Algorytmy z powracaniem

Materiały

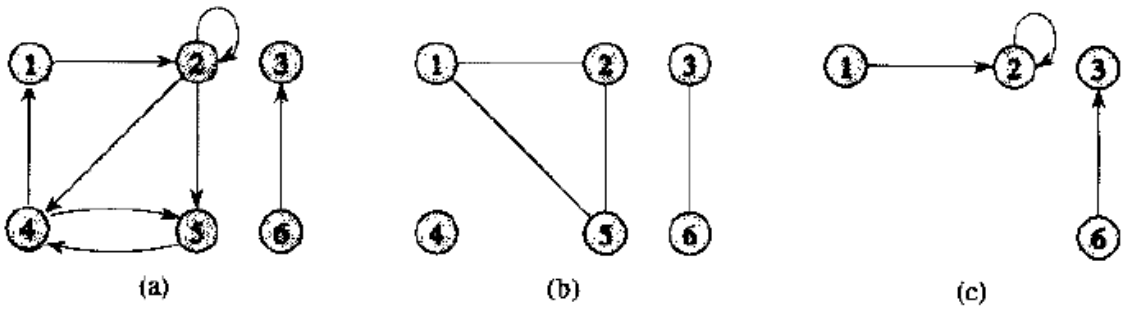
Grafem nazywamy zbiór G = (V, E), gdzie:

- **V** jest zbiorem **wierzchołków** (ang. vertex)
- **E** jest zbiorem **krawędzi** (E można też określić jako podzbiór zbioru nieuporządkowanych par z V)

E (edge) to **krawędź** – połączenie pomiędzy dwoma elementami zbioru V (choć może istnieć też krawędź z wierzchołka do niego samego).
Krawędź jest połączeniem nieskierowanym, tj. jeśli istnieje pomiędzy wierzchołkami *a* oraz *b*, to możemy z wierzchołka *a* przejść do *b*, ale także z wierzchołka *b* do wierzchołka *a*.

Graf skierowany (digraf) ***G*** opisany jest parą (*V, E*), gdzie *V* nazywany jest zbiorem wierzchołków *G*, a *E* zbiorem krawędzi *G*.
Na poniższym rysunku pokazano graf skierowany o zbiorze wierzchołków {1, 2, 3, 4, 5, 6}. Wierzchołki przedstawiono jako kółka, a krawędzie jako strzałki. Na rysunku widać, że możliwe jest istnienie pętli od danego wierzchołka do niego samego.

W grafie niekierowanym *G=(V, E)*, zbiór krawędzi *E* to zbiór nieuporządkowanych par wierzchołków. W grafie niekierowanym nie mogą występować pętle, zapis (u, v) i (v, u) oznacza tę samą krawędź.
Na poniższym rysunku pokazano graf nieskierowany o zbiorze wierzchołków {1, 2, 3, 4, 5, 6}.



Rys. 5.2. Grafy skierowane i nieskierowane. **(a)** Graf skierowany $G = (V, E)$, gdzie $V = \{1, 2, 3, 4, 5, 6\}$ i $E = \{(1, 2), (2, 2), (2, 4), (2, 5), (4, 1), (4, 5), (5, 4), (6, 3)\}$. Krawędź (2, 2) jest pętlą. **(b)** Graf nieskierowany $G = (V, E)$, gdzie $V = \{1, 2, 3, 4, 5, 6\}$ i $E = \{(1, 2), (1, 5), (2, 5), (3, 6)\}$. Wierzchołek 4 jest izolowany. **(c)** Podgraf grafu z części (a) utworzony ze zbioru wierzchołków $\{1, 2, 3, 6\}$

Jeżeli *(u, v)* jest krawędzią grafu skierowanego *G=(V, E)*, to mówimy, że krawędź *(u, v)* jest **wychodząca** z wierzchołka *u* i jest **wchodząca** do wierzchołka *v*. Jeżeli *(u, v)* jest krawędzią grafu nieskierowanego *G=(V, E)*, to mówimy, że *(u, v)* jest **incydentna** z wierzchołkami *u* i *v*. Jeżeli *(u, v)* jest krawędzią grafu *G=(V, E)*, to mówimy, że wierzchołek *v* jest **sąsiedni** do wierzchołka *u*.

Stopniem wierzchołka w grafie niekierowanym nazywamy liczbę incydentnych z nim krawędzi. W grafie skierowanym wyróżniamy **stopień wyjściowy** wierzchołka (liczba krawędzi wychodzących z niego) oraz **stopień wejściowy** wierzchołka (liczba krawędzi do niego wchodzących). **Stopniem** wierzchołka w grafie skierowanym nazywamy liczbę będącą sumą jego stopni: wejściowego i wyjściowego.

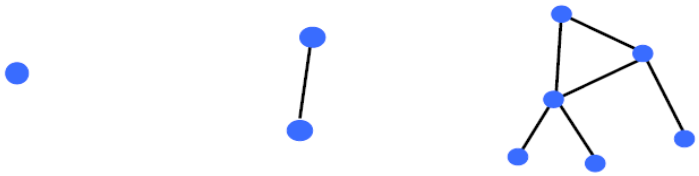
Ścieżka długości *k* z wierzchołka *u* do wierzchołka *u'* w grafie *G=(V, E)* jest ciągiem wierzchołków $\langle v_0, v_1, \ldots v_k \rangle$, takich, że $u=v_0$, $u'=v_k$ i $(v_{i-1},v_i) \in E$ dla $i=1,2,\ldots,k$. Długość ścieżki to liczba krawędzi ścieżki.

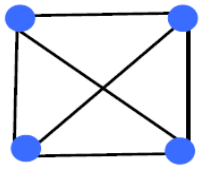
Drogą z wierzchołka v_0 do wierzchołka v_k w grafie *G=(V, E)* nazywamy uporządkowaną sekwencję krawędzi/łuków prowadzących z v_0 do v_k: $(v_0,v_1), (v_1,v_2), \ldots, (v_{k-1},v_k)$.

UWAGA: Formalnie, droga to uporządkowany ciąg krawędzi, ścieżka – uporządkowany ciąg wierzchołków. Potocznie często się o tym zapomina i stosuje wymiennie (ponieważ zazwyczaj

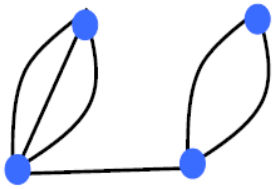
chodzi nam o to samo – znalezienie połączenia między dwoma wierzchołkami), jednak kiedy piszemy formalne definicje np. w sprawozdaniu, lepiej trzymać się teorii i definicji.

Grafy:

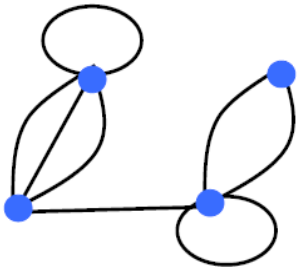




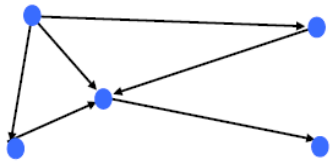
graf pełny – każdy wierzchołek jest połączony krawędzią z każdym wierzchołkiem



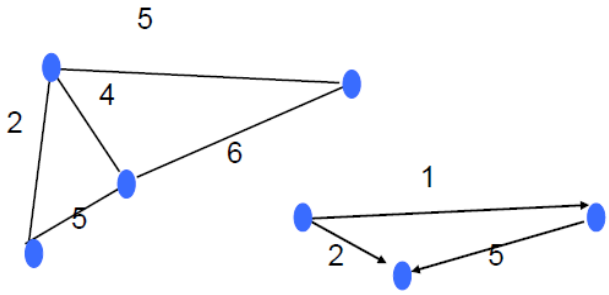
Multigraf – dwa wierzchołki mogą być połączone kilkoma krawędziami



Pseudograf – pojawiają się pętle czyli krawędzie wychodzące z i wchodzące do tego samego wierzchołka



Graf skierowany – krawędzie, zwane też łukami, spełniają warunek (u,v)≠(v,u)



Graf ważony – gdy krawędziom zostają przyporządkowane liczby – wagi

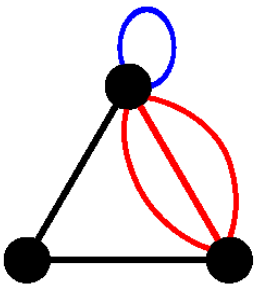
Klasy grafów:

- **Graf prosty (multigraf)**

  Graf prosty – graf bez pętli własnych oraz bez krawędzi wielokrotnych.

  Multigraf – graf w którym MOGĄ występować pętle własne i krawędzie wielokrotne.

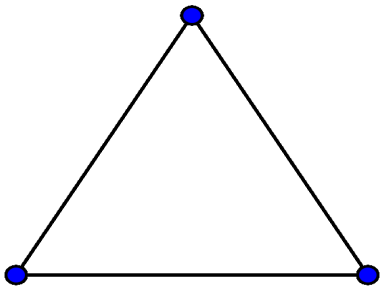
  - pętla własna: wierzchołek posiada łuk i/lub krawędź prowadzącą bezpośrednio do samego siebie
  - krawędź wielokrotna: na przykład z wierzchołka a do wierzchołka b prowadzą dwie krawędzie jeżeli w zbiorze E dwa razy występuje para (a,b).



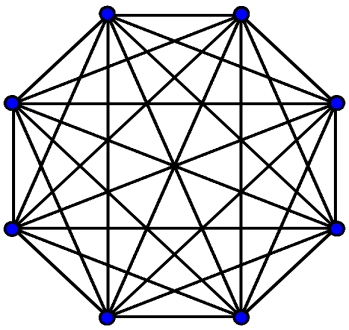
- **graf pełny (graf regularny)**

  W grafie pełnym każdy wierzchołek jest połączony z każdym innym – opisuje się je często jako $\mathbf{K_n}$ (n – liczba wierzchołków)

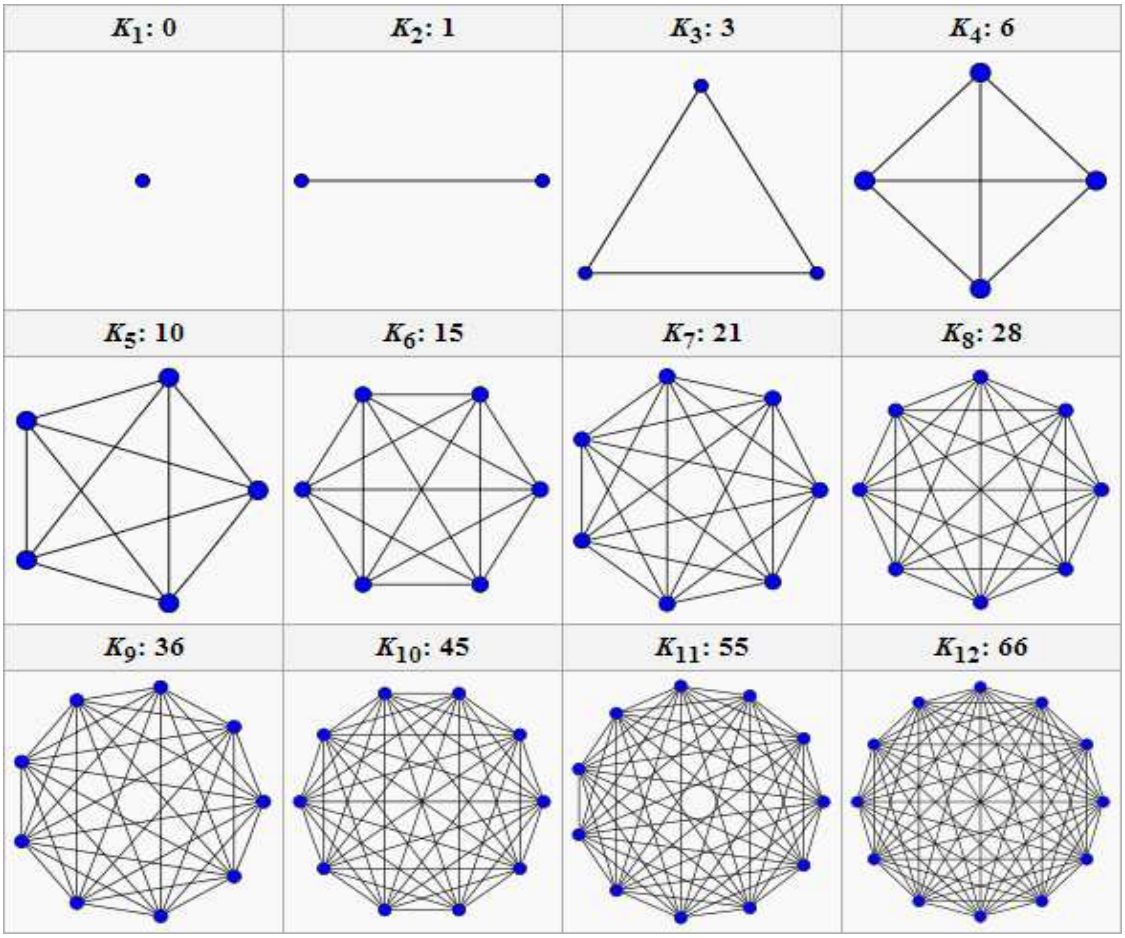
  Graf regularny stopnia k – każdy wierzchołek ma k krawędzi.



graf pełny K_3



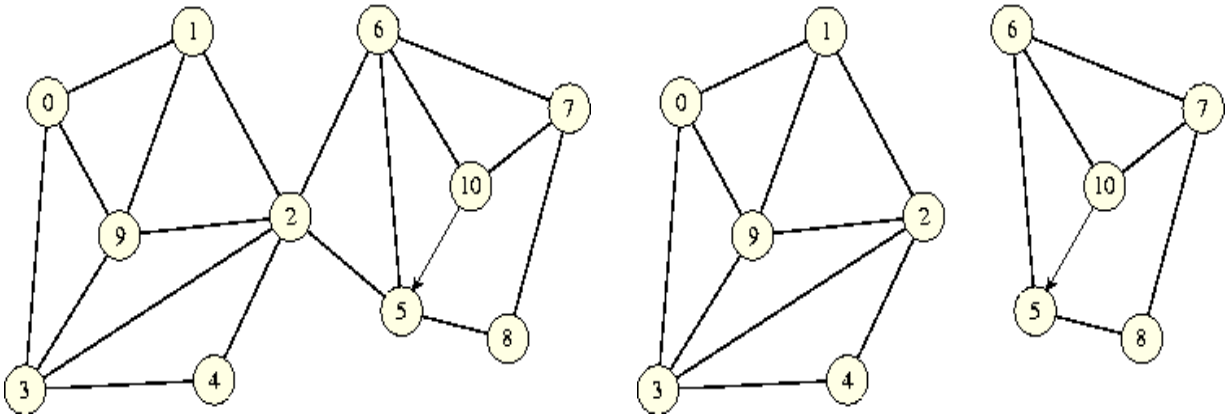
graf pełny K_8



W grafie skierowanym ścieżka <v_0, v_1, … v_k> tworzy **cykl** jeśli $v_0 = v_k$ oraz zawiera co najmniej jedną krawędź. **Cykl jest prosty**, jeśli v_0, v_1, … v_k są dodatkowo różne. Na powyższym rysunku 5.2a ścieżka <1, 2, 4, 1> tworzy cykl. W grafie niekierowanym ścieżka <v_0, v_1, … v_k> tworzy **cykl** jeśli $v_0 = v_k$ i v_0, v_1, … v_k są różne i $k \geq 2$. Mówimy, że cykl jest **prosty**, jeśli, dodatkowo,

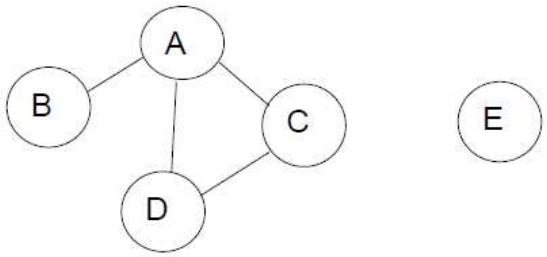
v_1, … v_k są różne. Na powyższym rysunku 5.2b ścieżka <1, 2, 5, 1> tworzy cykl. Graf nie zawierający cykli nosi nazwę **acyklicznego**.

Graf niekierowany jest **spójny**, jeśli każda para wierzchołków jest połączona ścieżką.



Reprezentacje maszynowe grafów (reprezentacja grafów w programach):

- **<u>Macierz sąsiedztwa</u>**

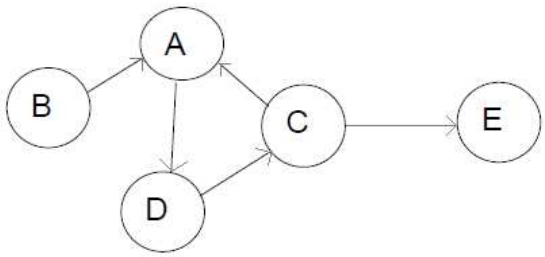


	A	B	C	D	E
A	0	1	1	1	0
B	1	0	0	0	0
C	1	0	0	1	0
D	1	0	1	0	0
E	0	0	0	0	0

<u>W programach jest to tablica dwuwymiarowa:</u>

- indeksy wierszy i kolumn reprezentują numery wierzchołków
- 1 oznacza, że krawędź pomiędzy dwoma wierzchołkami
- 0 brak krawędzi

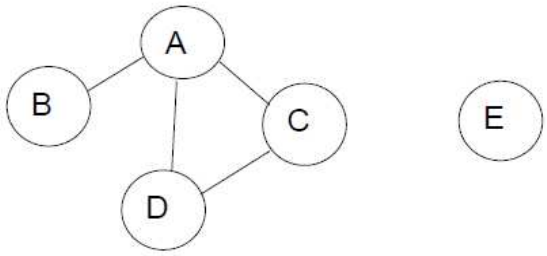
Przykład dla grafu skierowanego:



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

Zasady jak dla macierzy sąsiedztwa dla grafu z krawędziami, różnica jest jednak taka, że tablica NIE JEST symetryczna względem swojej przekątnej

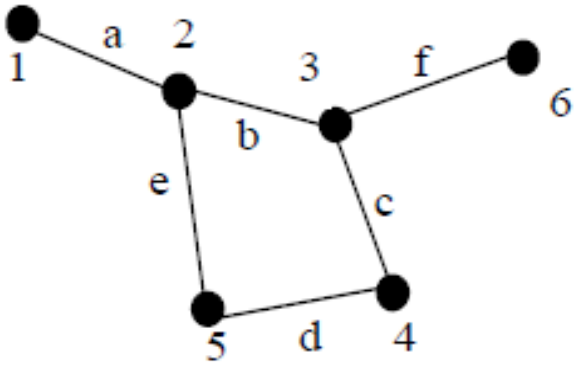
- **Lista sąsiedztwa**

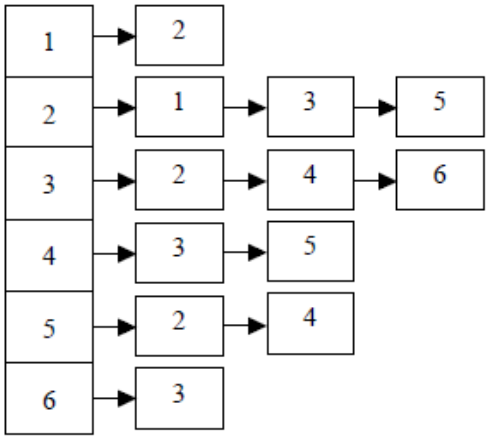


A	B C D
B	A
C	A D
D	A C
E	

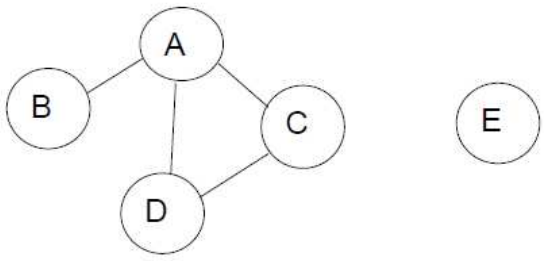
Można ją reprezentować jako tablicę struktur, w której każdy element tablicy to struktura zawierająca:

- numer wierzchołka
- wskaźnik na listę wierzchołków sąsiadujących z danym wierzchołkiem





- **Macierz incydencji**

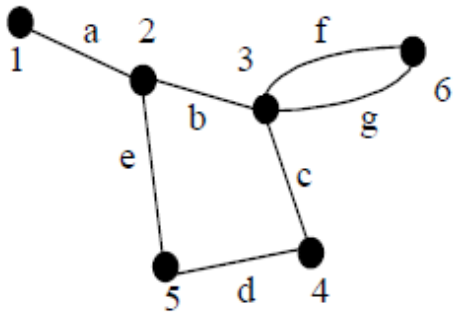


	AB	AC	AD	CD
A	1	1	1	0
B	1	0	0	0
C	0	1	0	1
D	0	0	1	1
E	0	0	0	0

W programach przedstawiana jako tablica dwuwymiarowa, gdzie:

- indeksy wierszy to numery wierzchołków

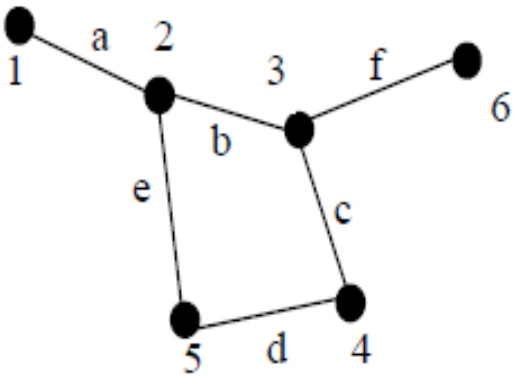
- indeksy kolumn to numery łuków

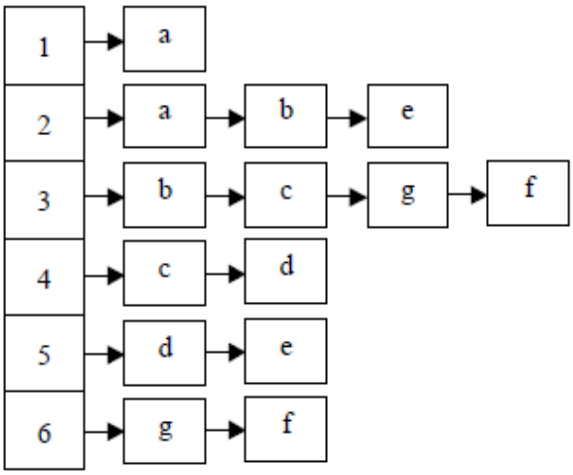


	a	*b*	*c*	*d*	*e*	*f*	*g*
1	1	0	0	0	0	0	0
2	1	1	0	0	1	0	0
3	0	1	1	0	0	1	1
4	0	0	1	1	0	0	0
5	0	0	0	1	1	0	0
6	0	0	0	0	0	1	1

- **Lista incydencji**

  Jak w przypadku listy sąsiedztwa, z tym, że z każdym wierzchołkiem związana jest lista KRAWĘDZI do niego incydentnych (a nie jak poprzednio: sąsiadujących z nim wierzchołków).





Cyklem Eulera w grafie *G=(V, E)* nazywamy każdy cykl, który przechodzi przez każdą krawędź grafu *G* dokładnie raz (chociaż ten sam wierzchołek może być odwiedzany więcej niż raz). Spójny graf skierowany ma cykl Eulera wtedy i tylko wtedy, gdy stopień wejściowy każdego wierzchołka *v* jest równy jego stopniowi wyjściowemu. W skierowanej wersji grafu nieskierowanego każdej nieskierowanej krawędzi (*u*, *v*) odpowiadają dwie skierowane krawędzie (*u*, *v*) i (*v*, *u*), więc wersja skierowana dowolnego spójnego grafu nieskierowanego ma cykl Eulera.

Przypomnienie: droga to uporządkowana sekwencja krawędzi w grafie od wierzchołka startowego do końcowego.

Cykl jest drogą, w której startowy i końcowy wierzchołek to jeden i ten sam: innymi słowy do wierzchołka z którego wyszliśmy musimy pod koniec wrócić.

Cykl Eulera w grafie to taki cykl, który zawiera każdą krawędź grafu dokładnie raz.

Warunki istnienia:

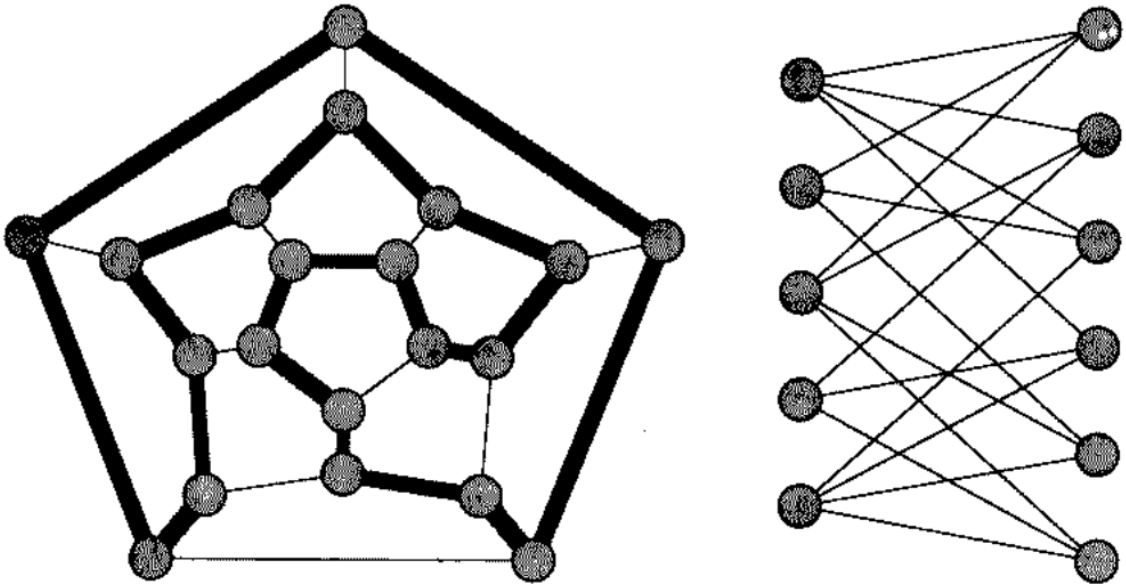
- spójność grafu
- dla grafu skierowanego liczba łuków wchodzących do wierzchołka musi być równa liczbie łuków z niego wychodzących
- dla grafu nieskierowanego każdy wierzchołek ma mieć parzystą liczbę krawędzi incydentnych

Cykl Hamiltona w grafie nieskierowanym *G=(V, E)* to cykl prosty zawierający wszystkie wierzchołki zbioru *V*. Graf, który ma cykl Hamiltona nosi nazwę **hamiltonowskiego**; w przeciwnym razie jest **niehamiltonowski**.

Ścieżka między dwoma wierzchołkami s i t to lista kolejnych wierzchołków, które należy przejść aby z s trafić do t.

Cykl Hamiltona to taki cykl, który zawiera każdy z wierzchołków grafu dokładnie raz.

Uwaga: ponieważ jest to cykl, pierwszy i ostatni wierzchołek to ten sam wierzchołek; ergo: jako jedyny może (i musi) być 'odwiedzony' dwukrotnie.



Rys. 36.1. (a) Graf reprezentujący wierzchołki, krawędzie i ściany dwunastościanu z zaznaczonymi szarym kolorem krawędziami cyklu Hamiltona. (b) Graf dwudzielny o nieparzystej liczbie wierzchołków. Żaden taki graf nie jest hamiltonowski

Cykl Eulera

Definicja:

Pamiętacie z dzieciństwa zabawę w rysowanie koperty bez odrywania długopisu? To jest właśnie problem znajdowania łańcucha Eulera w grafie. Gdyby dodać warunek, że musimy zacząć i skończyć rysować w tym samym miejscu (tak się nie da) to byłby to problem znajdowania cyklu Eulera.
Cykl Eulera, to taki cykl w grafie, która zawiera każdą krawędź grafu, przy czym każdą dokładnie raz. Natomiast łańcuch Eulera to taka droga (czyli nie musi się kończyć tam gdzie zaczyna), która zawiera każdą krawędź grafu dokładnie raz.
Dotyczy to zarówno grafów nieskierowanych, jak i skierowanych.

Warunek istnienia cyklu:

Żeby znaleźć cykl/łańcuch Eulera w grafie trzeba najpierw sprawdzić czy istnieje. Przede wszystkim (i nie można o tym zapomnieć!) należy sprawdzić czy graf jest spójny (ale nie liczymy samotnych wierzchołków). Dla grafu skierowanego należy sprawdzić spójność jego nieskierowanego odpowiednika (tzn. chwilowo pozbawiamy krawędzi kierunku). Oczywiście gdy graf nie jest spójny to nie istnieje cykl Eulera (no bo jak obejść wszystkie krawędzie jeżeli nie można do którejś dojść?).
Teraz wystarczy zliczyć ile jest krawędzi wychodzących/wchodzących z/do każdego wierzchołka i sprawdzić jeden prosty warunek. W przypadku grafów skierowanych cykl Eulera istnieje wtedy i tylko wtedy gdy do każdego wierzchołka wchodzi tyle samo krawędzi co wychodzi (uwaga: pętla jednocześnie

wchodzi i wychodzi). Natomiast w przypadku grafów nieskierowanych z każdego wierzchołka musi wychodzić parzysta liczba krawędzi (jedna połowa okaże się wchodzącymi, a druga połowa wychodzącymi).
Jeżeli chcemy znaleźć tylko łańcuch Eulera to możemy sobie pozwolić na drobne odstępstwo od powyższego warunku w dokładnie dwóch wierzchołkach. Czyli w przypadku grafów nieskierowanych dwa wierzchołki mogą mieć nieparzystą liczbę sąsiadów; w przypadku grafów skierowanych z dwóch wierzchołków liczba krawędzi wchodzących i wychodzących może się różnić o jeden.

Opis algorytmu:

Algorytm Fleury'ego

G – graf spójny
ES – ciąg krawędzi drogi lub cyklu Eulera
VS – ciąg wierzchołków drogi lub cyklu Eulera

Krok 1. Wybierz dowolny wierzchołek *v* nieparzystego stopnia, jeśli taki istnieje. W przeciwnym wypadku wybierz dowolny wierzchołek *v*. Niech *VS=v* i niech *ES=0* (ciąg pusty).

Krok 2. Jeśli z wierzchołka *v* nie wychodzi żadna krawędź, zatrzymaj się.

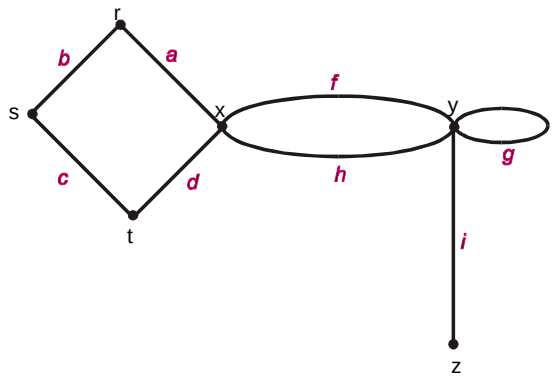
Krok 3. Jeśli pozostała dokładnie jedna krawędź wychodząca z wierzchołka *v*, powiedzmy krawędź *e* z wierzchołka *v* do *w*, to usuń *e* z *E(G)* oraz *v* z *V(G)* i przejdź do **Kroku 5**.

Krok 4. Jeśli została więcej niż jedna krawędź wychodząca z wierzchołka *v*, wybierz taką krawędź, powiedzmy *e* z *v* do *w*, po usunięciu której graf pozostanie spójny; następnie usuń *e* z *E(G)*.

Krok 5. Dołącz *w* na końcu ciągu *VS*, dołącz *e* na końcu ciągu *ES*, zastąp *v* wierzchołkiem *w* i przejdź do **Kroku 2.**

Przykład działania algorytmu Fleury'ego.

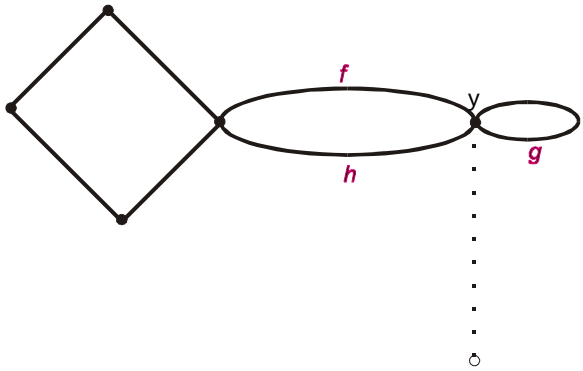
Dany jest graf spójny.



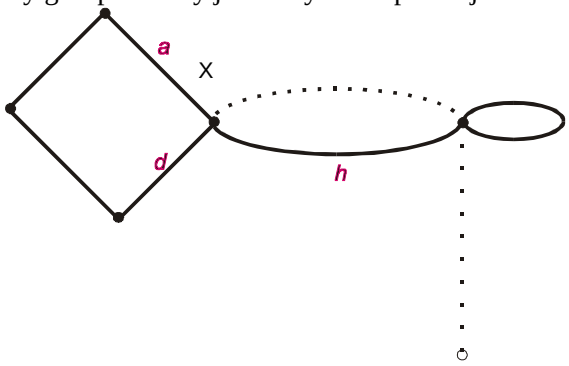
Graf ten nie ma cyklu Eulera, ale ma drogę Eulera łacząca wierzchołki *z* i *y* stopni nieparzystych.

Wybieramy na poczatek wierzchołek *z*, tzn. w kroku 1 kładziemy *v=z*. Zatem *VS=z* i *ES=0*. Jedyną krawędzią wychodzącą z wierzchołka *z* jest *i*. Idziemy zatem do kroku 3, wybieramy *e=i* i *w=y*, usuwamy

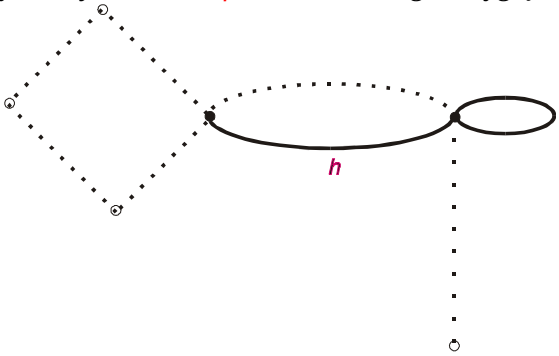
i z *E(G)*, usuwamy *z* z *V(G)* oraz przechodzimy do kroku 5. Wtedy *VS=zy* i *ES=i*. Nowy graf, po usunięciu *i* oraz *z* wygląda następująco:



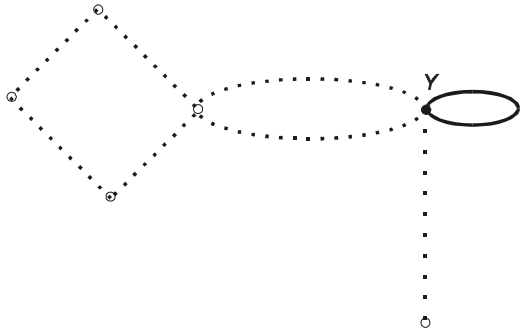
Przyjmujemy *v=y* i powracamy do kroku 2. Z wierzchołka *v* wychodzą trzy krawędzie, więc przechodzimy do kroku 4.
Teraz jako *e* możemy wybrać *f*, *g* lub *h*. Wybierzemy w kroku 4 krawędź *e=f*. W kroku 5 otrzymamy *VS=zyx*, *ES=if* oraz *v=x* i nowy graf pokazany jest na rysunku poniżej

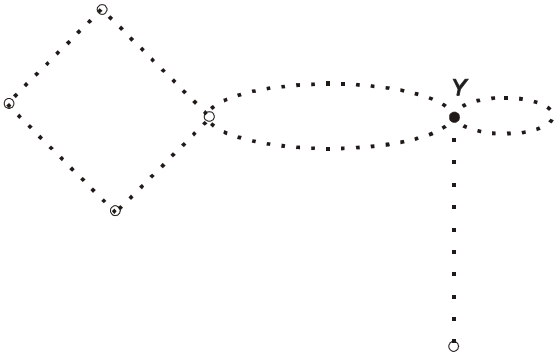


Powracamy do kroku 2. Znów z wierzchołka *v* wychodzą trzy krawędzie, więc przechodzimy do kroku 4. Teraz krawędzią *e* może być *a* lub *d*. Nie może natomiast być *h*, gdyż po usunięciu tej krawędzi graf przestałby być spójny. Wybierzmy *e=a*, w kroku 5 otrzymamy *VS=zyxr*, *ES=ifa* oraz *v=r*.
Nastepne trzy ruchy są wymuszone. Każdy prowadzi do kroku 3 i usunięcia krawędzi wraz z wierzchołkiem. Otrzymujemy *VS=zyxrstx*, *ES=ifabcd*, *v=x* oraz graf wygląda jak na rysunku poniżej.



Ostatnie dwa ruchy sa również wymuszone i prowadzą do grafów pokazanych na dwóch następnych rysunkach. Ostatecznie ciągami wierzchołków i rawędzi są *VS=zyxrstxyy*, *ES=ifabcdhg*.





Złożoność czasowa:

Może powiecie, że się powtarzam i jestem nudny, ale muszę to napisać. Algorytm przegląda każdą krawędź pewną stałą (niewielką) liczbę razy. Zatem działa w czasie O(E) (o ile została wybrana sensowna reprezentacja grafu).

Cykl Hamiltona

Trochę o:

Cykl Hamiltona, to taki cykl w grafie, który zawiera wszystkie wierzchołki grafu, przy czym każdy dokładnie raz.
Problem znajdowania cyklu Hamiltona jest jednak znacznie trudniejszy niż w znajdowania cyklu Eulera. Nie istnieje algorytm działający w sensownym czasie, który potrafi znaleźć cykl Hamiltona w dowolnym grafie.

Opis algorytmu:

Żeby znaleźć cykl Hamiltona w grafie należy zapuścić rekurencję z nawrotami do znajdowania wszystkich cykli (bez powtarzających się wierzchołków), a po znalezieniu jakiegoś cyklu sprawdzamy czy zawiera wszystkie wierzchołki. Niektórym to już mówi wszystko, a tym którym nie mówi to tłumaczę dalej.
Zaczynamy szukać drogę w jakimkolwiek wierzchołku, nazwijmy go s. Załóżmy, że w pewnym momencie wykonywania algorytmu doszliśmy do wierzchołka v. Mamy zapamiętane wierzchołki "użyte" do przejścia drogą z s do v, z oboma tymi wierzchołkami włącznie (uwaga: możliwe, że w v znajdziemy się kilkakrotnie i za każdym razem dojdziemy tam inną drogą z s). Dalej przechodzimy z v rekurencyjnie do jego następników (albo sąsiadów w przypadku grafów nieskierowanych), ale tylko tych nieużytych (zaznaczając je jako już użyte). Gdy obejrzymy wszystkie następniki v i nic nam to nie da, to musimy wrócić (rekurencyjnie) oraz !odznaczyć! wierzchołek v, żeby w przyszłości można było ponownie przez niego przejść. Jeżeli w pewnym momencie z jakiegoś wierzchołka będzie można przejść bezpośrednio do s to znaleźliśmy pewien cykl i musimy sprawdzić czy odwiedziliśmy wszystkie wierzchołki; jeśli tak to cykl jest hamiltonowski i możemy go wypisać, a jeśli nie to z powrotem i szukamy dalej.
Żeby móc sprawdzić czy znaleziony cykl jest hamiltonowskim wystarczy jedynie zliczać odwiedzone wierzchołki. Należy przy każdym zagłębieniu rekurencji zwiększyć jakiś licznik o 1, a przy powrocie zmniejszyć go o 1. Jeśli zrobimy cykl od s do s to jest on cyklem Hamiltona gdy licznik ma wartość równą liczbie wierzchołków w grafie (bo wtedy odwiedzone zostały wszystkie wierzchołki).
No to (pseudo)kod. Funkcja HamiltonCycle zwróci True jeżeli cykl Hamiltona zostanie znaleziony oraz w tablicy H znajdą się kolejne wierzchołki tego cyklu; w przeciwnym wypadku funkcja zwróci False.

Złożoność czasowa:

A tutaj niespodzianka. Algorytm by najmniej nie działa w czasie O(E).
Problem znajdowania cyklu Hamiltona w grafie jest NP-zupełny (w dodatku silnie NP-zupełny), co oznacza tyle, że nie istnieje algorytm rozwiązujący ten problem działający w czasie wielomianowym. I rzeczywiście - algorytm, który podałem działa pesymistycznie w czasie O(V!), co zalicza się do czasów wykładniczych (możliwe, że dla przeciętnych grafów działa to lepiej). Skutek tego jest straszny: dla grafów dwudziestoparowierzchołkowych (no, może trochę więcej - nie wiem) algorytm zacznie wymiękać czasowo.

```
{
  N - liczba wierzcholkow
  D[1..N] - tablica logiczna odwiedzin wierzcholkow
  H[1..N] - tablica kolejnych wierzcholkow cyklu Hamiltona
  }

  function PathRec(v: Integer): Boolean;
  {fun. rekurencyjnego znajdowania cykli od s do s}
  begin
    D[v]:=True;    {zaznczanie wierzcholka jako uzyty}
    Inc(used);     {zwiekszenie liczby uzytych wierzcholkow}

    PathRec:=True;
    for i:=Successor(v) do     {petla po nastepnikach wierzch. v}
    begin
      if i=s then     {jesli zrobilismy cykl ...}
        if used=N then    {jesli do tego wszystkie wierzcholki uzyte}
        begin
          H[1]:=v;
          Exit;                          {wtedy wychodzi z wartoscia True}
        end;
      if not D[i] then    {jesli wierzch. i jest nieuzyty ...}
        if PathRec(i) then
        {przechodzimy rek. do wierzcholka i jesli znalezlismy tam cykl H. ...}
        begin
          H[No]:=v;          {wstawiamy kolejny wierzcholek do tablicy H}
          Inc(No);
          Exit;          {wychodzi z wartoscia True}
        end;
    end;

    PathRec:=False;    {nie tedy droga, trza zwrocic False}
    D[v]:=False;  {zaznczanie wierzcholka jako nieuzyty}
    Dec(used);     {zmniejszenie liczby uzytych wierzcholkow}
  end;

  function HamiltonCycle: Boolean;
  begin
    for i:=1 to N do
      D[i]:=False;     {oznaczamy wszystkie wierzcholki jako nieuzyte}

    used:=0;     {zmienna potrzebna do liczenia przebytych wierzcholkow}
    No:=2;     {zmienna potrzebna do wpisywania wierzcholkow cyklu do tablicy H}

    s:=1;    {wierzcholek startowy - calkowicie dowolny z przedzialu 1..N}
    HamiltonCycle:=PathRec(s);
  end;
```

Literatura:

Thomas H. Cormen, Charles E. Leiserson, Ronald L.Rivest
Wprowadzenie do algorytmów
Wydanie czwarte, Wydawnictwo Naukowo-Techniczne, Warszawa, 2001

N. Wirth
Algorytmy + struktury danych = programy
Wydawnictwo Naukowo-Techniczne, Warszawa, 2004