Spis rzeczy

1 Definicje

Cięciwa - krawędź łącząca niekolejne wierzchołki na cyklu

Cykl indukowany - cykl bez cięciw

1.1 A-B ścieżka

A, B - zbiory wierzchołków

A-B ścieżka jest to ścieżka, której jedynymi wierzchołkami wspólnymi z A i B są odpowiednio pierwszy i ostatni wierzchołek ścieżki.

1.2 Separowanie zbiorów

$X \subseteq V \cup E$

A, B - zbiory wierzchołków grafu G

X jest *separuje A i B w G* jeżeli każda A-B ścieżka w G zawiera element z X.

1.3 Separator

X jest *separatorem* w G jeżeli separuje dwa wierzchołki z G-X.

1.4 Wierzchołek separujący

Wierzchołek v jest *wierzchołkiem separującym*, jeżeli v separuje dwa inne wierzchołki .

1.5 Most

Mostem nazywamy krawędź, która separuje swoje końce.

1.6 Spójność

Graf jest *spójny*, gdy każda para wierzchołków jest połączona ścieżką.

1.7 k-spójność

k - liczba naturalna

G - graf o liczbie wierzchołków $>$ k

G jest *k-spójny* (wierzchołkowo), jeżeli dla każdego zbioru wierzchołków X t.ż. |X| $<$ k G-X jest spójny.

1.8 Domknięcie przechodnie

Domknięciem przechodnim grafu G=(V,E) nazywamy graf o tym samym zbiorze wierzhołków w którym istnieje krawędź między u i v <=> istnieje ścieżka między u i v w G.

1.9 Monoid

System (S, $\oplus$, 0) nazywamy *monoidem*, gdy:

1. $\underset{a,b \in S}{\forall} \; a \oplus b \in S$
2. $\oplus$jest łączne
3. 0 jest elementem neutralnym $\oplus$

1.10 Idempotencja

Operację $\oplus$w systemie S nazywamy *idempotentną*, gdy

$\underset{a \in S}{\forall} \; a \oplus a = a$

1.11 Półpierścień domknięty

System (S, $\oplus$,$\odot$, 0, 1) nazywamy *półpierścieniem domkniętym*, gdy:

1. (S, $\oplus$, 0) jest monoidem
2. (S, $\odot$, 1) jest monoidem
3. 0 jest elementem zerowym $\odot$
4. $\oplus$ jest przemienne
5. $\oplus$ jest idempotentne
6. $\odot$ jest rozdzielne względem $\oplus$
7. Nieskończona suma ($\oplus$) przeliczalnej liczby elementów z S jest jednoznacznie określona i należy do S.
8. £ączność, przemienność i idempotentność $\oplus$ odnoszą się również do sum nieskończonych.
9. $\odot$ jest rozdzielne względem sum nieskończonych.

1.12 Szereg nieskończony

S - półpierścień domknięty

Dla a∈S definiujemy $a^* = \sum_{i=0}^{\infty} a^i$, gdzie $a^i = a \odot a^{i-1}$, $a^0 = 1$

1.13 Marszruta

Marszrutą w G nazywamy naprzemienny ciąg wierzchołków i krawędzi, zaczynający się i kończący w wierzchołku.

Uwaga! Wierzchołki i krawędzie *mogą* się powtarzać (w przeciwieństwie do ścieżki).

1.14 Funkcja *low*

G - graf spójny, niezorientowany

T - drzewo DFS na G

Wierzchołki G są ponumerowane zgodnie z kolejnością odwiedzania DFS-em i utożsamione z tymi numerami.

low[v]=MIN(v, najmniejszy numer wierzchołka do którego prowadzi krawędź powrotna od któregoś z potomków v)

Przy czym low[v] można określić jako minimum z v, wartości low dla następników v i numerów wierzchołków do których prowadzi krawędź powrotna z v

2 Przetrzenie cykli i cięć

2.1 Definicje

Przestrzenią krawędziową $\varepsilon(G)$ grafu G=(V,E) nazywamy przestrzeń liniową nad ciałem Z_2 wszystkich wektorów zero-jedynkowych (funkcji f : E -> Z_2)

Iloczyn dwóch wektorów a i b z $\varepsilon(G)$ definiujemy jako:

a = [a_1, ..., a_m]

b= [b_1, ..., b_m]

<a,b>= $\sum_{i=1}^{m} a_i \cdot b_i$, gdzie + (w Z_2) odpowiada XOR, * odpowiada AND, więc

<a,b>=0 <=> oba zbiory krawędzi mają parzystą liczbę wspólnych krawędzi.

Podprzestrzenią prostopadłą $F^{\perp}$ do podprzestrzeni $F \subseteq \varepsilon(G)$ nazywamy zbiór wszystkich wektorów a należących do $\varepsilon(G)$, takich że dla wszystkich b z F $<a,b> = 0$.

Przestrzenią cykli C=C(G) nazywamy podprzestrzeń G zawierającą wszystkie cykle z G.

2.1.1 Cięcie

G=(V,E)

U,W - podział V[1]

E(U,W) - zbiór wszystkich krawędzi łączących wierzchołki z U z wierzchołkami z W

E(U,W) nazywamy *cięciem*.

E(v)=E({v},V-{v})

Zbiór wszystkich cięć w G wraz ze zbiorem pustym oznaczamy przez C^* i nazywamy przestrzenią cięć.

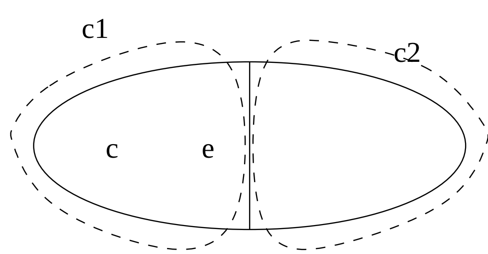
2.2 Twierdzenia

2.2.1 Spostrzeżenie 1

Teza:

Cykle indukowane (bez cięciw) w G rozpinają C(G).

Dowód:



e - cięciwa

c1, c2 - cykle indukowane

Cykl c jest równy c1 + c2 (+ w C(G) oznacza różnicę symetryczną)

[1] $U \cup W = V$ i $U \cap W = \emptyset$

2.2.2 Spostrzeżenie 2

G=(V,E)

Zbiór F$\subseteq E$ należy do C(G) <=> stopień każdego wierzchołka w G'=(V,F) jest parzysty.

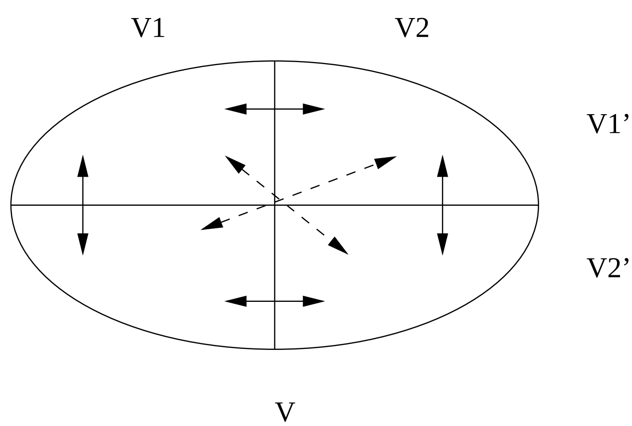
2.2.3 Spostrzeżenie 3

Cięcia w G plus zbiór pusty (czyli C*) tworzą podprzestrzeń przestrzeni krawędzi G.

Dowód:

Niech D, D' należą do C*

1. D+D=$\emptyset$
2. D+$\emptyset$=D
3. D=E(V_1,V_2), D'=E(V_1',V_2')



D+D' zawiera krawędzie, które *nie* należą jednocześnie do obu skojarzeń (linie ciągłe).

2.2.4 Spostrzeżenie 4

Cięcia postaci E(v) generują całą podprzestrzeń cięć.

Dowód:

Jeżeli weźmiemy sumę (czyli różnicę symetryczną) wszystkich cięć postaci E(v_i) dla v_i składających się na zbiór V_1 cięcia E(V_1, V_2), to otrzymamy wszystkie krawędzie wychodzące ze zbioru V_1 (i wchodzące do V_2).

2.3 Twierdzenie

Założenia:

G=(V,E)

C - przestrzeń cykli w G

C*- przestrzeń cięć w G

Teza:

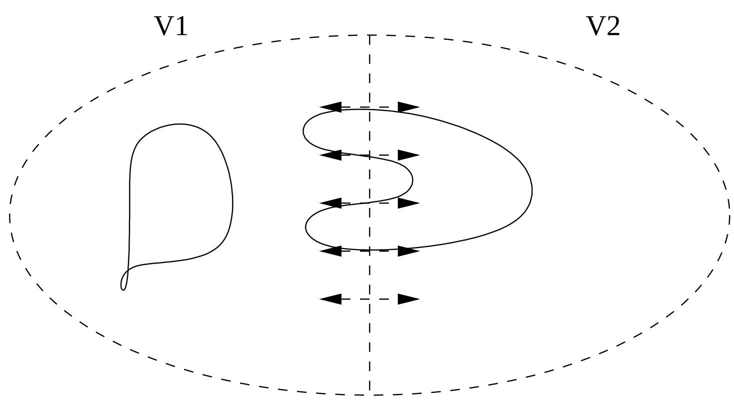
$C = C^{*\perp}$

$C^* = C^{\perp}$

Dowód:

1. C$\subseteq$ $C^{*\perp}$ i C*$\subseteq$C$^{\perp}$

Cykl ma parzystą liczbę krawędzi z każdym cięciem, bo albo jest z nim rozłączny albo musi przejść parzystą liczbą razy.



Czyli dla każdego cyklu c i cięcia t mamy <c, t> = 0, czyli każde cięcie należy do C$^{\perp}$ i każdy cykl należy do C$^{*\perp}$.

2. C$^{*\perp}$ $\subseteq C$

Pokażemy, że dla każdego zbioru F$\notin$C, F nie należy do C$^{*\perp}$.

Jeżeli F nie należy do C, to znaczy, że (V,F) zawiera wierzchołek v o stopniu nieparzystym (ze spostrzeżenia 2).

<E(v),F> = 1 (czyli liczba wspólnych krawędzi wszystkich krawędzi wychodzących z v i nieparzystej ich liczby należącej do F jest nieparzysta), co implikuje, że F$\in$C$^{*\perp}$.

3. C$^{\perp}$ $\subseteq C^*$

???

2.4 Twierdzenie (Mengera???)

Założenia:

G - spójny graf

n - liczba wierzchołków G

m - liczba krawędzi G

Teza:

1. dim(C(G)) = m-n+1

2. dim(C^*(G)) = n-1

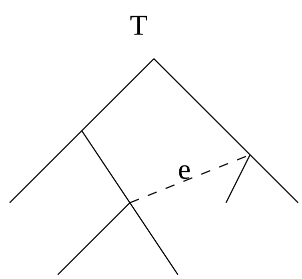
Dowód:

dim C + dim C^* = m, bo $C^* = C^\perp$

Wystarczy pokazać co najmniej m-n+1 liniowo niezależnych wektorów w C i co najmniej n-1 liniowo niezależnych wektorów w C^*.

Weźmy dowolne drzewo rozpinające T w G (istnieje, bo G spójny).

Dla każdej krawędzi e$\notin$T niech C_e będzie cyklem powstałym z dołączenia e do drzewa T.



Takie cykle są liniowo niezależne[2], bo każdy zawiera krawędź e nie występującą w pozostałych (np.współczynnik odpowiadający e w bazie kanonicznej przestrzeni krawędziowej = 1).

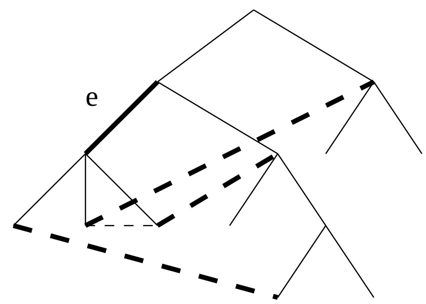
Jest ich tyle, co krawędzi niedrzewowych. Krawędzi drzewowych jest n-1, czyli pozostałych jest m-(n-1) = m-n+1.

Weźmy teraz dla każdej krawędzi drzewowej e cięcie określone następująco:

Wyjęcie e rospójnia nam T na dwa zbiory wierzchołków V_1 i V_2.

Jako cięcie bierzemy wszystkie krawędzie biegnące między nimi - jest to jedna krawędź drzewowa e, reszta to krawędzie niedrzewowe.

[2] *Nie* oznacza to, że <c1,c2>=0 !



Cięcia tej postaci są liniowo niezależne, bo każde zawiera tylko jedną krawędź drzewową.

Krawędzi drzewowych jest n-1, więc i niezależnych wektorów jest n-1.

♠

3 Domknięcie przechodnie

3.1 Definicje

G - graf pełny

(S, $\oplus$,$\odot$, 0, 1)

l: E-> S - funkcja wagi krawędzi

m - marszruta

c(m)=iloczyn ($\odot$) wag krawędzi na m

c(v,w) = $\sum_{m-marszruta\, z\, v\, do\, w} c(m)$

4 s-t numeracja

4.1 Definicja

G - dwuspójny graf n-wierzchołkowy

s-t - krawędź w G

Numerację wierzchołków G nazywamy *s-t numeracją*, gdy:

1. s ma numer 1
2. t ma numer n
3. każdy inny wierzchołek ma sąsiada o numerze większym i mniejszym

4.2 Algorytm znajdowania s-t numeracji

G - graf dwuspójny

Budujemy na grafie DFS o korzeniu w t.

Utożsamiamy wierzchołki z ich numerami w kolejności odwiedzania DFS-em.

Obliczamy funkcję *low*.

Nadajemy s i t oraz krawędzi s-t etykietę "stary".

Pozostałym nadajemy etykietę "nowy".

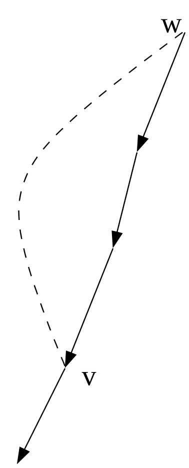
4.2.1 Algorytm PATHFINDER

Dane:

v - wierzchołek startowy

Rozpatrujemy kolejno następujące przypadki:

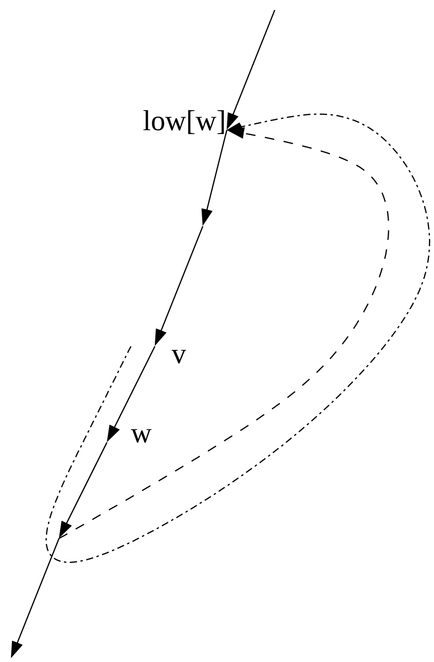
1. Istnieje "nowa" powrotna krawędź v-w t.ż. $w < v$



Zaznaczamy v-w jako "starą".

Zwracamy [v,w].

2. Istnieje "nowa" krawędź drzewowa v->w

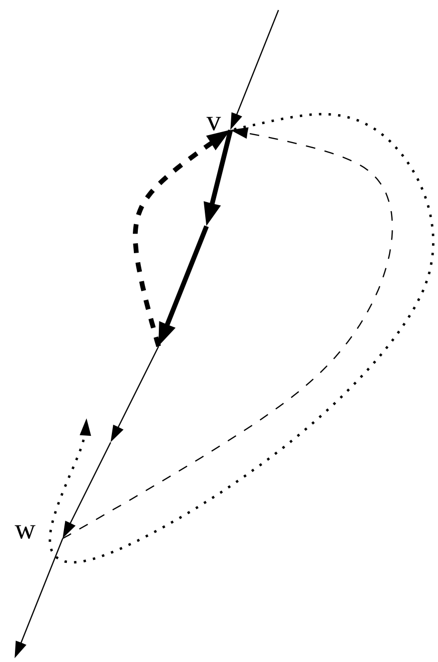


Znajdujemy ścieżkę do low[w]:

Zaznaczamy v-w jako "starą".

Wędrujemy nowymi krawędziami w dół aż znajdziemy krawędź powrotną sięgającą ponad v??? (taka krawędź istnieje, bo graf jest dwuspójny, a my w algorytmie nie ruszamy krawędzi powrotnych prowadzących w dół). Po drodze zaznaczamy wierzchołki i krawędzie jako "stare".??? Zwracamy ścieżkę, którą przeszliśmy.

3. Istnieje "nowa" krawędź powrotna v-w t.ż. $v < w$



Zaznaczamy v-w jako starą. Przechodzimy nią w dół, po czym idziemy krawędziami drzewa aż napotkamy "stary" wierzchołek (zaznaczając wierzchołki i krawędzie jako "stare"). Zwracamy ścieżkę, którą przeszliśmy.

Należy zauważyć, że do tego punktu przechodzimy po sprawdzeniu dwóch poprzednich, tzn. że wszystkie krawędzie drzewowe wychodzące z v są "stare".

4. W przeciwnym wypadku (wszystkie wychodzące krawędzie są stare) zwracamy ścieżkę pustą.

4.2.2 Lemat

Algorytm PATHFINDER zwraca ścieżkę składającą się z nowych krawędzi, zaczynającą się i kończącą w wierzchołku "starym".

Jeżeli v jest "stary", to wszyscy jego poprzednicy w drzewie też są starzy.

4.2.3 Algorytm znajdowania s-t numeracji

Dane:

G - graf dwuspójny

s,t - wierzchołek początkowy i końcowy, połączone krawędzią

Algorytm ST-NUMBER:

```
zaznacz s,t oraz s-t jako “stare”, pozostałe jako “nowe”.
PUSH(S,t);
PUSH(S,s);
i := 0;
while not empty(S)
{
   v := POP(S);
   P := PATHFINDER(v);
   if length(P)<>0 then
     for k := length(P)-1 downto 1 do PUSH(P(k)); //ostatni nie jest wkładany, pierwszy to v
   else
   {
      i := i+1;
      nr[v] := i;
   }
}
```

4.2.4 Lemat (poprawność s-t numeracji)

Teza:

Algorytm ST-NUMBER poprawnie oblicza s-t numerację G.

Dowód:

1. W każdej chwili każdy wierzchołek występuje na stosie co najwyżej raz.
2. Po umieszczeniu v na stosie wszystkie wierzchołki leżące pod nim na stosie otrzymają swój numer po nadaniu numeru wierzchołkowi v.
3. Żaden wierzchołek nie zostanie (na stałe - po znalezieniu pustej ścieżki) usunięty ze stosu zanim wszystkie krawędzie z nim incydentne nie staną się “stare”.

Ad 1. Wierzchołki są wkładane gdy znajdą się na ścieżce P i nie są na niej ostatnie, co znaczy, że przed wywołaniem PATHFINDER były “nowe”, a teraz są “stare”.

Ad 2. Ponieważ wierzchołki znajdują się na stosie co najwyżej raz, wierzchołek aktualnie przetwarzany albo otrzyma numer albo znajdzie się znowu na czubku stosu.

Ad 3. Wierzchołek zostaje na stałe usunięty ze stosu, gdy algorytm PATHFINDER zwróci pustą ścieżkę, co oznacza, że nie ma już wychodzących nowych krawędzi.

5 Spójność grafów

5.1 Dwuspójne składowe

5.1.1 Spostrzeżenie

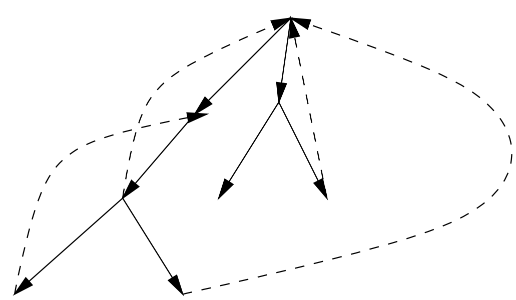
Założenia:

G=(V,E) - graf spójny, niezorientowany

T=(V,F) - drzewo DFS na tym grafie

Teza:

Krawędzie powrotne w drzewie T prowadzą od wierzchołka do jego przodka, czyli nie ma krawędzi biegnących między różnymi poddrzewami.



5.1.2 Lemat

Założenia:

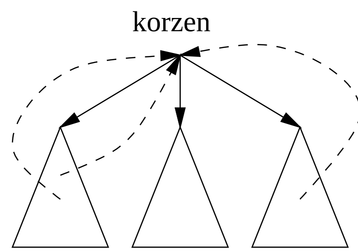
G=(V,E) - graf spójny, niezorientowany

T=(V,F) - drzewo DFS na tym grafie

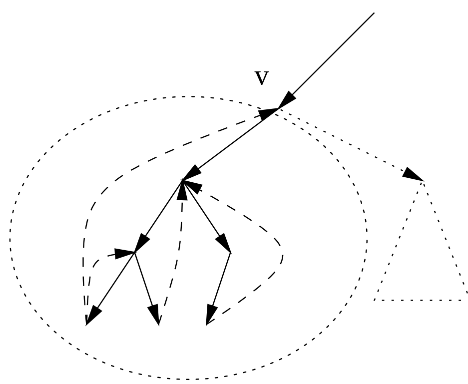
Teza:

Wierzchołek a jest wierzchołkiem rozdzielającym, gdy

1. a jest korzeniem T i ma więcej niż jednego syna w T



2. a nie jest korzeniem i istnieje taki następnik s wierzchołka a, że nie istnieje krawędź pomiędzy żadnym potomkiem s (łącznie z s) a właściwym przodkiem a (czyli wszystkie krawędzie powrotne prowadzą co najwyżej do a[3].



5.1.3 Wniosek

Wierzchołek v różny od korzenia jest rozdzielający <=> posiada następnik s t.ż. low[s] >= v

5.1.4 Algorytm

SEARCHB(v):

Zaznaczamy v jako odwiedzony

count++;

numer_dfs[v] := count;

low[v] := numer_dfs[v];

Dla każdego sąsiada w wierzchołka v:

[3] KL: *istnieje* taki następnik, czyli pozostałe nie muszą spełniać tego założenia

1. Jeśli w jest nieodwiedzony (nie należy do drzewa)

v staje się ojcem w

Wkładamy v->w na stos

SEARCHB(w);

Jeśli low[w] >= numer_dfs[v] to zdejmujemy i wypisujemy krawędzie ze stosu aż napotkamy krawędź v->w[4].

Uaktualniamy low[v] - low[v] := MIN(low[v], low[w]).

2. v->w jest krawędzią powrotną (prowadzi do wierzchołka odwiedzonego)

Wstawiamy v->w na stos.

Uaktualniamy low[v] - low[v] := MIN(low[v], low[w]).

3. W przeciwnym wypadku nic nie robimy

Inicjalizacja:

count := 0;

SEARCHB(1);

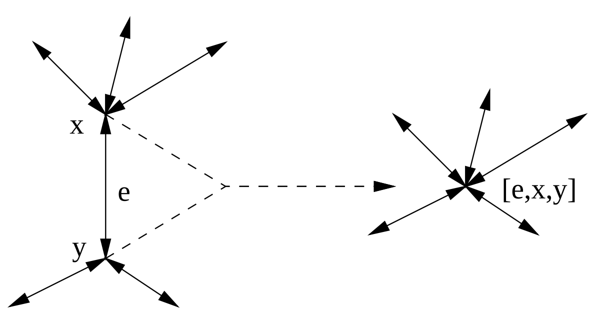
Czas:

O(|V|+|E|)

5.2 3-spójność

5.2.1 Oznaczenia

G/e - graf powstały z G poprzez sklejenie końców krawędzi e



[4]Tworzą one dwuspójną składową

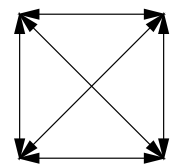
5.2.2 Fakt 1[5]

Teza:

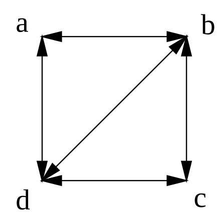
Graf K_4 jest jedynym 4-wierzchołkowym grafem 3-spójnym.

Dowód:

1. Graf K_4 jest 3-spójny.



2. Graf powstały z K_4 przez usunięcie którejkolwiek krawędzi nie jest 3-spójny.



Usunięcie wierzchołków b i d odseparowuje a i c.

5.2.3 Lemat 1

Założenia:

G - graf 3-spójny

$|G| > 4$

Teza:

G zawiera krawędź e, t.ż. G/e jest nadal 3-spójny.

Dowód:

???

5.2.4 Twierdzenie 1 (Tutte)

Teza:

Graf G jest 3-spójny $<=>$ istnieje ciąg grafów G_0, ..., G_n t.ż. :

[5] KL

1. $G_0 = K_4$
2. $G_n = G$
3. w G_{i+1} istnieje krawędź x-y, t.ż stopień x i stopień y są >=3 i dla każdego i $G_i = G_{i+1}/x - y$

Dowód:

=>

Graf G=G_n jest 3-spójny.

Skonstruujmy ten ciąg grafów następująco:

1. $|G_i| = 4$

Zaprzestajemy konstrukcji. Otrzymanym grafem jest K_4, ponieważ jest to jedyny 3-spójny graf 4-wierzchołkowy.

2. $|G_i|>4$

Dodajemy jako graf i-1 graf powstały ze zlepienia pewnej krawędzi x-y grafu i-tego.

Taki graf jest 3-spójny z lematu.

Stopnie x i y w G_i są >= 3, ponieważ graf i-ty jest 3-spójny.

<=

1. Graf K_4 jest 3-spójny.

2. Graf G_{i+1} powstały ze rozbicia pewnego wierzchołka v 3-spójnego grafu G_i na wierzchołki x i y połączone krawędzią jest 3-spójny, a stopnie x i y są >=3.

Wystarczy pokazać, że rozbicie wierzchołka nie wpływa na ścieżki między dwoma wierzchołkami grafu i że między x i y są co najmniej 3 ścieżki.

Żeby pokazać, że stopnie x i y są >=3 trzeba pokazać, że stopień v był >=4 (pokazać, że istnieje taki wierzchołek w G_i).

5.2.5 Twierdzenie 2 (Tutte)

Teza:

Przestrzeń pseudocykli (grafów o stopniach parzystych) grafu 3-spójnego jest generowana przez jego nieseparujące[6] cykle indukowane.

Dowód:

???

[6] tzn. że istnieją 2 wierzchołki w różnych spójnych składowych???

6 Przepływy

6.1 Twierdzenie 1

Założenia:

G - graf niezorientowany

s, t - niesąsiednie wierzchołki w G

Teza:

Minimalna liczba wierzchołków separujących s od t, różnych od s i t, jest równa maksymalnej liczbie niezależnych[7] ścieżek z s do t.

6.2 Twierdzenie 2

Założenia:

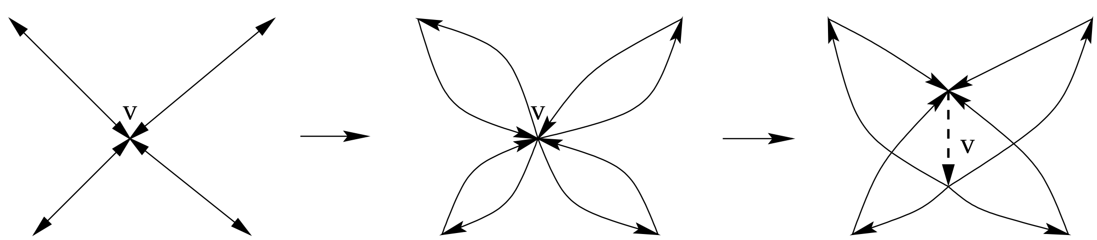
G - graf niezorientowany

s, t - wierzchołki w G

Teza:

Minimalna liczba krawędzi separujących s od t jest równa maksymalnej liczbie krawędziowo rozłącznych ścieżek z s do t.

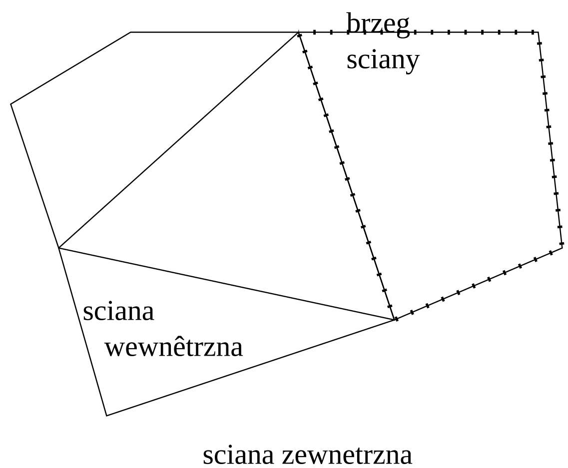
6.3 Związek między przepływami a k-spójnością grafu



[7]KL: wierzchołkowo

7 Grafy planarne

7.1 Definicje



7.2 Twierdzenia

7.2.1 Twierdzenie Eulera

Założenia:

G - spójny graf płaski

n - liczba wierzchołków w G

m - liczba krawędzi

f - liczba ścian

Teza:

n+f = m+2

Dowód:

m >= n-1 bo graf G jest spójny

Indukcja po m:

1. m = n-1, to f=1 (bo G jest drzewem)

2. m > n-1

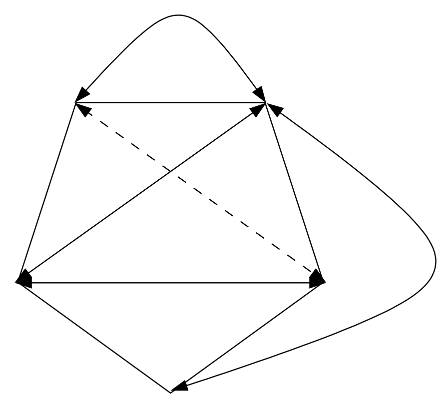
Wyjmijmy z grafu G jedną krawędź z jakiegoś cyklu (cykl istnieje, bo G jest spójny i nie jest drzewem). W tak otrzymanym grafie liczba wierzchołków się nie zmienia, liczba krawędzi jest o 1 mniejsza i liczba ścian jest o 1 mniejsza, jest on spójny i płaski, więc możemy zastosować założenie indukcyjne, czyli:

n+f-1=m-1+2, czyli

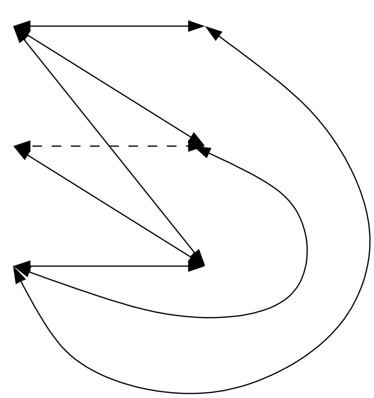
n+f=m+2

7.2.2 Wnioski z twierdzenia Eulera

1. W każdym grafie planarnym m <= 3n-3, a dla n>=3 m<=3n-6
2. Graf K_5 i $K_{3,3}$ nie są planarne



K_5



$K_{3,3}$

7.2.3 Lemat

Teza:

W każdym grafie planarnym istnieje wierzchołek stopnia ≤ 5

Dowód:

Załóżmy przeciwnie: każdy wierzchołek jest stopnia co najmniej 6. Wtedy liczba krawędzi w grafie jest większa lub równa (6*n)/2 (bo każda krawędź ma dwa końce), czyli 3n. Z wniosków z twierdzenia Eulera m <= 3n-3, więc graf nie może być planarny.

8 Kolorowanie grafów

8.1 Definicje

$\Delta(G)$- maksymalny stopień wierzchołka w G

N(v) - sąsiedzi v

d(v) - stopień v

8.2 Twierdzenie 1

Założenia:

G - graf

k = $\max\limits_{H} \delta(H)$[8] , gdzie H to podgraf G

Teza:

X(G)[9] <= k+1

Dowód:

Indukcja:

Dla grafu o rozmiarze <=k możemy go pokolorować k kolorami.

Krok indukcyjny:

Weźmy wierzchołek v o stopniu k. Po usunięciu v z grafu maksimum z minimalnych stopni grafu się nie zwiększy, bo otrzymany graf jest podgrafem G (czyli uwzględniliśmy go w liczeniu maksimum). W tak otrzymanym grafie możemy pokolorować wierzchołki k+1 kolorami, a ponieważ v ma k sąsiadów, więc wstawiamy go do grafu i malujemy innym kolorem.

♠

[8] $\delta(H)$ - minimalny stopień wierzchołka w podgrafie H

[9] Liczba chormatyczna, czyli minimalna liczba kolorów potrzebna do pokolorowania grafu

8.3 Twierdzenie Brooksa

Założenia:

G - spójny graf

G nie jest grafem pełnym[10]

G nie jest cyklem o nieparzystej długości[11]

Δ - maksymalny stopień w G

Teza:

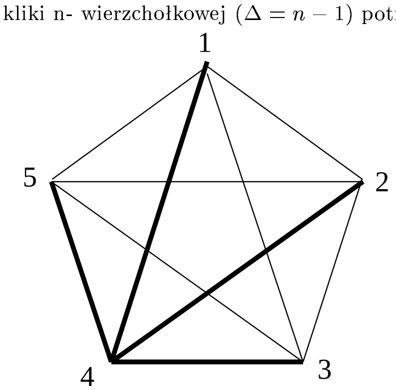
X(G)$<=\Delta$

Dowód:

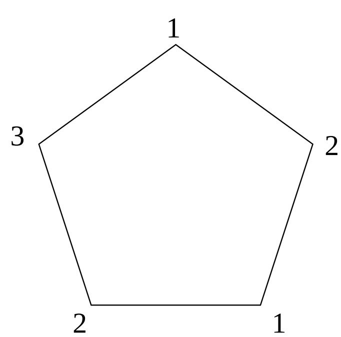
1. Zakładamy, że G jest dwuspójny.

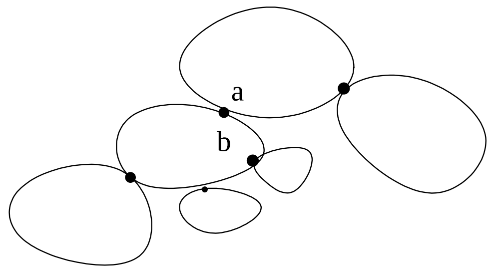
Jeżeli G nie jest dwuspójny, to możemy podzielić go na dwuspójne składowe (tworzą one zbiór drzew - punktami wspólnymi są wierzchołki). Kolorujemy każdą dwuspójną składową. Następnie poczynając od korzenia sprawdzamy, czy kolor ustalony w punkcie łączenia ojca z synem się zgadza . Załóżmy, że ojciec ustalił w tym punkcie kolor a, a syn - b.

[10]Bo do pokolorowania kliki n- wierzchołkowej ($\Delta = n - 1$) potrzeba n=$\Delta + 1$ kolorów



[11]Cykl ma stopień każdego wierzchołka = 2, a jeśli jest nieparzystej długości, to muszą być 3 kolory





Wówczas zamieniamy w synie kolor a z b.

Powtarzamy to aż poprawimy całe drzewo.

2. Zakładamy, że G jest Δ-regularny[12].

Jeżeli G nie jest Δ-regularny, to korzystamy z twierdzenia 1:

H zawiera wierzchołek o stopniu $<\Delta$, więc ma minimalny stopień co najwyżej stopień Δ-1.

Każdy podgraf właściwy H ma stopień mniejszy od Δ:

Załóżmy, że jakiś właściwy podgraf F grafu G ma minimalny stopień $= \Delta$.

Czyli każdy wierzchołek ma stopień maksymalny w danym grafie, a ponieważ graf G jest spójny, więc istnieje conajmniej jedna krawędź od jednego z wierzchołków F do reszty grafu, czyli ten wierzchołek w G miałby stopień $>\Delta$, a Δ jest maksymalnym stopniem w G.

Czyli każdy podgraf ma stopień minimalny co najwyżej Δ-1. Czyli z twierdzenia 1 możemy go pokolorować Δ kolorami.

3. $\Delta>=3$

Jeżeli $\Delta=0$ to graf G nie jest spójny.

Jeżeli $\Delta=1$, to G jest parą wierzchołków, czyli cyklem o nieparzystej długości.

Jeżeli $\Delta=2$, to G jest ścieżką, którą da się pokolorować $\Delta=2$ kolorami.

4. Mamy graf spełniający powyższe warunki

(a) G jest 3-spójny

Niech v_n będzie dowolnym wierzchołkiem w G.

v_n posiada dwóch sąsiadów niepołączonych krawędzią:

Załóżmy, że wszyscy sąsiedzi v_n są połączeni krawędziami.

v_n ma dokładnie Δ sąsiadów. Każdy z tych sąsiadów jest połączony z v_n i Δ-1 pozostałymi sąsiadami v_n. Ponieważ Δ jest maksymalnym stopniem wierzchołka w grafie, więc żaden z sąsiadów v_n ani on sam nie jest połączony z

[12] Każdy wierzchołek ma stopień Δ

innym wierzchołkiem, co oznacza, że albo G nie jest spójny, albo jest kliką (co jest sprzeczne z założeniami).

⋄

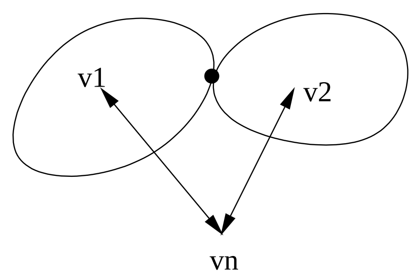
Oznaczmy przez v_1 i v_2 tych dwóch sąsiadów v_n nie połączonych krawędzią.

Usunięcie ich z grafu nie rozspójni go, bo jest 3-spójny.

(b) G nie jest 3-spójny

G jest 2-spójny z wcześniejszych założeń.

Weźmy jako v_n wierzchołek należący do separatora dwóch 3-spójnych składowych, jako v_1 jego sąsiada z jednej spójnej składowej, a jako v_2 z drugiej.

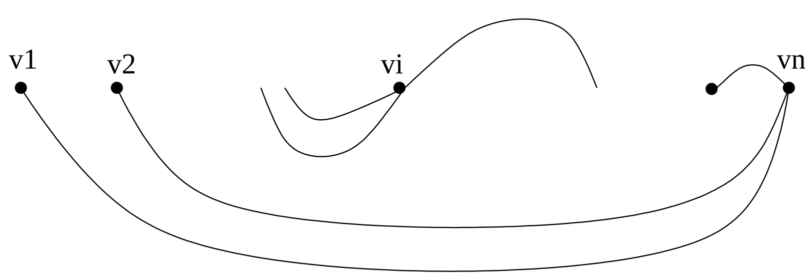


Usunięcie v_1 i v_1z grafu nie rozspójni go, bo leżą w oddzielnych 3-spójnych składowych.

Kolorowanie:

Ustawmy wierzchołki G-{v_1,v_2} w ciąg v_3 .. v_n taki, że każdy wierzchołek v_i ma sąsiada wśród wierzchołków v_{i+1}, ..., v_n. Możemy tak zrobić, bo G- jest spójny {v_1,v_2} i Δ>=3 (n-1 wierzchołek istnieje, bo v_n ma jeszcze conajmniej jedną krawędź oprócz prowadzących do v_1 i v_2).

Następnie dołączmy do ciągu na początku wierzchołki v_1 i v_2.



Każdy wierzchołek ma conajmniej jedną krawędź prowadzącą na prawo, więc co najwyżej Δ-1 prowadzących na lewo.

Kolorujemy v_1 i v_2 tym samym kolorem. Następnie dla każdego wierzchołka v_i dla i=3..n-1 kolorujemy go kolorem, którego nie mają jego sąsiedzi z lewej (jest ich $<=\Delta$, więc zużywamy co najwyżej Δ kolorów). v_n ma Δ sąsiadów po lewej, ale ponieważ dwóch sąsiadów v_n ma ten sam kolor, więc też możemy go pokolorować jednym z Δ kolorów.

♠

8.4 Twierdzenie Vizinga

Założenia:

G - graf

Δ - maksymalny stopień wierzchołka w G

X'(G) - liczba chromatyczna krawędziowa G[13]

Teza:

$\Delta <= X'(G) <= \Delta+1$

Dowód:

$\Delta <= X'(G)$, bo w wierzchołku o maksymalnym spotyka się Δ krawędzi, więc muszą mieć one różny kolor.

$X'(G) <= \Delta+1$

Indukcja po liczbie krawędzi:

$|E|=0$ to możemy pokolorować 0 kolorami.

$|E| > 0$

Załóżmy, że G-{e}jest pokolorowany $\Delta+1$ kolorami.

Pokażemy, jak rozszerzyć to kolorowanie na G.

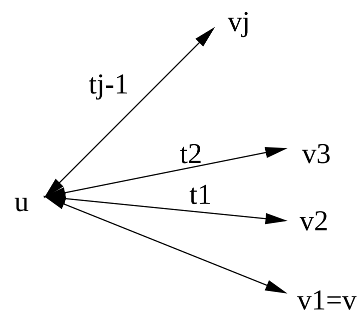
W każdym wierzchołku jest przynajmniej jeden kolor wolny (bo kolorów jest $\Delta+1$, a liczba krawędzi wychodzących z wierzchołka to Δ).

Niech e=(u,v). Chcemy ją pokolorować tak, by nie popsuć kolorowania w innych wierzchołkach.

Ustawmy sąsiadów u w maksymalny taki ciąg, że v_1=v i dla i>1 kolor krawędzi u-v_i jest kolorem wolnym dla wierzchołka v_{i-1} (oznaczmy kolor wolny w wierzchołku v_k przez t_k).

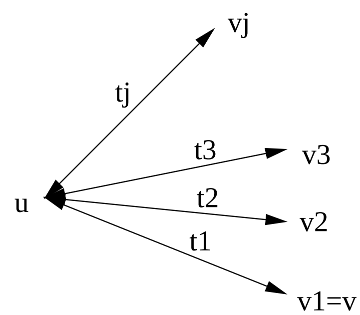
Załóżmy, że ostatni wierzchołek ciągu ma numer j.

[13] Minimalna liczba kolorów potrzebnych do pokolorowania krawędziowego G



1. Żadna krawędź wychodząca z u nie jest pokolorowana kolorem t_j.

Kolorujemy krawędź u-v_i kolorem t_{i+1} dla i=1..j.

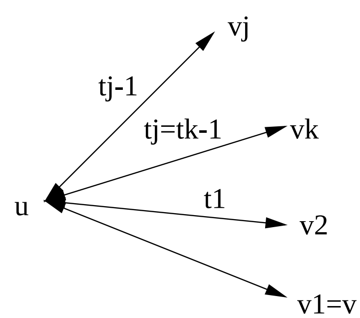


To kolorowanie jest dobre ponieważ t_i był wolny w v_i.

2. Jedna z krawędzi wychodzących z u jest pokolorowana kolorem t_j

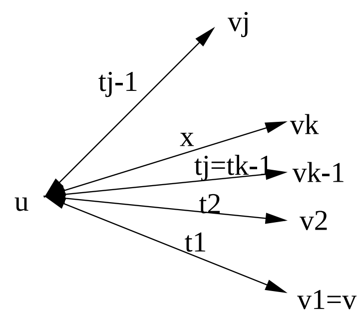
Załóżmy, że jest to krawędź do wierzchołka o numerze k>j. Wówczas moglibyśmy przedłużyć nasz ciąg ustawiając tę krawędź jako j+1-szą, co jest sprzeczne z założeniem o maksymalności naszego ciągu.

Zatem musi to być krawędź biegnąca do wierzchołka v_k, k<j.



Kolorujemy krawędzie u-v_i kolorem t_i dla i<k.

Krawędź u-v$_k$ zaznaczamy jako "bez koloru".

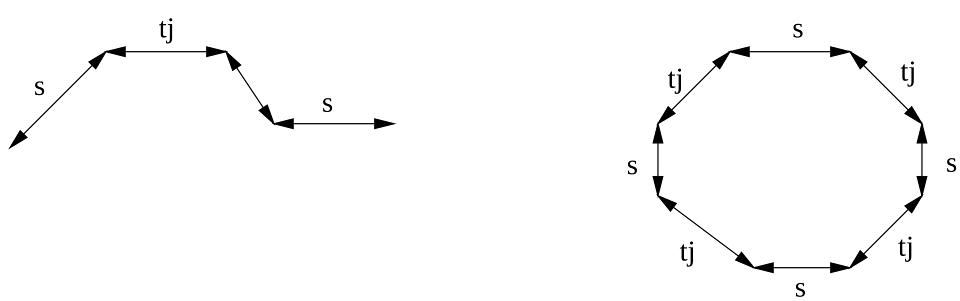


Musimy teraz pokolorować na nowo u-v$_k$.

Niech s będzie kolorem wolnym w u.

Rozważmy podgraf H(t$_j$,s) grafu G rozpięty na krawędziach G o kolorach s i t$_j$.

Podgraf H jest zbiorem ścieżek i cykli (parzystej długości) (bo krawędzie o tym samym kolorze nie mogą się spotkać w jednym wierzchołku, więc w każdym wierzchołku może się kończyć co najwyżej jedna krawedź o kolorze s i co najwyżej jedna o kolorze t$_j$).



Ponieważ s jest wolnym kolorem w u, więc do u wchodzi dokładnie jedna krawędź z H (krawędź u-v$_{k-1}$), czyli u jest końcem ścieżki.

Ponieważ krawędź u-v$_k$ była koloru t$_j$, więc do wierzchołka v$_k$ może wchodzić co najwyżej tylko jedna krawędź s, więc v$_k$ też jest końcem ścieżki (lub jest izolowany w H).

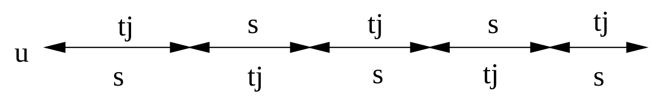
Ponieważ t$_j$ jest wolny w v$_j$, więc analogicznie więc v$_j$ też jest końcem ścieżki (lub jest izolowany w H).

Z tego wynika, że u, v$_j$ i v$_k$ nie leżą w jednej spójnej składowej w H (bo albo któryś jest izolowany, albo ścieżka by miała 3 końce).

Ponieważ u jest końcem ścieżki (bo jest krawędź o kolorze t$_j$ wychodząca z u), więc leży w spójnej składowej z v$_j$, v$_k$ lub z żadnym z nich.

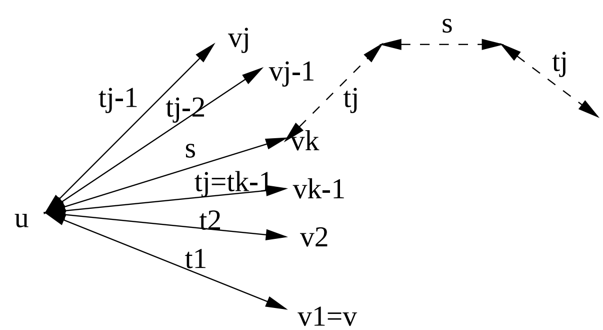
Zauważmy, że możemy zamienić na ścieżce wychodzącej z u kolor t_j z kolorem s.

Możemy tak zrobić, bo kolorowanie zapewnia, że w jednym wierzchołku nie spotykają się krawędzie o tym samym kolorze, a my tylko zamieniamy jeden z drugim.



1. Z żadnym lub z v_j

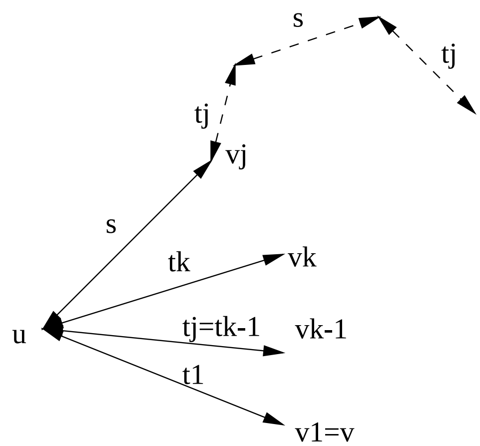
W spójnej składowej zawierającej v_k (która nie zawiera u) dokonujemy zamiany kolorów s <-> t_j. W ten sposób v_k nie jest incydentny z krawędzią o kolorze s, więc kolorujemy u-v_k kolorem s.



2. Z v_k

Ścieżka wchodząca do v_j wchodzi tam krawędzią o kolorze s. Przekolorowujemy więc ją zamieniając s<->t_j i w ten sposób s jest wolny w v_j.

Kolorujemy, więc krawędzie u-v_i kolorami t_i dla i=k..j-1, a u-v_j kolorem s.



♠

8.5 Algorytm aproksymacyjny dla grafów 3-kolorowalnych

8.5.1 Fakt 1 (algorytm brute-force)

Każdy graf G można pokolorować $\Delta(G) + 1$ kolorami.

Dowód:

Kolorujemy wierzchołek kolorem, którego nie mają sąsiedzi.

8.5.2 Fakt 2

Założenia:

G - graf 3-kolorowalny

Teza:

Dla każdego wierzchołka v z G podgraf generowany przez N(v) jest dwudzielny.

Dowód:

Wierzchołek v ma inny kolor niż jego sąsiedzi, czyli jego sąsiedzi mogą mieć dwa kolory. Ponieważ wierzchołki sąsiednie nie mogą mieć tego samego koloru, więc zbiór sąsiadów jest dwudzielny.

8.5.3 Algorytm

Dane:

G - graf 3-kolorowalny

Algorytm:

n := |G|;

i := 1;

while $\Delta(G) \geq \sqrt{n}$ do

{

v := wierzchołek o maksymalnym stopniu

H - podgraf oparty na N(v)

Kolorujemy H kolorami i, i+1

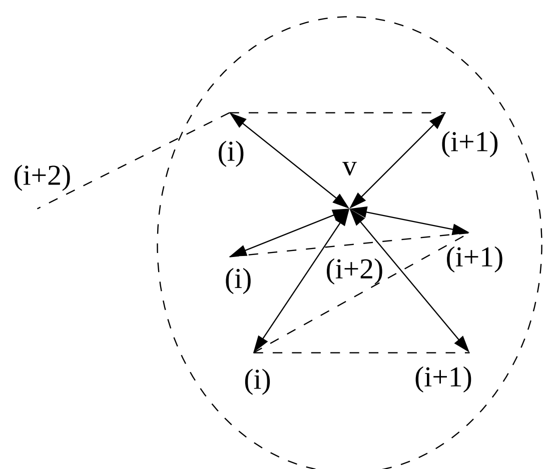
Kolorujemy v kolorem i+2;

i := i+2;

G := G - N(v) - {v};

}

BRUTE_FORCE(G,i);



Czas: KL

Pierwsza faza:

Koszt liniowy (każdy wierzchołek i każda krawędź są przetwarzane dokładnie raz).[14]

Brute force:

[14]Bardziej skomplikowane tłumaczenie: Za każdym razem usuwamy $\Delta(G)+1$ wierzchołków. Ponieważ $\Delta(G) \geq \sqrt{n}$, więc tych faz może być co najwyżej $\sqrt{n}$. Koszt kolorowania H jest liniowy, więc czas pierwszej fazy jest$O(\sqrt{n}\sqrt{n}) = O(n)$. Dochodzi jeszcze koszt przejrzenia krawędzi, czyli O(|E|)

Koszt liniowy.

W sumie: O(|V|+|E|)

Liczba kolorów:

Pierwsza faza:

$\sqrt{n}$ faz po 2 kolory =2$\sqrt{n}$

Brute force:

$\Delta(G) < \sqrt{n}$. Z faktu 1 można pokolorować G $\sqrt{n}$ kolorami.

W sumie: 3$\sqrt{n}$ kolorów

8.6 Algorytm aproksymacyjny dla grafów k-kolorowalnych

8.6.1 Fakt 3 (uogólnienie faktu 2 dla grafów k-kolorowalnych)

Założenia:

G - graf k+1-kolorowalny

Teza:

Dla każdego wierzchołka v z G graf H indukowany przez N(v) jest k-kolorowalny.

Dowód:

Żaden z wierzchołków z N(v) nie może mieć tego samego koloru, co v. Ponieważ graf, jest k+1 kolorowalny, a v zużywa jeden kolor, więc H jest k-kolorowalny.

8.6.2 Algorytm

Dane:

G - graf

k - liczba kolorów wystarczająca do pokolorowania G

i - kolor od którego można kolorować

Zwraca:

Liczba kolorów zużytych do kolorowania G.

Algorytm KCOLOR(k, G, i):

Jeśli k=2 kolorujemy G kolorami i, i+1 i zwracamy 2.

Wpp.

n := |G|;

while $\Delta(G) \geq f_k(G)$ do

```
{
    v := wierzchołek o maksymalnym stopniu
    H - podgraf indukowany przez N(v)
    j := KCOLOR(k-1, H, i);
    Kolorujemy v kolorem i+j
    i := i+j;
    G := G - N(v) - {v};
}
BRUTE_FORCE(G,i);
```

Zwracamy liczbę kolorów zużytych do kolorowania G.

Czas: O(k*(|V|+|E|))

Liczba kolorów: $kf_k(n) = kn^{1-\frac{1}{k}}$

8.6.3 Funkcja f_k

f_k jest określona następująco:

$$\begin{cases} f_{k+1} = \frac{n}{f_{k+1}(n)} f_k(f_{k+1}(n)) \\ f_2(n) = 1 \end{cases} \quad ???$$

8.7 Algorytm kolorowania grafów planarnych

8.7.1 Twierdzenie o 5-kolorowalności grafów planarnych

Teza:

Każdy graf planarny jest 5-kolorowalny.

Dowód:

Indukcja po n=|V|.

1. n<5

Oczywiste

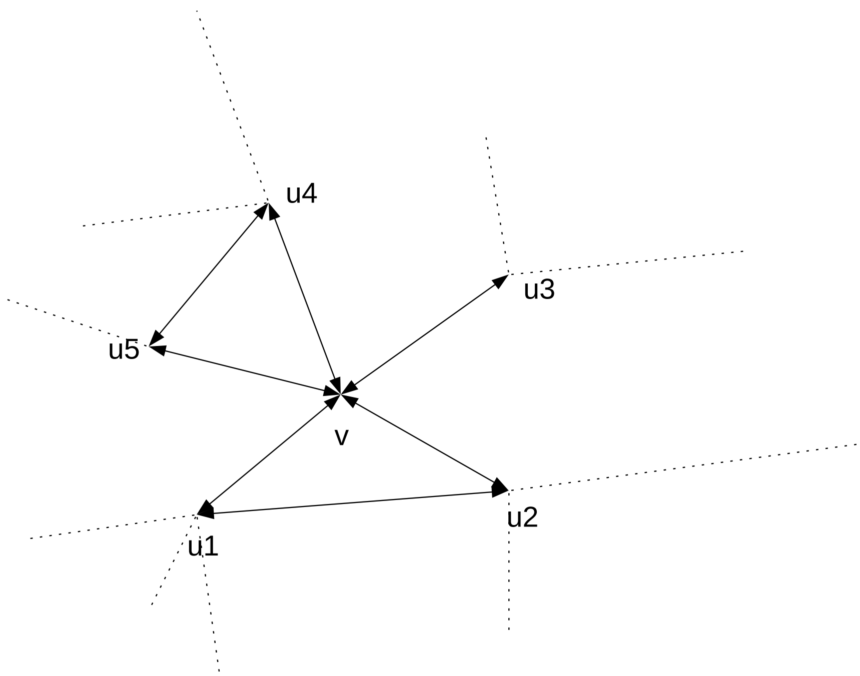
2. n>5

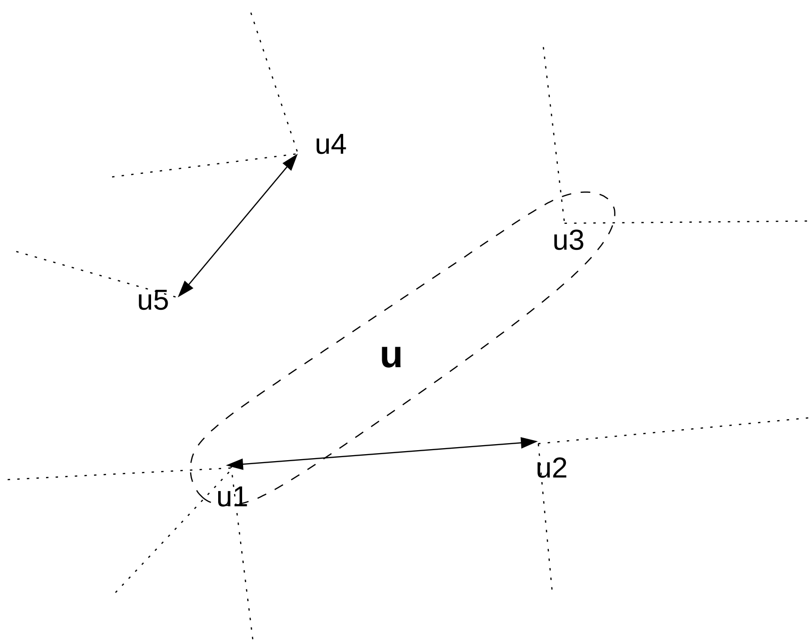
W G istnieje wierzchołek v o stopniu <= 5.

Niech k=deg(v).

Jeżeli k<=4, to sąsiedzi v zużywają 4 kolory, więc kolorujemy v piątym.

Jeżeli k=5, to istnieją dwaj sąsiedzi v nie połączeni krawędzią (bo inaczej graf rozpięty na sąsiadach v byłby K_5, więc nie byłby planarny). Mogą one otrzymać ten sam kolor, więc utożsamiamy te dwa wierzchołki, jednocześnie usuwając v i kolorujemy v wolnym kolorem.





8.7.2 Lemat

Założenia:

G - graf planarny

Wszystkie wierzchołki w G mają stopień >= 5

Teza:

W G istnieje wierzchołek o stopniu 5, który posiada dwóch nie połączonych krawedzią sąsiadów o stopniach <= 11.

Dowód:

? trzeba?

8.7.3 Algorytm

Wierzchołek jest *redukowalny,* jeżeli stopień v <=4 albo stopień v=5 i v posiada dwóch niepołączonych sąsiadów o stopniach <= 11.

Struktury danych:

L[v] - lista sąsiadów v, dwukierunkowa, cykliczna, z wiązaniami krzyżowymi[15]

[15] Tzn. ze wskaźnikiem na wystąpienie v na liście sąsiadów jego sąsiada

Q - kolejka wierzchołków redukowalnych o stopniu <=4

R - kolejka wierzchołków redukowalnych o stopniu 5

S - stos wierzchołków usuwanych z grafu. Może zawierać:

1. parę [x, L[x]] - gdy usuwamy wierzchołek o stopniu <=4[16]
2. [t, x, L[x], y, L[y]] - gdy usuwamy wierzchołki x i y utożsamiając je z nowym wierzchołkiem t

Procedury pomocnicze:

Insert(v, P) - jeżeli v nie należy do struktury P, to go wstawiamy

Delete(v, P) - jeżeli v należy do struktury P, to go usuwamy

Consider(v):

1. deg(v)<=4 - przesuwamy v z kolejki R (5-stopniowych) do kolejki Q (4 i mniej-stopniowych)
2. deg(v)=5 - jeśli v jest redukowalny to go wstawiamy do R, wpp go stamtąd usuwamy[17]
3. deg(v)=11[18] - dla każdego sąsiada x wierzchołka v sprawdzamy, czy jest wierzchołkiem redukowalnym o stopniu 5 i jeżeli tak, to wstawiamy go do R.

Reduce(v):

Wstawiamy [v, L[v]] na stos S i usuwamy v z grafu. Następnie stosujemy Consider do każdego sąsiada v.

Identify(x,y):

Tworzymy nowy wierzchołek t, którego sąsiadami są sąsiedzi x i y. Na stos wstawiamy [t, x, L[x], y, L[y]] i usuwamy x i y z grafu. Następnie stosujemy Consider do t i wszystkich sąsiadów t.

Algorytm:

Faza 1: redukcja[19]

Oblicz stopnie wierzchołków i odpowiednio powstawiaj je do Q i R.

while |G|>5 do

---

[16] KL

[17] Wierzchołek może przestać być redukowalny gdy straci jednego sąsiada

[18] Consider wywołujemy usuwając każdy wierzchołek, więc nie przegapimy stosując tylko =, a nie <=

[19] KL

```
{
    if not empty(Q) then
    {
        v := Get(Q);
        Reduce(v);
    }
    else
    {
        v := Get(R);
        Niech x,y będą sąsiadami v o stopniach <= 11
        Reduce(v);
        Identify(x,y);
    }
}
```

Faza 2: kolorowanie[20]

Pokoloruj każdy wierzchołek G innym kolorem (jednym z pięciu, bo |G|<=5).

```
while not empty(S)
{
    v := Pop(S);
    if v=[x,L[x]] then
    {
        Pokoloruj v kolorem innym niż mają jego sąsiedzi
        Dodaj v do grafu
    }
    else
    {
        v=[t, x, L[x], y, L[y]];
        Pokoloruj x i y kolorem, który ma wierzchołek t
        Usuń t z grafu
        Dodaj x i y do grafu
    }
}
```

[20] KL

9 Skojarzenia w grafach

9.1 Definicje

Skojarzenie - podzbiór E w którym żadne dwie krawędzie nie mają wspólnych końców

Wierzchołek wolny - wierzchołek nie będący końcem żadnej krawędzi skojarzenia

Wierzchołek pokryty - przeciwieństwo wolnego

Ścieżka naprzemienna - ścieżka, której krawędzie na przemian należą i nie należą do skojarzenia

Ścieżka rozszerzająca - ścieżka naprzemienna, która zaczyna się i kończy w wierzchołku wolnym

9.2 Twierdzenia

9.2.1 Twierdzenie Hopcrofta

Założenia:

M - skojarzenie

M* - skojarzenie maksymalne

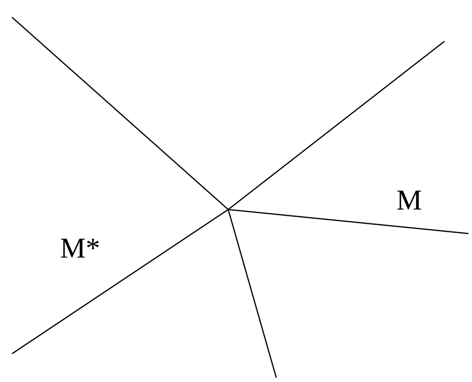
Teza:

W grafie istnieje |M*|-|M| rozłącznowierzchołkowych ścieżek rozszerzających względem M.

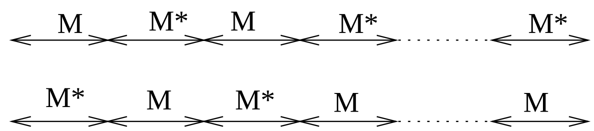
Dowód:

Rozważamy graf rozpięty na krawędziach $M^* \oplus M$.

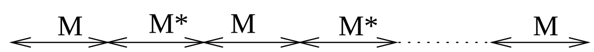
Krawędzie w tym zbiorze tworzą rozłączne ścieżki i cykle (ponieważ w jednym wierzchołku może znajdować się koniec co najwyżej jednej krawędzi M i jednej krawędzi M*, bo są to skojarzenia)



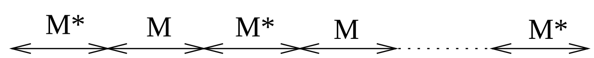
Ścieżki mogą być postaci:



Nie są rozszerzające względem M.
Liczba krawędzi z M i M*jest taka sama.

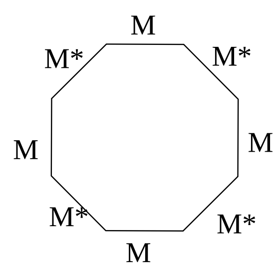


Nie może być takiej ścieżki , bo jest rozszerzająca względem M*



Taka ścieżka jest rozszerzająca.
Takich ścieżek jest |M*|-|M|, bo dla każdej takiej ścieżki jest o jedną krawędź M* więcej

Cykle:



Mogą być cykle tylko parzystej długości, bo inaczej spotykałyby się w jednym wierzchołku dwie krawędzie skojarzenia.
Taki cykl nie jest rozszerzający.
Liczba krawędzi M i M* jest taka sama.

9.2.2 Twierdzenie (o wierzchołku wolnym)

Założenia:

M - skojarzenie

p - wierzchołek wolny

W p nie zaczyna się żadna ścieżka rozszerzająca względem M.

Teza:

Istnieje skojarzenie maksymalne, w którym p jest wolne.

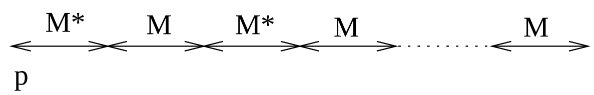
Dowód:

M* - skojarzenie maksymalne.

Załóżmy, że p nie jest wolne w M* (jakby było to mamy to co chcemy).

Rozważmy $M^* \oplus M$.

Wówczas p nie należy do żadnej ścieżki lub należy do ścieżki D o postaci:



Krawędź wychodząca z p to M*, bo p jest pokryte w M*(z założenia).

Wierzchołek p nie może wypaść w środku D, bo byłby pokryty przez krawędź M.

Ścieżka D musi się kończyć krawędzią należącą do M, bo inaczej w p by się zaczynała ścieżka rozszerzająca dla M.

Czyli jeżeli zamienimy w D wszystkie krawędzie M na M*i odwrotnie to otrzymamy ścieżkę o tej samej długości (czyli skojarzenie o tym samym rozmiarze), w którym p nie jest pokryty.

9.3 Algorytm generyczny znajdowania maksymalnego skojarzenia

M := $\emptyset$;

while *istnieje ścieżka rozszerzająca względem M* do

{

  znajdź ścieżkę rozszerzającą P;

  M := $M \oplus P$;

}

9.4 Naiwne znajdowanie ścieżek

Budujemy drzewo najkrótszych ścieżek rozszerzających (DFS lub BFS) - zaczynamy w wierzchołkach wolnych i idziemy na przemian krawędziami nie należącymi i należącymi do M, aż znajdziemy się w wierzchołku wolnym.

Czas: O(n(n+m))

9.5 Algorytm Hopcrofta

Znajdujemy maksymalny zbiór P najkrótszych ścieżek rozszerzających - z każdego napotkanego wierzchołka wolnego próbujemy wyprowadzić kolejną ścieżkę rozszerzającą.

$M = M \oplus \bigcup_{p \in P} p$

Faz jest co najwyżej $\sqrt{n}$, czyli czas O($\sqrt{n}$(n+m))

9.6 Twierdzenie (o liczbie faz algorytmu Hopcrofta)

Teza:

Liczba faz w algorytmie Hopcrofta jest $\leq \sqrt{n}$.

Dowód:

9.6.1 Lemat 1(?)

Założenia:

M - skojarzenie

p - najkrótsza ścieżka rozszerzająca względem M

q - ścieżka rozszerzająca względem $M \oplus p$

Teza:

$|q| \geq |p| + 2|p \cap q|$

Dowód:

Jeżeli p jest rozłączne z q to OK.

Wpp.

$|p \cup q| = |p \oplus q| + 2|p \cap q|$

Ponieważ p jest najkrótszą ścieżką rozszerzającą, więc $|p \oplus q| \geq 2p$. Stąd

$|p| + |q| \geq |p \cup q| \geq 2|p| + 2|p \cap q|$, więc

$|q| \geq |p| + 2|p \cap q|$

9.6.2 Lemat 2

Teza:

W każdej kolejnej fazie długość najkrótszej ścieżki rozszerzającej jest co najmniej o 2 większa.

Dowód:

Ścieżka rozszerzająca ma długość nieparzystą. W każdej fazie znajdujemy wszystkie najkrótsze ścieżki rozszerzające.

9.6.3 Dowód

Rozważmy M po pierwszych $\frac{1}{2}\sqrt{n}$ fazach.

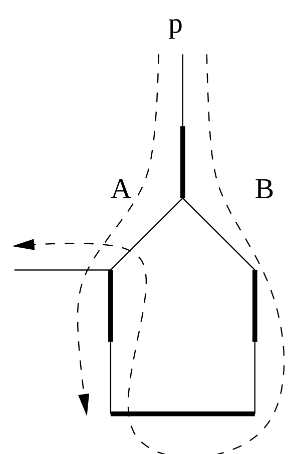
Długość najkrótszej ścieżki rozszerzającej w M jest równa co najmniej $\sqrt{n}$.

Wobec tego może być takich rozłącznych ścieżek co najwyżej $\sqrt{n}$. KL

Stąd wynika, że potrzeba jeszcze co najwyżej drugie tyle faz.

9.7 Algorytm dla grafów ogólnych

9.7.1 Dlaczego algorytm Hopcrofta nie działa



Kielich

W grafach ogólnych mogą się pojawić kielichy (cykle naprzemienne o nieparzystej długości).

W takim kielichu jeżeli przejdziemy (np. BFS-em) ścieżką A, to nie znajdziemy ścieżki B.

9.7.2 Algorytm

W każdej fazie bierzemy wolny wierzchołek p i sprawdzamy, czy istnieje ścieżka rozszerzająca o początku w p (DFS-em lub BFS-em).

Jeśli nie istnieje, to usuwamy p z grafu (na podstawie twierdzenia istnieje skojarzenie maksymalne w którym p jest wolny).

Jeśli istnieje, to poprawiamy skojarzenie.

Z kielichami radzimy sobie w następujący sposób:

Szukając ścieżki oznaczamy wierzchołki etykietami O (jeśli jest w nieparzystej odległości od korzenia) lub E (jeśli w parzystej). Jeśli wychodząc z wierzchołka E przechodzimy do wierzchołka E, to znaczy że napotkaliśmy kielich i wtedy dokonujemy kontrakcji kielicha do jednego wierzchołka parzystego.

Liczba faz: $O(n)$

Każda faza kosztuje $O(n^2)$

Koszt: $O(n^3)$