, v_1)$

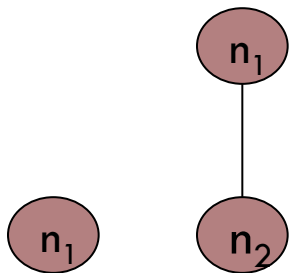


Graf nieskierowany reprezentujący drogi.

Pewne pojęcia z teorii grafów

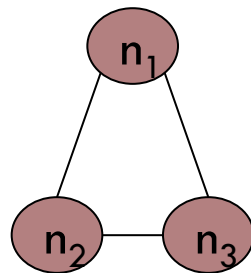
10

- **Teoria grafów** jest dziedziną matematyki zajmującą się właściwościami grafów.
- **Grafy pełne:**
 - **Nieskierowany graf** posiadający krawędzie pomiędzy każdą parą różnych wierzchołków nosi nazwę grafu pełnego (ang. *complete graph*). Graf pełny o n wierzchołkach oznacza się przez K_n .
 - Liczba krawędzi w nieskierowanym grafie K_n wynosi $n(n-1)/2$, w skierowanym grafie K_n wynosi n^2 .

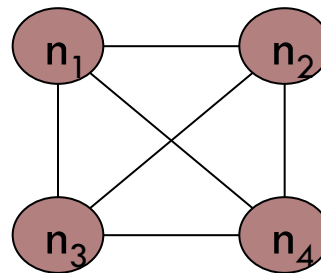


K_1

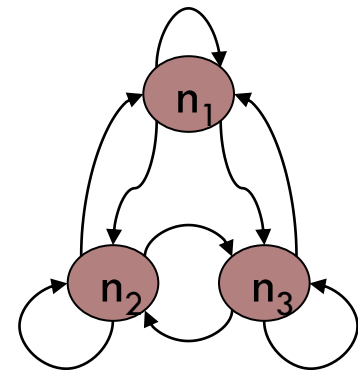
K_2



K_3



K_4



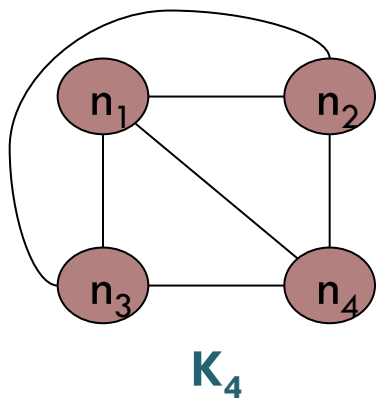
K_3

Grafy planarne i nieplanarne

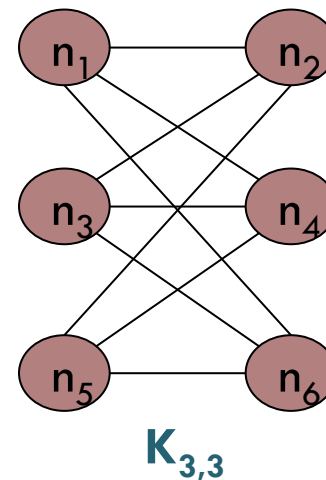
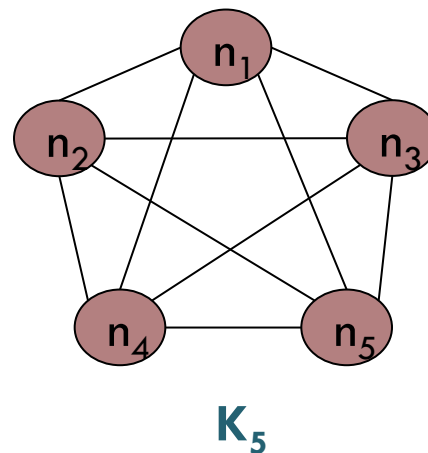
11

- O grafie nieskierowanym mówi się że jest planarny (ang. planar) wówczas, gdy istnieje możliwość rozmieszczenia jego wierzchołków na płaszczyźnie, a następnie narysowania jego krawędzi jako linii ciągłych które się nie przecinają.
- Grafy nieplanarne (ang. nonplanar) to takie które nie posiadają reprezentacji płaskiej.

Reprezentacja planarna:



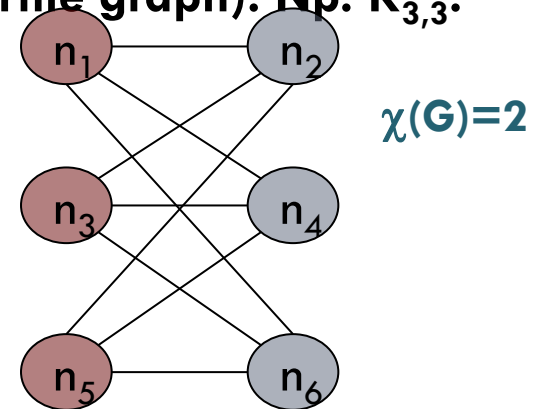
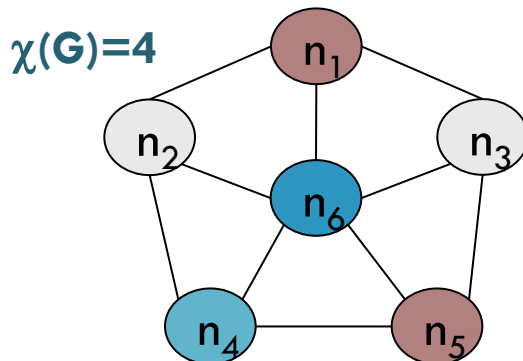
Najprostsze grafy nieplanarne:



Kolorowanie grafów

12

- Kolorowanie grafu (ang. graph coloring) polega na przypisaniu do każdego wierzchołka pewnego koloru, tak aby żadne dwa wierzchołki połączone krawędzią nie miały tego samego koloru.
- Minimalna liczba kolorów potrzebna do takiej operacji nazwana jest liczbą chromatyczną grafu (ang. chromatic number), oznaczaną $c(G)$.
 - ▣ Jeżeli graf jest pełny to jego liczba chromatyczna jest równa liczbie wierzchołków
 - ▣ Jeżeli graf możemy pokolorować przy pomocy dwóch kolorów to nazywamy go dwudzielnym (ang. bipartite graph). Np. $K_{3,3}$.



Implementacje grafów

13

- Istnieją dwie standardowe metody reprezentacji grafów:
 - ▣ **Listy sąsiedztwa** (ang. *adjacency lists*), jest, ogólnie rzecz biorąc, podobna do implementacji relacji binarnych.
 - ▣ **Macierze sąsiedztwa** (ang. *adjacency matrices*), to nowy sposób reprezentowania relacji binarnych, który jest bardziej odpowiedni dla relacji, w przypadku którym liczba istniejących par stanowi znaczącą część całkowitej liczby par, jakie mogłyby teoretycznie istnieć w danej dziedzinie.

Listy sąsiedztwa

14

- Wierzchołki są ponumerowane kolejnymi liczbami całkowitymi **0,1,....., MAX-1** lub oznaczone za pomocą innego adekwatnego typu wyliczeniowego (używamy poniżej typu **NODE** jako synonimy typu wyliczeniowego).
- Wówczas można skorzystać z podejścia opartego na wektorze własnym.
- Element **successors[u]** zawiera wskaźnik do listy jednokierunkowej wszystkich bezpośrednich następników wierzchołka **u**. Następniki mogą występować w dowolnej kolejności na liście jednokierunkowej.

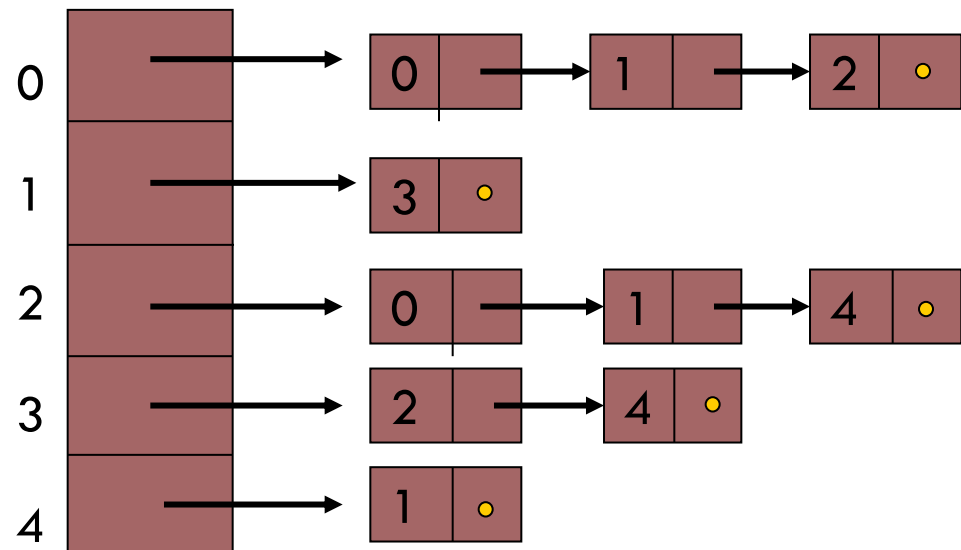
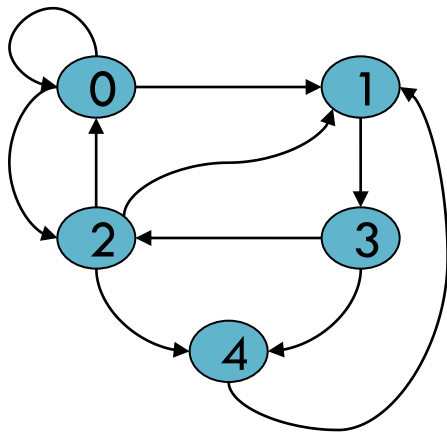
Listy sąsiedztwa:

```
typedef struct CELL *LIST;
struct CELL {
    NODE nodeName;
    LIST next;
}
LIST successors[MAX]
```

Listy sąsiedztwa

15

- Listy sąsiedztwa zostały posortowane wg. kolejności, ale następniki mogą występować w **dowolnej kolejności** na odpowiedniej liście sąsiedztwa.



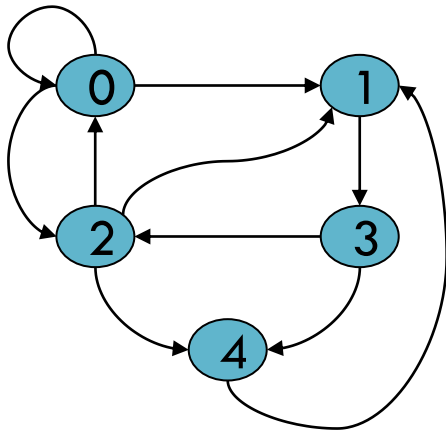
Macierz sąsiedztwa

16

Tworzymy dwuwymiarową tablicę;

BOOLEAN `vertices[MAX][MAX]`;

w której element `vertices[u][v]` ma wartość **TRUE** wówczas, gdy istnieje krawędź (u, v) , zaś **FALSE**, w przeciwnym przypadku.



	0	1	2	3	4
0	1	1	1	0	0
1	0	0	0	1	0
2	1	1	0	0	1
3	0	0	1	0	1
4	0	1	0	0	0

Listy sąsiedztwa a macierz sąsiedztwa

17

- **Macierze sąsiedztwa** są preferowanym sposobem reprezentacji grafów wówczas, gdy **grafy są gęste** (ang. dense), to znaczy, kiedy liczba krawędzi jest bliska maksymalnej możliwej ich liczby.
- Dla grafu skierowanego o n wierzchołkach maksymalna liczba krawędzi wynosi n^2 .
- Jeśli **graf jest rzadki** (ang. sparse) to reprezentacja oparta na **listach sąsiedztwa** może pozwolić zaoszczędzić pamięć.
- Istotne różnice między przedstawionymi reprezentacjami grafów są widoczne już przy wykonywaniu prostych operacji.

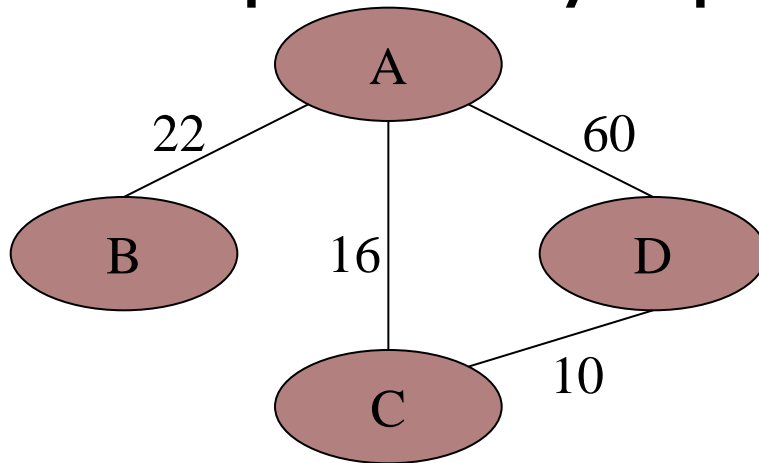
Preferowany sposób reprezentacji:

OPERACJA	GRAF GĘSTY	GRAF RZADKI
Wyszukiwanie krawędzi	Macierz sąsiedztwa	Obie
Znajdowanie następników	Obie	Lista sąsiedztwa
Znajdowanie poprzedników	Macierz sąsiedztwa	Obie

Składowa spójna grafu nieskierowanego

18

- Każdy graf nieskierowany można podzielić na jedną lub większą liczbę **spójnych składowych** (ang. *connected components*).
- Każda spójna składowa to taki zbiór wierzchołków, że dla każdych dwóch z tych wierzchołków istnieje łącząca je ścieżka. Jeżeli graf składa się z jednej spójnej składowej to mówimy że jest **spójny** (ang. *connected*).



To jest graf spójny

Algorytm wyznaczania spójnych składowych

19

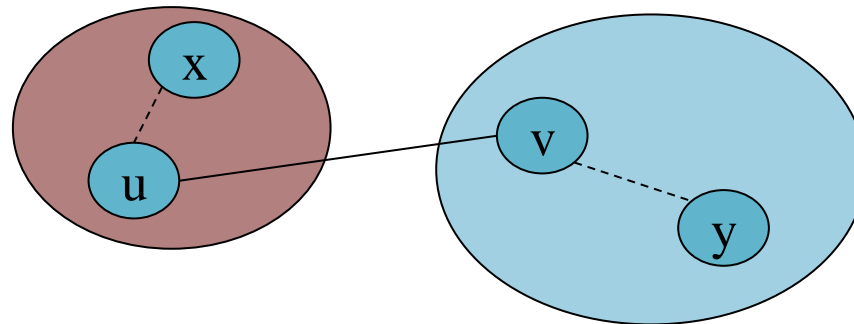
Chcemy określić spójne składowe grafu G . Przeprowadzamy rozumowanie indukcyjne.

□ **Podstawa:**

- Graf G_0 zawiera jedynie wierzchołki grafu G i żadnej jego krawędzi. Każdy wierzchołek stanowi odrębną spójną składową.

□ **Indukcja:**

- Zakładamy, że znamy już spójne składowe grafu G_i po rozpatrzeniu pierwszych i krawędzi, a obecnie rozpatrujemy $(i+1)$ krawędź $\{u, v\}$.
 - jeżeli wierzchołki u, v należą do jednej spójnej składowej to nic się nie zmienia
 - jeżeli do dwóch różnych, to łączymy te dwie spójne składowe w jedną.



Struktura danych dla wyznaczania spójnych składowych

20

- Biorąc pod uwagę przedstawiony algorytm, musimy zapewnić szybką wykonywalność następujących operacji:
 - 1) gdy jest określony wierzchołek to znajdź jego bieżącą spójną składową
 - 2) połącz dwie spójne składowe w jedną
- Dobre wyniki daje ustawienie wierzchołków każdej składowej w strukturze drzewiastej, gdzie spójna składowa jest reprezentowana przez korzeń.
 - aby wykonać operacje (1) należy przejść do korzenia: $O(\log n)$
 - aby wykonać operacje (2) wystarczy korzeń jednej składowej określić jako potomka korzenia składowej drugiej ($O(1)$).
 - Przyjmijmy zasadę ze korzeń drzewa o mniejszej wysokości czynimy potomkiem.
- Wyznaczenie wszystkich spójnych składowych $O(m \log n)$, m -krawędzi i n -wierzchołków.

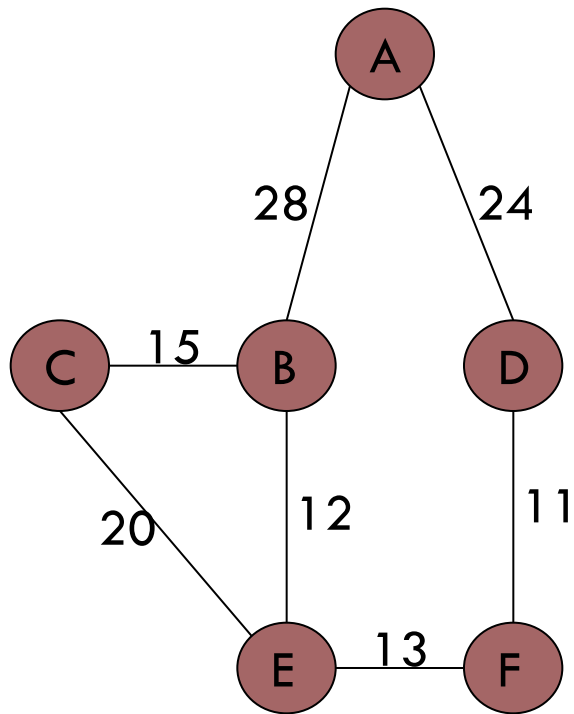
Minimalne drzewa rozpinające

21

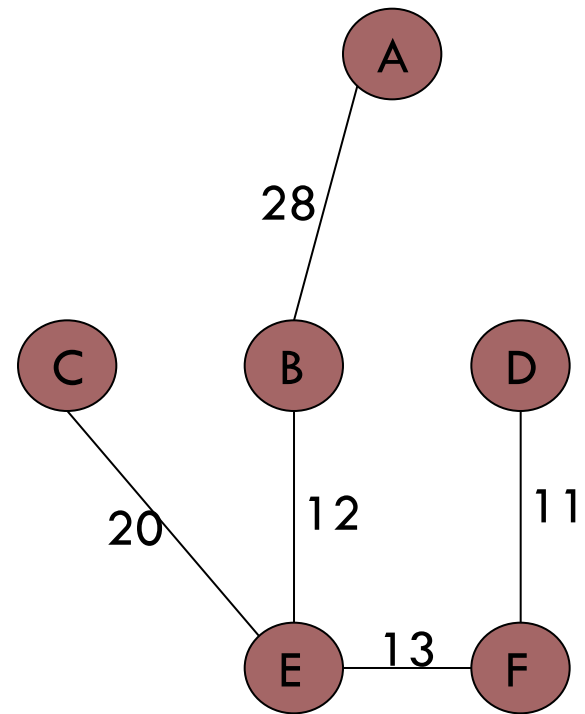
- **Drzewo rozpinające** (ang. *spanning tree*) grafu nieskierowanego G stanowi zbiór wierzchołków tego grafu wraz z podzbiorem jego krawędzi, takich że:
 - łączą one wszystkie wierzchołki, czyli istnieje droga między dwoma dowolnymi wierzchołkami która składa się tylko z krawędzi drzewa rozpinającego.
 - tworzą one drzewo nie posiadające korzenia, nieuporządkowane. Oznacza to że nie istnieją żadne (proste) cykle.
- Jeśli graf G stanowi pojedynczą spójną składową to drzewo rozpinające zawsze istnieje. **Minimalne drzewo rozpinające** (ang. *minimal spanning tree*) to drzewo rozpinające, w którym suma etykiet jego krawędzi jest najmniejsza ze wszystkich możliwych do utworzenia drzew rozpinających tego grafu.

Minimalne drzewa rozpinające

22



Graf nieskierowany



Drzewo rozpinające

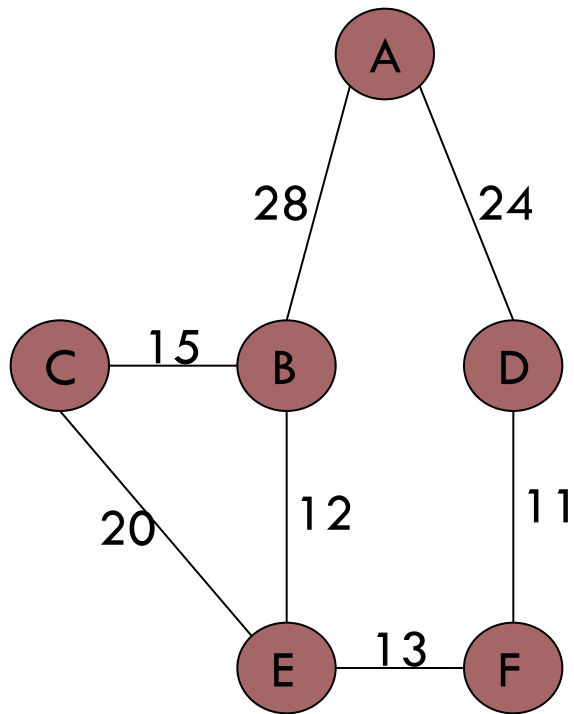
Algorytm Kruskala

23

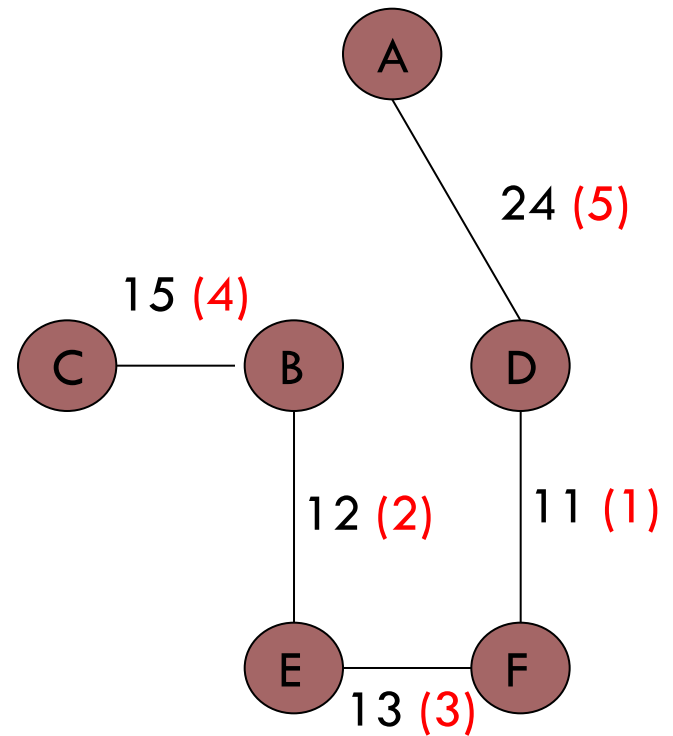
- **Istnieje wiele algorytmów do znajdowania minimalnego drzewa rozpinającego.**
- **Jeden z nich to algorytm Kruskala, który stanowi proste rozszerzenie algorytmu znajdowania spójnych składowych. Wymagane zmiany to:**
 - ▣ **należy rozpatrywać krawędzie w kolejności zgodnej z rosnącą wartością ich etykiet,**
 - ▣ **należy dołączyć krawędź do drzewa rozpinającego tylko w takim wypadku gdy jej końce należą do dwóch różnych spójnych składowych.**

Algorytm Kruskala

24



Graf nieskierowany



Minimalne drzewo rozpinające
(w nawiasach podano kolejność dodawanych krawędzi)

Algorytm Kruskala

25

- Algorytm Kruskala jest dobrym przykładem **algorytmu zachłannego** (ang. greedy algorithm), w przypadku którego podejmowany jest szereg decyzji, z których każdą stanowi wybranie opcji najlepszej w danym momencie.
 - Lokalnie podejmowane decyzje polegają w tym przypadku na wyborze krawędzi dodawanej do formowanego drzewa rozpinającego.
 - Za każdym razem wybierana jest krawędź o najmniejszej wartości etykiety, która nie narusza definicji drzewa rozpinającego, zabraniającej utworzenia cyklu.
- Dla algorytmu Kruskala można wykazać, że jego rezultat jest optymalny globalnie, to znaczy że daje on w wyniku drzewo rozpinające o minimalnej wadze.
- Czas wykonania algorytmu jest $O(m \log m)$ gdzie m to jest większa z wartości liczby wierzchołków i liczby krawędzi.

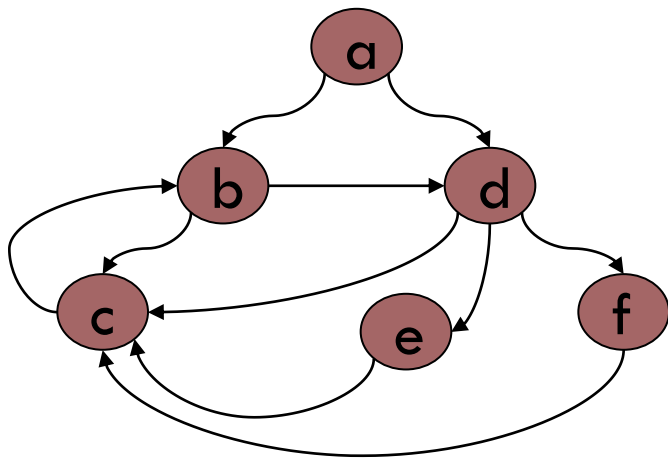
Algorytm przeszukiwania w głąb

26

- Jest to podstawowa metoda badania grafów skierowanych.
- Bardzo podobna do stosowanych dla drzew, w których startuje się od korzenia i rekurencyjnie bada wierzchołki potomne każdego odwiedzonego wierzchołka.
- Trudność polega na tym że w grafie mogą pojawiać się cykle... Należy wobec tego znaczyć wierzchołki już odwiedzone i nie wracać więcej do takich wierzchołków.
 - Z uwagi na fakt, że w celu uniknięcia dwukrotnego odwiedzenia tego samego wierzchołka jest on odpowiednio oznaczany, graf w trakcie jego badania zachowuje się podobnie do drzewa.
- W rzeczywistości można narysować drzewo, którego krawędzie rodzic-potomek będą niektórymi krawędziami przeszukiwanego grafu G .
 - Takie drzewo nosi nazwę **drzewa przeszukiwania w głąb** (ang. depth-first-search) dla danego grafu.

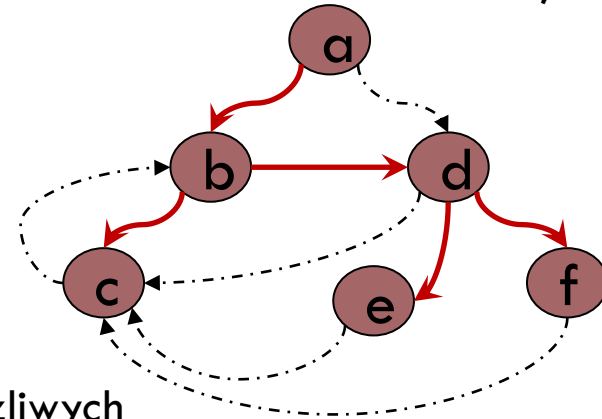
Drzewa przeszukiwania w głąb

27

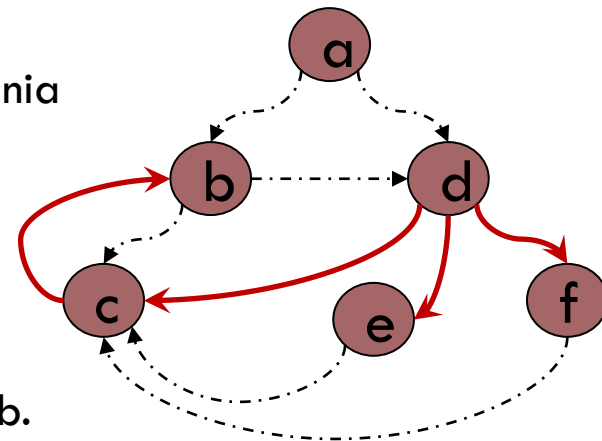


Graf skierowany

Las przeszukiwania:
dwa drzewa o korzeniach a, d



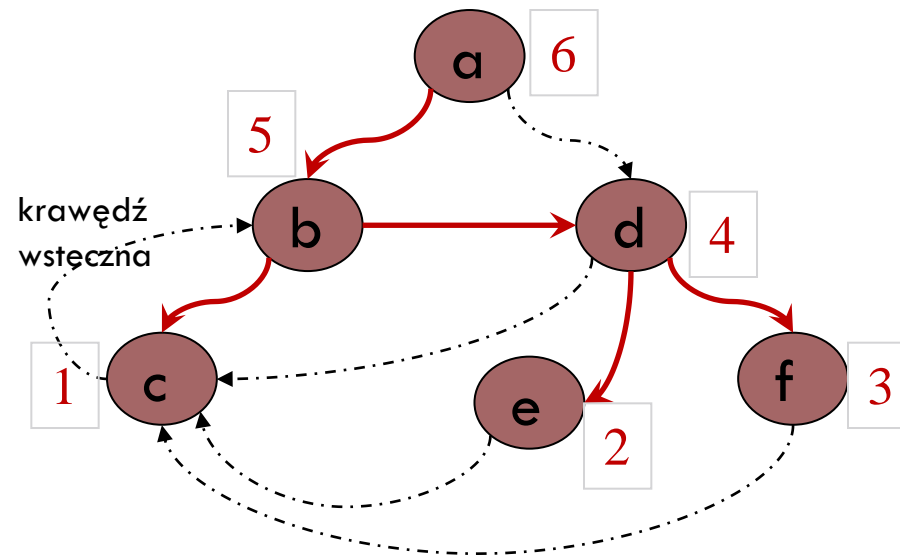
Jedno z możliwych
drzew
przeszukiwania



Las przeszukiwania w głąb.

Drzewo przeszukiwania w głąb

28



- Po (podczas) konstruowaniu drzewa przeszukiwania w głąb można ponumerować jego wierzchołki w **kolejności wstecznej** (ang. *post-order*).

Cykle w grafie skierowanym

29

- Podczas przeszukiwania w głąb grafu skierowanego G można wszystkim wierzchołkom przypisać numery zgodne z **kolejnością wsteczną** w czasie rzędu $O(m)$.
- **Krawędzie wsteczne** to takie dla których początki są równe lub mniejsze końcom ze względu na numerację wsteczną.
Zawsze gdy istnieje krawędź wsteczna w grafie musi istnieć cykl.
- Prawdziwe jest również twierdzenie odwrotne. Aby stwierdzić czy w grafie występuje cykl należy przeprowadzić numerację wsteczną a następnie sprawdzić wszystkie krawędzie.
- Całkowity czas wykonania **testu cykliczności** to $O(m)$, gdzie m to większa z wartości liczby wierzchołków i liczby krawędzi.

Sortowanie topologiczne

30

- Załóżmy, że graf skierowany G jest acykliczny.
- Dla każdego grafu możemy określić las poszukiwania w głąb, określając numerację wsteczną jego wierzchołków.
- Załóżmy, że (n_1, n_2, \dots, n_k) określa listę wierzchołków grafu G w kolejności odwrotnej do numeracji wstecznej. To znaczy: n_1 jest wierzchołkiem opatrzonym numerem n , n_2 wierzchołkiem opatrzonym numerem $n-1$ i ogólnie wierzchołek n_i jest opatrzony numerem $n-i+1$.
- Kolejność wierzchołków na tej liście ma tę własność, że wszystkie krawędzie grafu G biegną od początku do końca, tzn. początek poprzedza koniec.

Sortowanie topologiczne

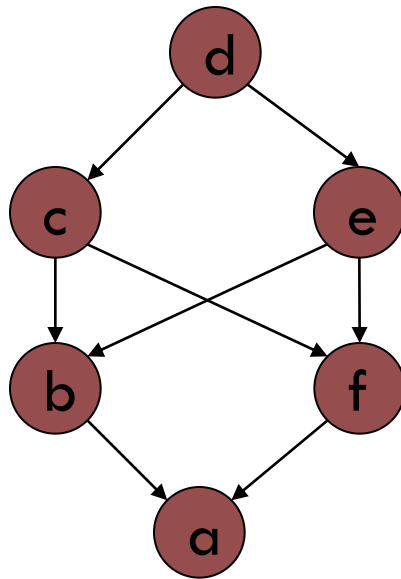
31

- Takie uporządkowanie nazywamy **topologicznym** (ang. *topological order*), a proces znajdowania takiego uporządkowania to **sortowanie topologiczne** (ang. *topological sorting*).
- Jedynie grafy acykliczne posiadają uporządkowanie topologiczne.
- Wykonując poszukiwanie w głąb możemy je określić w czasie $O(m)$.
- Jedną z możliwości: **odkładać kolejno znalezione wierzchołki „na stos”**. Po zakończeniu lista znajdująca się na stosie będzie reprezentować uporządkowanie topologiczne grafu.

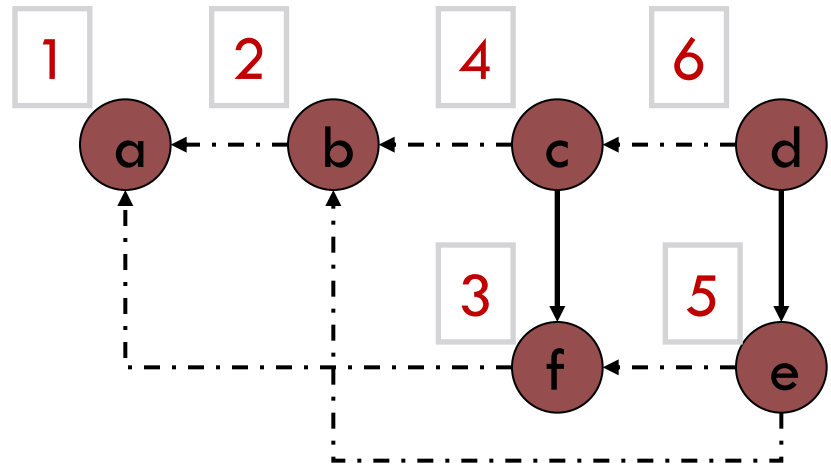
Sortowanie topologiczne

32

Uporządkowanie topologiczne to (d, e, c, f, b, a)



Skierowany graf cykliczny



Las przeszukiwania w głąb

Sortowanie topologiczne

33

- **Uporządkowanie topologiczne** przydaje się **wówczas, gdy istnieją pewne ograniczenia odnośnie kolejności w jakiej mają być wykonywane zadania.**
- **Jeśli krawędź wiodącą od wierzchołka u do wierzchołka v jest rysowana wówczas, gdy zadanie u musi zostać wykonane przed zadaniem v , to uporządkowaniem zapewniającym wykonanie wszystkich zadań jest właśnie uporządkowanie topologiczne.**

Sortowanie topologiczne

34

- Podobny przykład to **graf wywołań** nierekurencyjnego zbioru funkcji, kiedy należy przeanalizować każdą funkcję dopiero po dokonaniu analizy funkcji ją wywołującej.
 - ▣ Jeśli krawędzie wiodą od funkcji wywołujących do wywoływanych, kolejność, w której należy przeprowadzić takie analizy, to odwrócenie porządku topologicznego, czyli **uporządkowanie wsteczne**.
 - ▣ Zapewnia to że każda funkcja zostanie przeanalizowana dopiero po dokonaniu analizy wszystkich innych wywoływanych przez nią funkcji.

Sortowanie topologiczne

35

- **Istnienie cyklu** w grafie reprezentującym priorytety zadań mówi o tym, że nie istnieje takie uporządkowanie, dzięki któremu możliwe byłoby wykonanie wszystkich zadań.
- **Istnienie cyklu** w grafie wywołań pozwala stwierdzić występowanie rekurencji.

Problem osiągalności

36

- **Naturalne pytanie związane z grafem skierowanym jest:**
 - ▣ **które wierzchołki są osiągalne z danego wierzchołka u przy założeniu, że po grafie można się poruszać tylko zgodnie z kierunkiem krawędzi? Taki zbiór wierzchołków określa się mianem **zbioru osiągalności** (ang. *reachable set*) danego wierzchołka u .**
- **Możemy wykorzystać rekurencyjną funkcję poszukiwania w głąb. Całkowity czas wykonania takiego zapytania to $O(mn)$.**

Znajdowanie spójnych składowych

37

- Do znajdowania spójnych składowych możemy użyć algorytmu poszukiwania w głąb.
- Traktujemy graf nieskierowany jako graf skierowany, w którym każda krawędź nieskierowana została zastąpiona dwiema krawędziami skierowanymi wiodącymi w obu kierunkach.
- Do reprezentacji grafu używamy list sąsiedztwa.
- Tworzymy las przeszukiwania w głąb grafu skierowanego. Każde drzewo w tym lesie odpowiada jednej składowej spójności grafu nieskierowanego.
- Czas wykonania algorytmu $O(m)$
 - ▣ przy użyciu struktury drzewiastej czas wykonania wynosi $O(m \log n)$.

Algorytm Dikstry

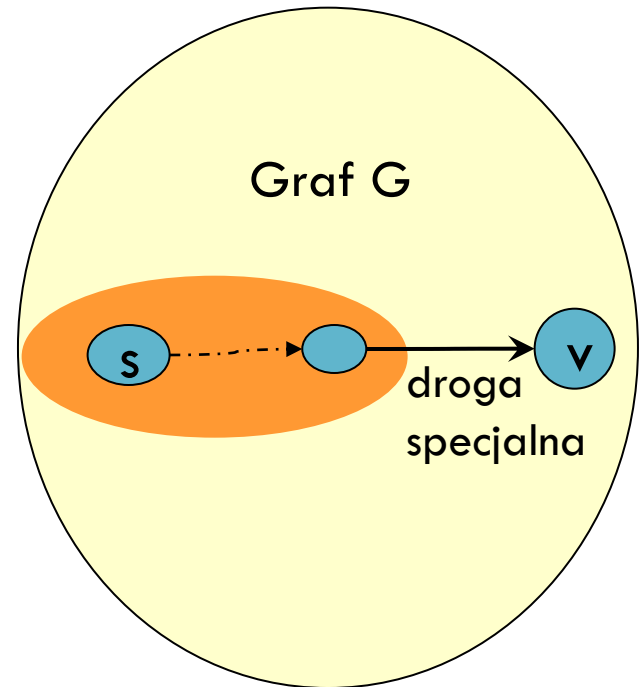
38

- Szukamy najkrótszej drogi pomiędzy dwoma wierzchołkami
 - ▣ Rozpatrujemy graf G (skierowany lub nieskierowany), w którym wszystkie krawędzie zaetykietowano wartościami reprezentującymi ich długości.
 - ▣ **Długość** (ang. distance) danej drogi stanowi wartość sumy etykiet związanych z nią krawędzi. Minimalna odległość z wierzchołka u do wierzchołka v to minimalna długość którejs z dróg od u do v .

Algorytm Dikstry

39

- Traktujemy wierzchołek s jako **wierzchołek źródłowy**. Na etapie pośrednim wykonywania algorytmu w grafie G istnieją tzw. **wierzchołki ustalone** (ang. *settled*), tzn. takie dla których znane są odległości minimalne. W szczególności zbiór takich wierzchołków zawiera również wierzchołek s .
- Dla **nieustalonego wierzchołka** v należy zapamiętać długość najkrótszej **drogi specjalnej** (ang. *special path*) czyli takiej która rozpoczyna się w wierzchołku źródłowym, wiedzie przez ustalone wierzchołki, i na ostatnim etapie przechodzi z obszaru ustalonego do wierzchołka v .



Algorytm Dikstry

40

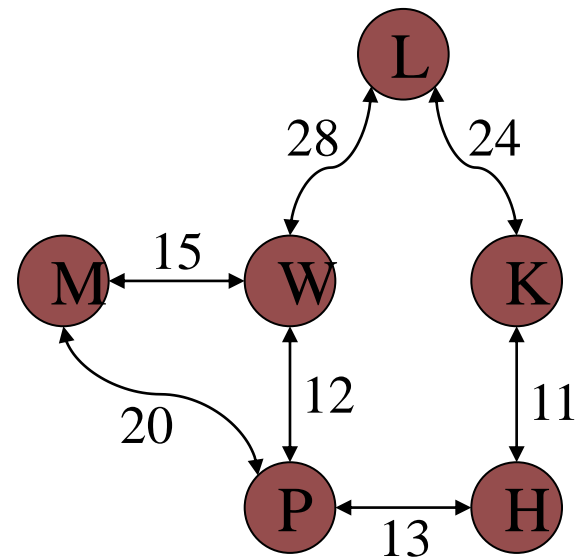
- Dla każdego wierzchołka u zapamiętujemy wartość $\text{dist}(u)$.
- Jeśli u jest wierzchołkiem ustalonym, to $\text{dist}(u)$ jest długością najkrótszej drogi ze źródła do wierzchołka u .
- Jeśli u nie jest wierzchołkiem ustalonym, to $\text{dist}(u)$ jest długością drogi specjalnej ze źródła do u .
- Na czym polega **ustalanie wierzchołków**:
 - znajdujemy wierzchołek v który jest nieustalony ale posiada najmniejszą $\text{dist}(v)$ ze wszystkich wierzchołków nieustalonych
 - przyjmujemy wartość $\text{dist}(v)$ za minimalną odległość z s do v
 - dostosowujemy wartości wszystkich $\text{dist}(u)$ dla innych wierzchołków, które nie są ustalone, wykorzystując fakt, że wierzchołek v jest już ustalony.
 - Czyli porównujemy stare $\text{dist}(u)$ z wartością $\text{dist}(v) + \text{etykieta}(v, u)$ jeżeli taka (v, u) krawędź istnieje.
- Czas wykonania algorytmu jest $O(m \log n)$.

Algorytm Dikstry

41

Etapy wykonania algorytmu

MIASTO	ETAPY ustalania wierzchołków				
	(1)	(2)	(3)	(4)	(5)
H	0*	0*	0*	0*	0*
P	13	13	13*	13*	13*
M	INF	INF	33	33	33*
W	INF	INF	25	25*	25*
L	INF	35	35	35	35
K	11	11*	11*	11*	11*



Algorytmy znajdowania najkrótszych dróg

42

- Jeśli potrzebne jest **poznanie minimalnych odległości między wszystkimi parami wierzchołków w grafie o n wierzchołkach**, które posiadają etykiety o wartościach nieujemnych, można uruchomić algorytm Dijkstry dla każdego z n wierzchołków jako wierzchołka źródłowego.
- Czas wykonania **algorytmu Dijkstry wynosi $O(m \log n)$** , gdzie m oznacza większą wartość z liczby wierzchołków i liczby krawędzi. Znalezenie w ten sposób minimalnych odległości między wszystkimi parami wierzchołków zajmuje czas rzędu **$O(m n \log n)$** .
- Jeśli m jest bliskie swojej maksymalnej wartości $m \approx n^2$ to można skorzystać z implementacji algorytmu Dijkstry który działa w czasie **$O(n^2)$** . Wykonanie go n razy daje czas rzędu **$O(n^3)$** .

Algorytmy znajdowania najkrótszych dróg

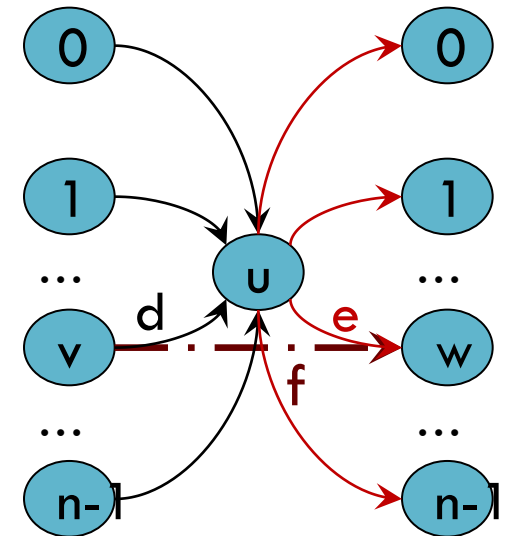
43

- Istnieje inny algorytm znajdowania **minimalnych odległości między wszystkimi parami wierzchołków**, noszący nazwę algorytmu **Floyda-Warshalla**.
- Jego wykonanie zajmuje czas rzędu **$O(n^3)$** . Operuje na macierzach sąsiedztwa a nie listach sąsiedztwa i jest koncepcyjnie prostszy.

Algorytm Floyda-Warshalla

44

- Podstawa algorytmu jest działanie polegające na rozpatrywaniu po kolei **każdego wierzchołka** grafu jako **elementu centralnego** (ang. *pivot*).
- Kiedy wierzchołek **u** jest elementem centralnym, staramy się wykorzystać fakt, że **u** jest wierzchołkiem pośrednim między wszystkimi parami wierzchołków.
- Dla każdej pary wierzchołków, na przykład **v** i **w**, jeśli suma etykiet krawędzi **(v, u)** oraz **(u, w)** (na rysunku **d+e**), jest mniejsza od bieżąco rozpatrywanej etykiety **f** krawędzi wiodącej od **v** do **w**, to wartość **f** jest zastępowana wartością **d+e**.

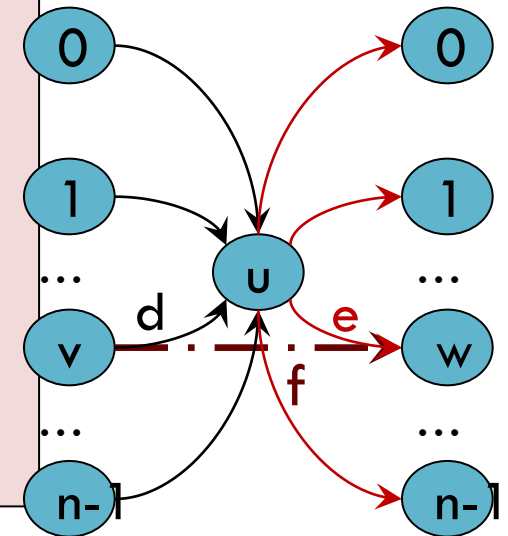


Algorytm Floyda-Warshalla

45

Node u, v, w ;

```
for (v = 0; w < MAX; v++)  
  for (w=0; w < MAX; w++)  
    dist[v][w] = edge[v][w];  
for (u=0; u < MAX; u++)  
  for (v=0; v < MAX; v++)  
    for (w=0; w < MAX; w++)  
      if( dist[v][u]+dist[u][w] < dist[v][w])  
        dist[v][w] = dist [v][u] + dist [u][w];
```

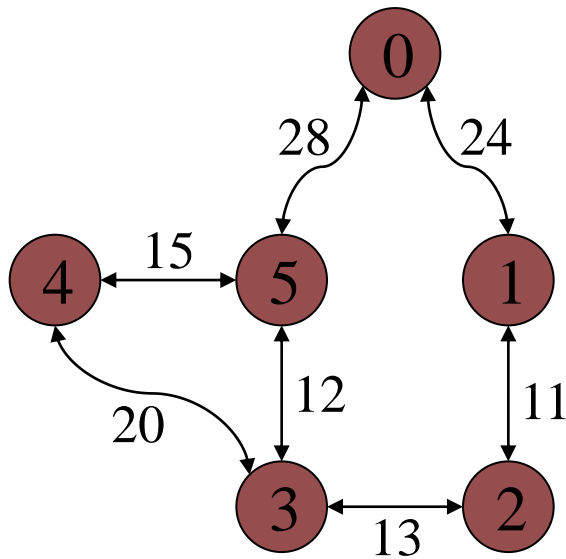


edge[v][w] –etykieta krawędzi, wierzchołki numerowane

Algorytm Floyda-Warshalla

46

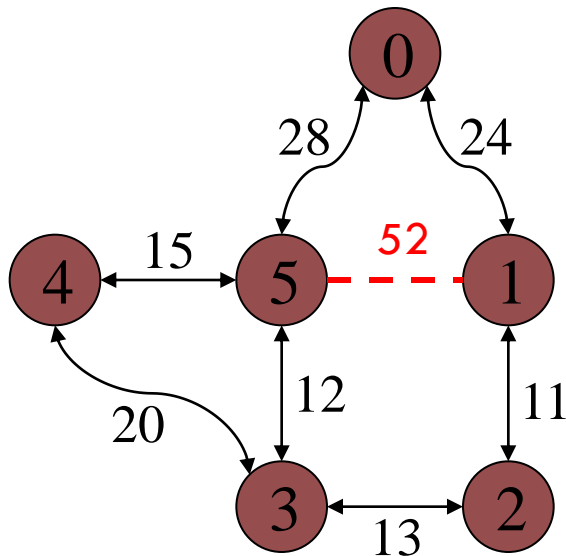
Macierz która odzwierciedla początkową postać macierzy odległości (ang. *dist*)



	0	1	2	3	4	5
0	0	24	INF	INF	INF	28
1	24	0	11	INF	INF	INF
2	INF	11	0	13	INF	INF
3	INF	INF	13	0	20	12
4	INF	INF	INF	20	0	15
5	28	INF	INF	12	15	0

Algorytm Floyda-Warshalla

47

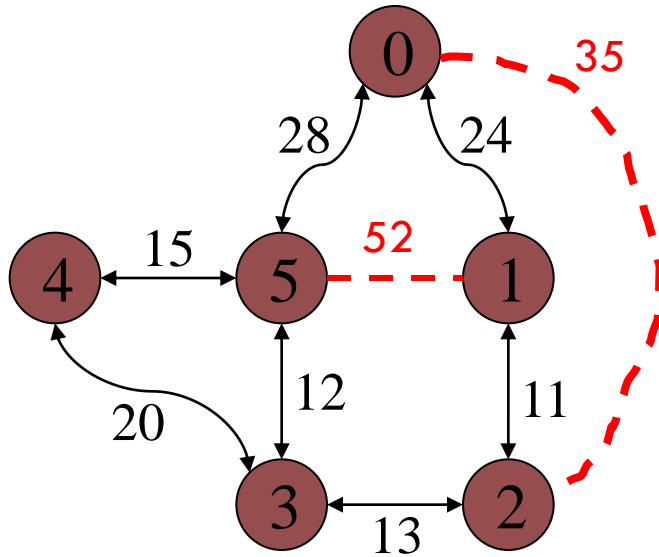


Macierz odległości, po użyciu wierzchołka 0 jako elementu centralnego

	0	1	2	3	4	5
0	0	24	INF	INF	INF	28
1	24	0	11	INF	INF	52
2	INF	11	0	13	INF	INF
3	INF	INF	13	0	20	12
4	INF	INF	INF	20	0	15
5	28	52	INF	12	15	0

Algorytm Floyda-Warshalla

48



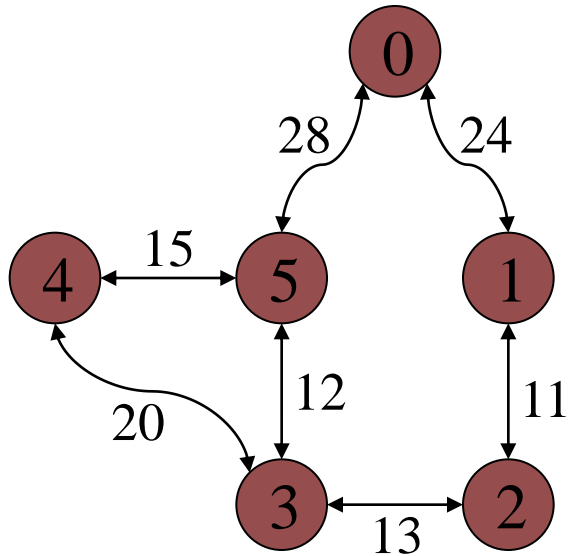
Macierz odległości, po użyciu wierzchołka 1 jako elementu centralnego

	0	1	2	3	4	5
0	0	24	35	INF	INF	28
1	24	0	11	INF	INF	52
2	35	11	0	13	INF	63
3	INF	INF	13	0	20	12
4	INF	INF	INF	20	0	15
5	28	52	63	12	15	0

itd... itd...

Algorytm Floyda-Warshalla

49



Końcowa postać macierzy odległości.

	0	1	2	3	4	5
0	0	24	35	40	43	28
1	24	0	11	24	44	52
2	35	11	0	13	33	25
3	40	24	13	0	20	12
4	43	44	33	20	0	15
5	28	36	25	12	15	0

Podsumowanie

50

PROBLEM	ALGORYTM(Y)	CZAS WYKONANIA
Minimalne drzewo rozpinające	Algorytm Kruskala	$O(m \log n)$
Znajdowanie cykli	Przeszukiwanie w głąb	$O(m)$
Uporządkowanie topologiczne	Przeszukiwanie w głąb	$O(m)$
Osiągalność w przypadku pojedynczego źródła	Przeszukiwanie w głąb	$O(m)$
Spójne składowe	Przeszukiwanie w głąb	$O(m)$
Najkrótsza droga dla pojedynczego źródła	Algorytm Dijskry	$O(m \log n)$
Najkrótsza droga dla wszystkich par	Algorytm Dijskry	$O(m n \log n)$
	Algorytm Floyd	$O(n^3)$

Pytania do egzaminu

51

- 1) **Co to jest grafowy model danych? Omów podstawową terminologię dotyczącą grafów**
- 2) **Co to znaczy implementacja przy pomocy macierzy sąsiedztwa? Zilustruj przykładem.**
- 3) **Co to znaczy implementacja przy pomocy listy sąsiedztwa? Zilustruj przykładem.**
- 4) **Co to jest składowa spójna grafu? Co to jest drzewo rozpinające?**
- 5) **Na czym polega algorytm Kruskala dla znajdowania minimalnego drzewa rozpinającego?**
- 6) **Na czym polega sortowanie topologiczne grafu?**
- 7) **Na czym polega algorytm Dikstry do znajdowania najkrótszej drogi, jaka jest jego złożoność obliczeniowa?**
- 8) **Na czym polega algorytm Floyda do znajdowania najkrótszej drogi, jak jest jego złożoność obliczeniowa?**