# Znajdowanie skojarzeń na maszynie równoległej

Kamil Lenartowicz

11 grudnia 2008

# Spis treści

# Skojarzenia w drzewach

**Fakt**

*Wybierając krawędź do skojarzenia w drzewie zawsze warto jest wybrać krawędź połączoną z liściem.*

# Skojarzenia w drzewach

**Fakt**

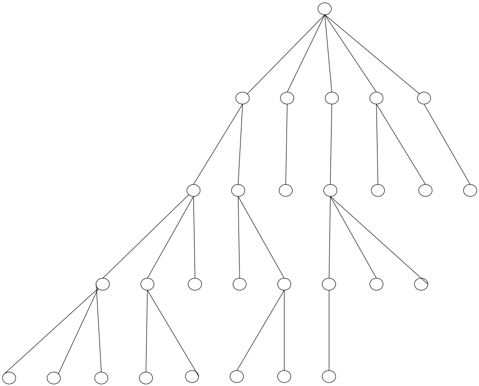
*Wybierając krawędź do skojarzenia w drzewie zawsze warto jest wybrać krawędź połączoną z liściem.*

Funkcja: Wszystkim liściom przypisz $value(v) := 0$, dla pozostałych:

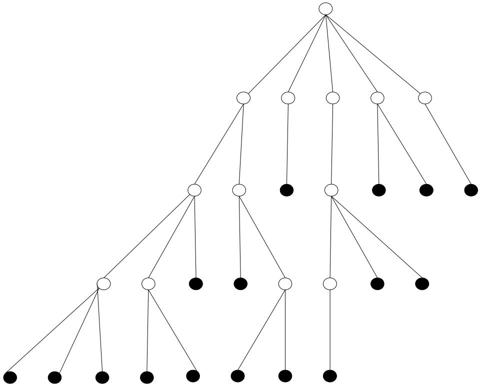
$$value(v) = NAND(x_1, ..., x_k)$$

gdzie $x_i$ to dziecko $v$, $NAND(x_1, ..., x_k) = \neg(x_1 \wedge ... \wedge x_k)$.
Wartość 1 oznacza więc, że wierzchołek zostanie skojarzony z którymś ze swoich dzieci (posiada albo liść jako dziecko albo wierzchołek który nie skojarzy się z żadnym ze swoich dzieci).

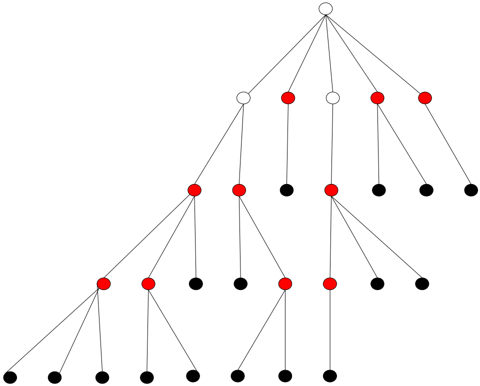
# Skojarzenia w drzewach: algorytm



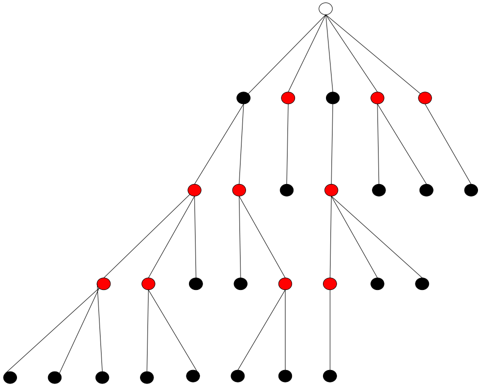
# Skojarzenia w drzewach: algorytm



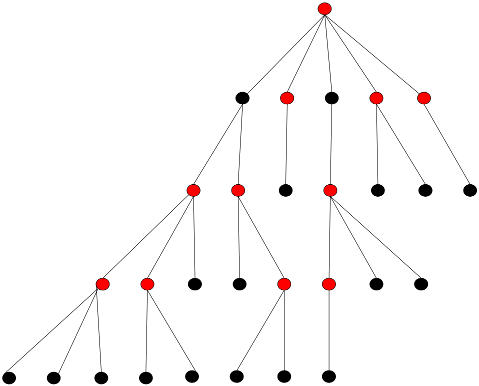
# Skojarzenia w drzewach: algorytm



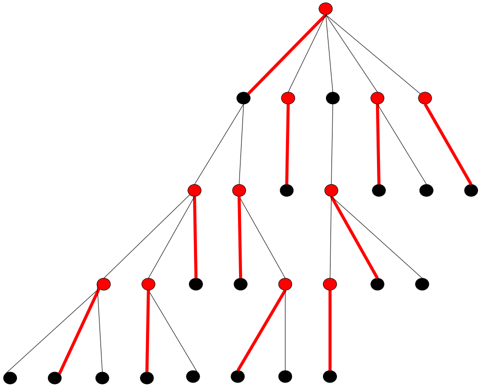
# Skojarzenia w drzewach: algorytm



# Skojarzenia w drzewach: algorytm



# Skojarzenia w drzewach: algorytm



# Skojarzenia w drzewach: algorytm

for each leaf $v$ do in parallel
$value(v) := 0;$
dla każdego wierzchołka wewnętrznego wykonaj *NAND*
używając tree-constraction;
for each $v$: $value(v) = 1$ do in parallel
wybierz dziecko $w$: $value(w) = 0;$
dodaj $(v, w)$ do skojarzenia;

# Skojarzenia w drzewach: algorytm

for each leaf $v$ do in parallel
$\quad value(v) := 0;$
dla każdego wierzchołka wewnętrznego wykonaj *NAND*
używając tree-constraction;
for each $v$: $value(v) = 1$ do in parallel
$\quad$ wybierz dziecko $w$: $value(w) = 0$;
$\quad$ dodaj $(v, w)$ do skojarzenia;

**Twierdzenie**

*Najliczniejsze skojarzenie w drzewie może być wyliczone w czasie $O(log(n))$ przy użyciu $O(n/log(n))$ procesorów na maszynie EREW PRAM.*

# Grafy gęste

## Definicja

*Graf nazywamy gęstym jeżeli stopień każdego wierzchołka to minimum $n/2$.*

# Grafy gęste

**Definicja**

*Graf nazywamy gęstym jeżeli stopień każdego wierzchołka to minimum $n/2$.*

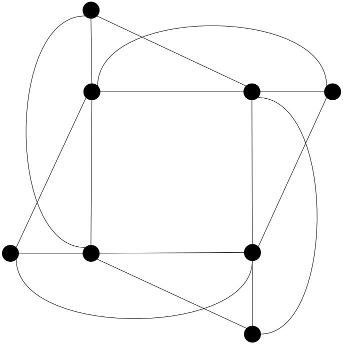
**Lemat**

*W grafach gęstych dowolne maksymalne skojarzenie pokrywa co najmiej $n/2$ wierzchołków.*

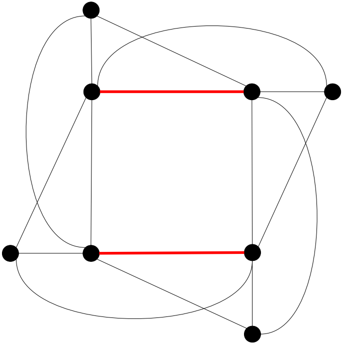
# Algorytm: grafy gęste o parzystej liczbie wierzchołków

- znajdź maksymalne skojarzenie w $G$
- stwórz ścieżki powiększające długości 3 pomiędzy nieskojarzonymi wierzchołkami zawierające po jednej krawędzi skojarzonej
- znajdź skojarzenie doskonałe pomiędzy możliwymi ścieżkami a krawędziami skojarzenia z $G$
- wykorzystaj wybrane ścieżki osiągając skojarzenie doskonałe

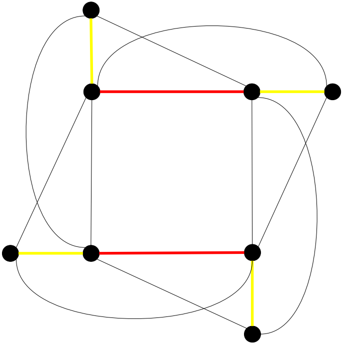
# Algorytm: grafy gęste o parzystej liczbie wierzchołków



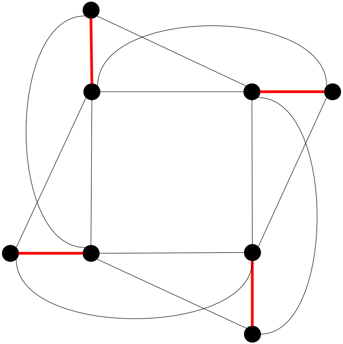
# Algorytm: grafy gęste o parzystej liczbie wierzchołków



# Algorytm: grafy gęste o parzystej liczbie wierzchołków



# Algorytm: grafy gęste o parzystej liczbie wierzchołków



# Grafy gęste

**Fakt**

*Dla każdej pary wierzchołków wolnych istnieje minimum $k$ możliwych ścieżek powiększających gdzie $k$ to liczba par wierzchołków wolnych.*

# Grafy gęste

**Fakt**

*Dla każdej pary wierzchołków wolnych istnieje minimum k możliwych ścieżek powiększających gdzie k to liczba par wierzchołków wolnych.*

**Fakt**

*Mając maszynę z Common Read można wyznaczyć ścieżki powiększające w czasie* $O(log(n))$.

# Grafy gęste

**Fakt**

*Dla każdej pary wierzchołków wolnych istnieje minimum k możliwych ścieżek powiększających gdzie k to liczba par wierzchołków wolnych.*

**Fakt**

*Mając maszynę z Common Read można wyznaczyć ścieżki powiększające w czasie* $O(log(n))$.

**Twierdzenie**

*W grafie gęstym o parzystej liczbie wierzchołków skojarzenie doskonałe można wyznaczyć w czasie* $O(log^4(n))$ *używając* $O(m)$ *procesorów na maszynie CREW PRAM.*

# Grafy regularne dwudzielne

**Fakt**

*Mając kolorowanie krawędziowe grafu regularnego z użyciem $\Delta(G)$ kolorów gdzie $\Delta(G)$ to stopień wierzchołków w G można łatwo znaleźć skojarzenie doskonałe - wystarczy wziąć krawędzie pokolorowane na jeden wybrany kolor.*

# Grafy regularne dwudzielne

## Fakt

*Mając kolorowanie krawędziowe grafu regularnego z użyciem $\Delta(G)$ kolorów gdzie $\Delta(G)$ to stopień wierzchołków w $G$ można łatwo znaleźć skojarzenie doskonałe - wystarczy wziąć krawędzie pokolorowane na jeden wybrany kolor.*

## Lemat

*Niech $2^k$ będzie największą potęgą dwójki nie przekraczającą $\Delta > 0$. Wtedy istnieje rozkład*

$$\Delta = \delta_1 + \delta_2 + \delta_3 + \delta_4$$

*gdzie $\delta_i$ są nieujemnymi liczbami całkowitymi oraz $\delta_i + \delta_j \leq 2^k$ dla każdej pary $1 \leq i \neq j \leq 4$.*

# Grafy o stopniu $2^k$

$Easy\_Bipartite\_Color(G, X)$
    if $\Delta(G) = 1$ then pokoloruj graf 1 kolorem;
    else
        $G1 := Halve(G);$
        $G2 := G - G1;$
        podziel zbiór $X$ na równe zbiory $X1$ i $X2$;
        perform in parallel
            $Easy\_Bipartite\_Color(G1, X1);$
            $Easy\_Bipartite\_Color(G2, X2);$

# Grafy o dowolnym stopniu

$\{\lambda_i\}_{i=1..4}$ - podział zbioru kolorów spełniający podział liczby z lematu ($|\lambda_i| = \delta_i$)

# Grafy o dowolnym stopniu

$\{\lambda_i\}_{i=1..4}$ - podział zbioru kolorów spełniający podział liczby z lematu ($|\lambda_i| = \delta_i$)
$\Phi_{ij} = \lambda_i \cup \lambda_j \cup X$ - zbiór $X$ - dowolne dodatkowe kolory tak aby $|\Phi_{ij}|$ było potęgą dwójki

# Grafy o dowolnym stopniu

$\{\lambda_i\}_{i=1..4}$ - podział zbioru kolorów spełniający podział liczby z lematu $(|\lambda_i| = \delta_i)$

$\Phi_{ij} = \lambda_i \cup \lambda_j \cup X$ - zbiór $X$ - dowolne dodatkowe kolory tak aby $|\Phi_{ij}|$ było potęgą dwójki

$Locally\_Valid\_Coloring(G, \Phi)$ - kolorowanie niepokolorowanych krawędzi w $G$ tak aby z każdego wierzchołka wychodziły krawędzie o różnych kolorach - jednak krawędzie mogą mieć różne kolory na końcach

# Grafy o dowolnym stopniu

$\{\lambda_i\}_{i=1..4}$ - podział zbioru kolorów spełniający podział liczby z lematu ($|\lambda_i| = \delta_i$)
$\Phi_{ij} = \lambda_i \cup \lambda_j \cup X$ - zbiór $X$ - dowolne dodatkowe kolory tak aby $|\Phi_{ij}|$ było potęgą dwójki
$Locally\_Valid\_Coloring(G, \Phi)$ - kolorowanie niepokolorowanych krawędzi w $G$ tak aby z każdego wierzchołka wychodziły krawędzie o różnych kolorach - jednak krawędzie mogą mieć różne kolory na końcach
$BadEdges(G)$ - ilość krawędzi mających różne kolory na końcach w $G$

# Grafy o dowolnym stopniu

$\{\lambda_i\}_{i=1..4}$ - podział zbioru kolorów spełniający podział liczby z lematu ($|\lambda_i| = \delta_i$)
$\Phi_{ij} = \lambda_i \cup \lambda_j \cup X$ - zbiór $X$ - dowolne dodatkowe kolory tak aby $|\Phi_{ij}|$ było potęgą dwójki
$Locally\_Valid\_Coloring(G, \Phi)$ - kolorowanie niepokolorowanych krawędzi w $G$ tak aby z każdego wierzchołka wychodziły krawędzie o różnych kolorach - jednak krawędzie mogą mieć różne kolory na końcach
$BadEdges(G)$ - ilość krawędzi mających różne kolory na końcach w $G$
$SubGraph(G, \Phi)$ - podgraf $G$ w którym obydwa kolory każdej z krawędzi należą do $\Phi$

# Grafy o dowolnym stopniu

$\{\lambda_i\}_{i=1..4}$ - podział zbioru kolorów spełniający podział liczby z lematu ($|\lambda_i| = \delta_i$)
$\Phi_{ij} = \lambda_i \cup \lambda_j \cup X$ - zbiór $X$ - dowolne dodatkowe kolory tak aby $|\Phi_{ij}|$ było potęgą dwójki
$Locally\_Valid\_Coloring(G, \Phi)$ - kolorowanie niepokolorowanych krawędzi w $G$ tak aby z każdego wierzchołka wychodziły krawędzie o różnych kolorach - jednak krawędzie mogą mieć różne kolory na końcach
$BadEdges(G)$ - ilość krawędzi mających różne kolory na końcach w $G$
$SubGraph(G, \Phi)$ - podgraf $G$ w którym obydwa kolory każdej z krawędzi należą do $\Phi$

**Fakt**

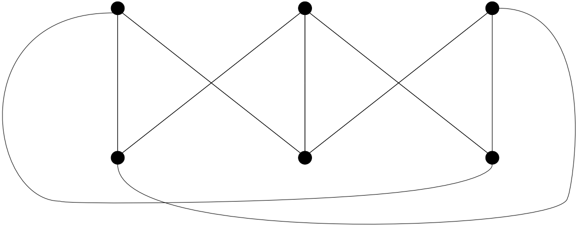
*Istnieją i, j takie że:* $BadEdges(SubGraph(G, \Phi_{ij})) \geq \frac{1}{6} BadEdges(G)$.

# Grafy o dowolnym stopniu - algorytm

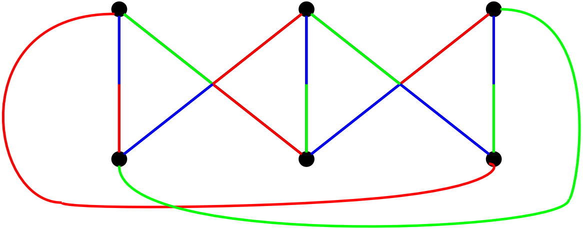
$Bipartite\_Edge\_Coloring(G, \Phi)$

- $\Phi :=$ kolorowanie puste;
- $Locally\_Valid\_Coloring(G, \Phi)$;
- while istnieje zła krawędź w G do
  - znajdź $i, j$ maksymalizujące $BadEdges(SubGraph(G, \Phi_{ij}))$;
  - $Easy\_Bipartite\_Color(SubGraph(G, \Phi_{ij}), \Phi_{ij})$;
  - usuń kolor ze złych krawędzi;
  - $Locally\_Valid\_Coloring(G, \Phi)$;

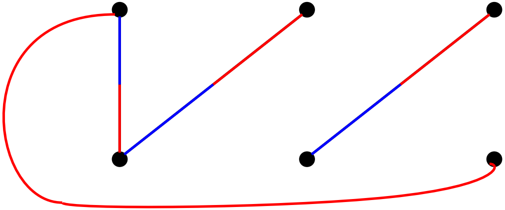
# Grafy o dowolnym stopniu - algorytm



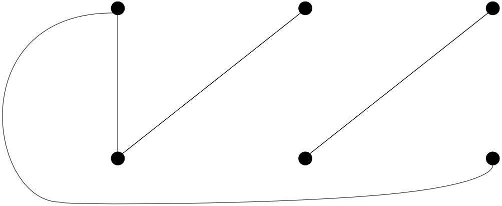
# Grafy o dowolnym stopniu - algorytm



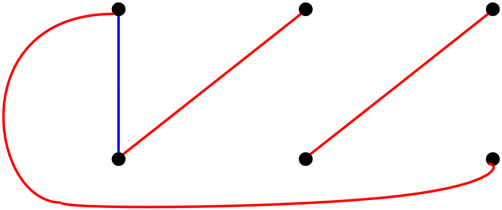
# Grafy o dowolnym stopniu - algorytm



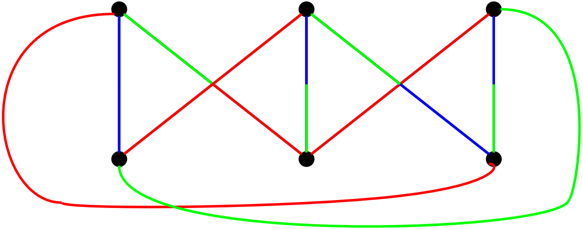
# Grafy o dowolnym stopniu - algorytm



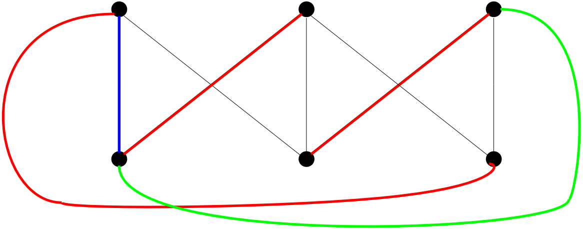
# Grafy o dowolnym stopniu - algorytm



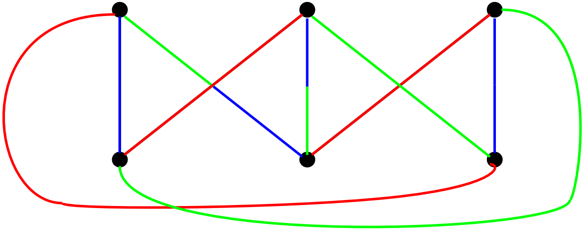
# Grafy o dowolnym stopniu - algorytm



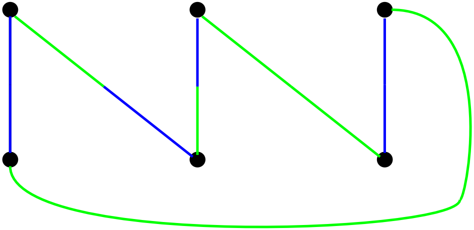
# Grafy o dowolnym stopniu - algorytm



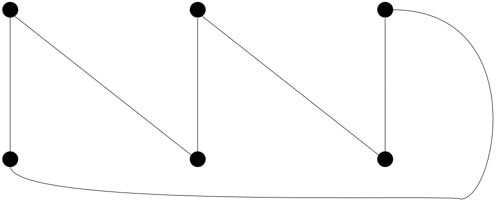
# Grafy o dowolnym stopniu - algorytm



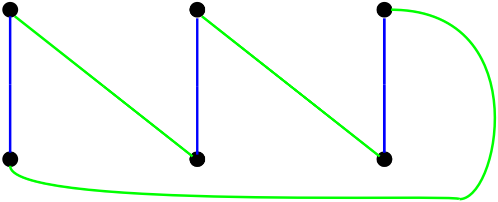
# Grafy o dowolnym stopniu - algorytm



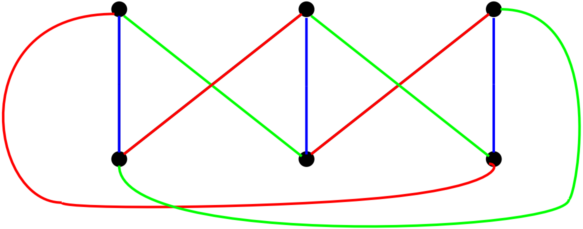
# Grafy o dowolnym stopniu - algorytm



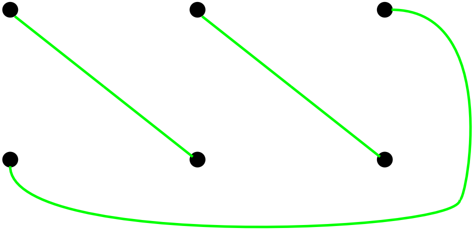
# Grafy o dowolnym stopniu - algorytm



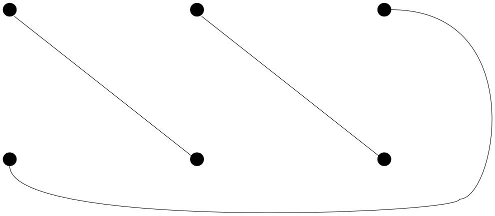
# Grafy o dowolnym stopniu - algorytm



# Grafy o dowolnym stopniu - algorytm



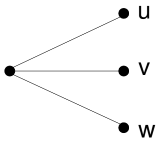
# Grafy o dowolnym stopniu - algorytm



# Claw-free graphs

## Definicja

*Claw-free graph (pl: graf bez szponów) to graf który nie zawiera indukowanego podgrafu $K_{1,3}$ (claw).*




Pomiędzy którąś parą wierzchołków $u$, $v$, $w$ musi istnieć krawędź.

# Skojarzenie pseudo-doskonałe

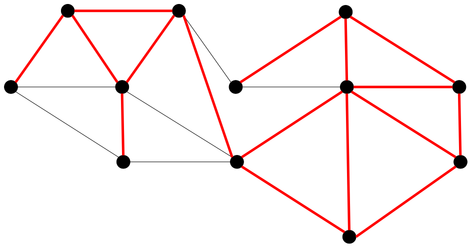
**Definicja**

*Skojarzenie pseudo-doskonałe jest grafem rozpinającym $G$ w którym każdy wierzchołek ma nieparzysty stopień.*

# Skojarzenie pseudo-doskonałe

**Definicja**

*Skojarzenie pseudo-doskonałe jest grafem rozpinającym G w którym każdy wierzchołek ma nieparzysty stopień.*



# Budowa skojarzenia pseudo-doskonałego

**Lemat**

*Istnieje algorytm należący do klasy NC konstruujący/testujący skojarzenie pseudo-doskonałe.*

# Budowa skojarzenia pseudo-doskonałego

**Lemat**

*Istnieje algorytm należący do klasy NC konstruujący/testujący skojarzenie pseudo-doskonałe.*

Układ $n$ równań z $m$ zmiennymi nad $GF(2)$ można rozwiązać w czasie $O(log^2(n))$ przy użyciu $m^{5.5}$ procesorów.

# Cykle poprawiające

## Definicja

$x \oplus y = x + y(mod2)$ - *operacja na zbiorach krawędzi* - $x, y \in \{0,1\}^m$ *reprezentują zbiory krawędzi.*

# Cykle poprawiające

**Definicja**

*$x \oplus y = x + y(mod2)$ - operacja na zbiorach krawędzi - $x, y \in \{0,1\}^m$ reprezentują zbiory krawędzi.*

Np. Jeśli $1^m$ - zbiór $E$ (wszystkie krawędzie) to $1^m \oplus 1^m = 0^m$ - zbiór pusty.

# Cykle poprawiające

**Definicja**

*$x \oplus y = x + y (mod 2)$ - operacja na zbiorach krawędzi - $x, y \in \{0, 1\}^m$ reprezentują zbiory krawędzi.*

Np. Jeśli $1^m$ - zbiór $E$ (wszystkie krawędzie) to $1^m \oplus 1^m = 0^m$ - zbiór pusty.

**Fakt**

*Jeśli $M$ to skojarzenie pseudo-doskonałe, a $C$ jest cyklem to $M \oplus C$ jest skojarzeniem pseudo-doskonałym.*

# Cykle poprawiające

## Definicja

*$x \oplus y = x + y (mod 2)$ - operacja na zbiorach krawędzi - $x, y \in \{0,1\}^m$ reprezentują zbiory krawędzi.*

Np. Jeśli $1^m$ - zbiór $E$ (wszystkie krawędzie) to $1^m \oplus 1^m = 0^m$ - zbiór pusty.

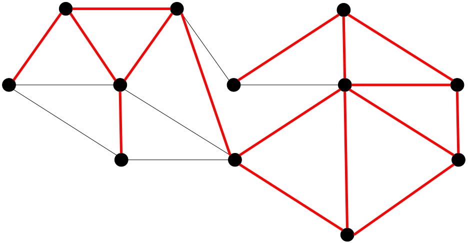
## Fakt

*Jeśli $M$ to skojarzenie pseudo-doskonałe, a $C$ jest cyklem to $M \oplus C$ jest skojarzeniem pseudo-doskonałym.*

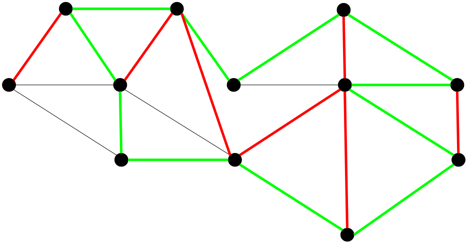
## Definicja

*Dodatkowo jeśli $M \oplus C$ ma mniej krawędzi niż $M$ to $C$ nazywamy cyklem poprawiającym (augmenting cycle).*

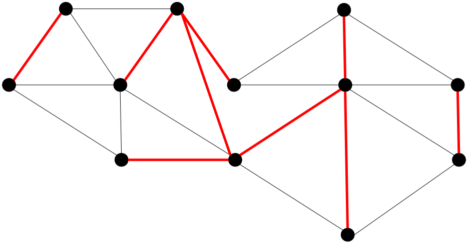
# Cykle poprawiające



# Cykle poprawiające



# Cykle poprawiające



# Skojarzenie pseudo-doskonałe

**Twierdzenie**

*Niech G graf claw-free o parzystej liczbie wierzchołków, wtedy każda spójna składowa w G posiada skojarzenie doskonałe.*

# Skojarzenie pseudo-doskonałe

**Twierdzenie**

*Niech $G$ graf claw-free o parzystej liczbie wierzchołków, wtedy każda spójna składowa w $G$ posiada skojarzenie doskonałe.*

**Obserwacja**

*Niech $M$ skojarzenie pseudo-doskonałe grafu claw-free $G$, jeśli $M$ jest lasem indukowanym to $M$ jest skojarzeniem doskonałym.*

# Skojarzenie pseudo-doskonałe

**Twierdzenie**

*Niech $G$ graf claw-free o parzystej liczbie wierzchołków, wtedy każda spójna składowa w $G$ posiada skojarzenie doskonałe.*

**Obserwacja**

*Niech $M$ skojarzenie pseudo-doskonałe grafu claw-free $G$, jeśli $M$ jest lasem indukowanym to $M$ jest skojarzeniem doskonałym.*

Algorithm Perfect_Matching

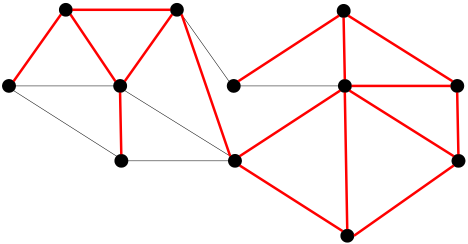
skonstruuj skojarzenie pseudo-doskonałe $M$;
konwertuj $M$ do skojarzenia pseudo-doskonałego $F$ które jest lasem;
konwertuj $F$ do skojarzenia pseudo-doskonałego $F'$ które jest lasem indukowanym;

# Cykle fundamentalne

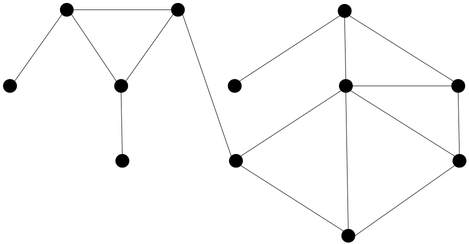
**Definicja**

*Niech $T$ będzie drzewem rozpinającym grafu $M$. Cyklem fundamentalnym ze względu na $T$ nazywamy cykl powstały przez dodanie jednej krawędzi z $M$ do $T$.*

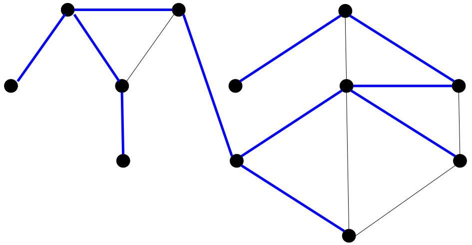
# Cykle fundamentalne



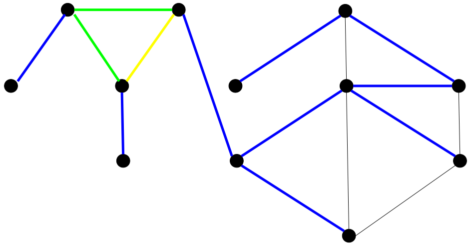
# Cykle fundamentalne



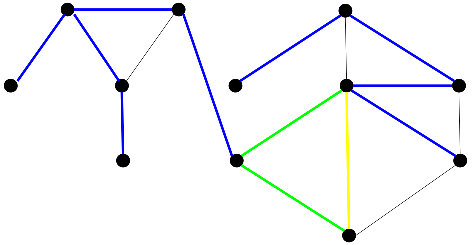
# Cykle fundamentalne



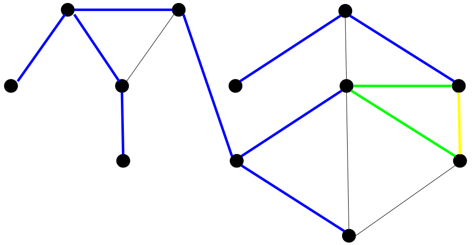
# Cykle fundamentalne



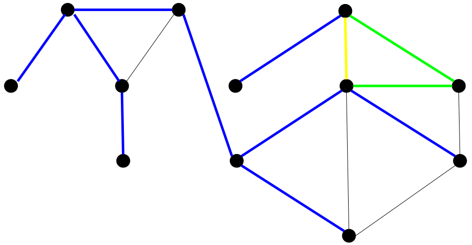
# Cykle fundamentalne



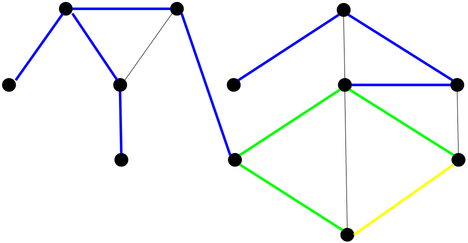
# Cykle fundamentalne



# Cykle fundamentalne



# Cykle fundamentalne



# Algorytm: konwersja $M$ do lasu $F$

**Fakt**

*Niech $M$ skojarzenie pseudo-doskonałe, $T$ drzewo rozpinające $M$ a $C_1, \ldots, C_r$ zbiór cykli fundamentalnych ze względu na $T$, wtedy $M \oplus C_1 \oplus \ldots \oplus C_r$ jest skojarzeniem pseudo-doskonałym które jest lasem.*

# Algorytm: konwersja $M$ do lasu $F$

**Definicja**

*$P(e, T, M)$ - ilość cykli fundamentalnych w $M$ ze względu na $T$ zawierających $e$ modulo 2.*

# Algorytm: konwersja $M$ do lasu $F$

## Definicja

*$P(e,T,M)$ - ilość cykli fundamentalnych w $M$ ze względu na $T$ zawierających $e$ modulo 2.*

## Fakt

*Niech $e=(u,v)$ gdzie $v$ jest dzieckiem $u$. Wtedy $P(e,T,M)$ jest sumą modulo 2 wartości $p(w)$ po wszystkich wierzchołkach $w$ poddrzewa o korzeniu w $v$, gdzie $p(w)$ oznacza ilość niedrzewowych krawędzi z $M$ wychodzących z $w$.*

# Algorytm: konwersja $M$ do lasu $F$

**Definicja**

$P(e, T, M)$ - *ilość cykli fundamentalnych w $M$ ze względu na $T$ zawierających $e$ modulo 2.*

**Fakt**

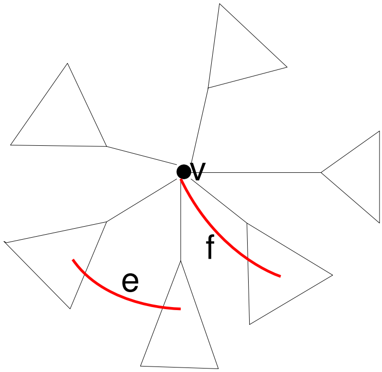
*Niech $e = (u, v)$ gdzie $v$ jest dzieckiem $u$. Wtedy $P(e, T, M)$ jest sumą modulo 2 wartości $p(w)$ po wszystkich wierzchołkach $w$ poddrzewa o korzeniu $w$ $v$, gdzie $p(w)$ oznacza ilość niedrzewowych krawędzi z $M$ wychodzących z $w$.*

Czyli: $F = \{e : e \in T \wedge P(e, T, M) = 0\}$

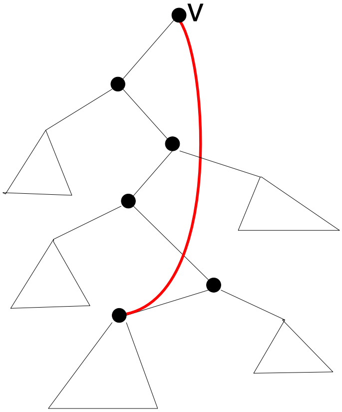
# Algorytm: konwersja lasu $F$ do lasu indukowanego $F'$

- Podejście dziel i rządź - szukamy wierzchołka który dzieli drzewo na komponenty maksimum $\frac{2}{3}$ wielkości skojarzenia.
- W każdej fazie zmniejszamy rozmiar największego złego komponentu o współczynnik maksymalnie $\frac{2}{3}$.

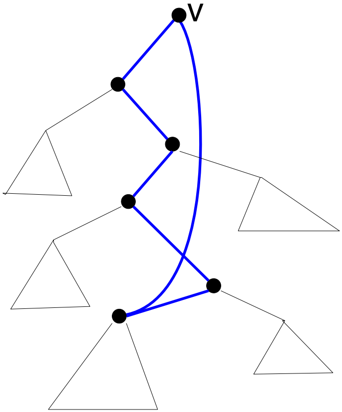
# Algorytm: konwersja drzewa $F$ do drzewa indukowanego $F'$



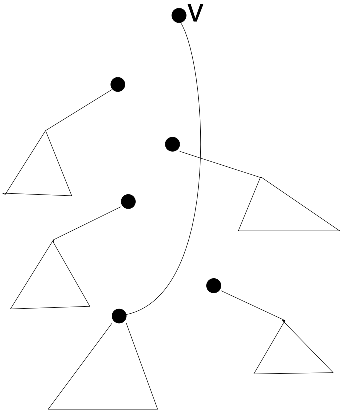
# Algorytm: konwersja drzewa $F$ do drzewa indukowanego $F'$



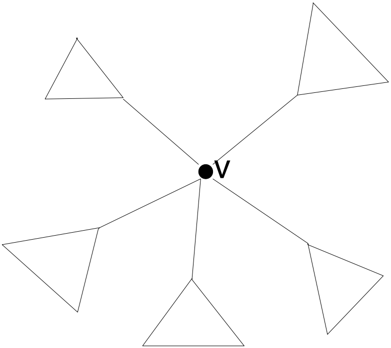
# Algorytm: konwersja drzewa $F$ do drzewa indukowanego $F'$



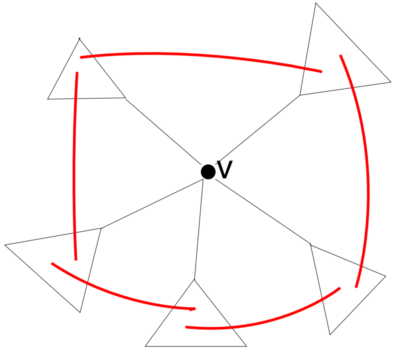
# Algorytm: konwersja drzewa $F$ do drzewa indukowanego $F'$



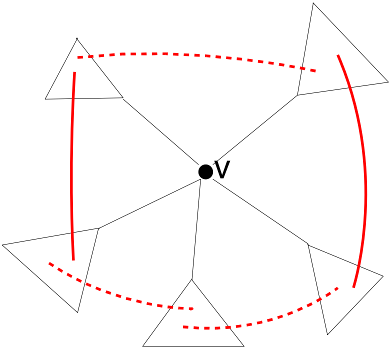
# Algorytm: konwersja drzewa $F$ do drzewa indukowanego $F'$



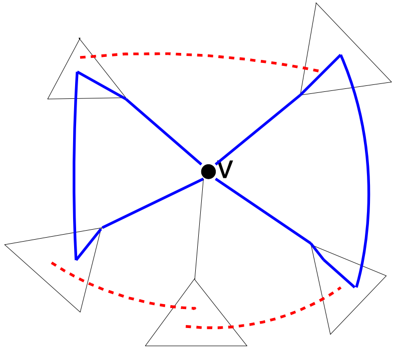
# Algorytm: konwersja drzewa $F$ do drzewa indukowanego $F'$



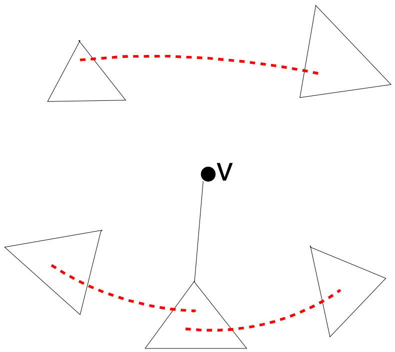
# Algorytm: konwersja drzewa $F$ do drzewa indukowanego $F'$



# Algorytm: konwersja drzewa $F$ do drzewa indukowanego $F'$



# Algorytm: konwersja drzewa $F$ do drzewa indukowanego $F'$



# Bibliografia


- Marek Karpinski, Wojciech Rytter (1998) “Fast Parallel Algorithms for Graph Matching Problems: Combinatorial, Algebraic, and Probabilistic Approach”, rozdziały 1,7,9