Determining 4-edge-connected components in linear time

Mateusz Pach, Michał Wronka

Wojciech Nadara, Mateusz Radecki, Marcin Smulewicz, Marek Sokołowski
Uniwersytet Warszawski

4 listopada 2021

Wprowadzenie

k-spójność krawędziowa

Dla $k > 1$ graf jest *k-spójny krawędziowo* jeśli pozostaje spójny po usunięciu z niego dowolnego zbioru co najwyżej $k - 1$ krawędzi.

Wprowadzenie

k-spójność krawędziowa

Dla $k > 1$ graf jest *k-spójny krawędziowo* jeśli pozostaje spójny po usunięciu z niego dowolnego zbioru co najwyżej $k - 1$ krawędzi.

k-osiągalność krawędziowa

Dla $k > 1$ oraz pary wierzchołków u, v grafu G mówimy, że u jest *k-osiągalny krawędziowo* z v jeśli po usunięciu z G dowolnego zbioru co najwyżej $k - 1$ krawędzi wierzchołki u i v pozostają w tej samej spójnej składowej.

Wprowadzenie

k-spójność krawędziowa

Dla $k > 1$ graf jest *k-spójny krawędziowo* jeśli pozostaje spójny po usunięciu z niego dowolnego zbioru co najwyżej $k - 1$ krawędzi.

k-osiągalność krawędziowa

Dla $k > 1$ oraz pary wierzchołków u, v grafu G mówimy, że u jest *k-osiągalny krawędziowo* z v jeśli po usunięciu z G dowolnego zbioru co najwyżej $k - 1$ krawędzi wierzchołki u i v pozostają w tej samej spójnej składowej.

k-spójna krawędziowo składowa

K-spójna krawędziowo składowa to element klasy równoważności na relacji k-osiągalności krawędziowej.

Dotychczasowe wyniki

Dotychczasowe wyniki

- $k = 1 \to$ dfs

Dotychczasowe wyniki

- $k = 1 \rightarrow$ dfs
- $k = 2 \rightarrow$ funkcja low (Hopcroft i Tarjan)

Dotychczasowe wyniki

- $k = 1 \rightarrow$ dfs
- $k = 2 \rightarrow$ funkcja low (Hopcroft i Tarjan)
- $k = 3 \rightarrow$ 3-spójność wierzchołkowa (Hopcroft i Tarjan)
  redukcja k-spójności krawędziowej do wierzchołkowej (Galil i Italiano)

Dotychczasowe wyniki

- $k = 1 \rightarrow$ dfs
- $k = 2 \rightarrow$ funkcja low (Hopcroft i Tarjan)
- $k = 3 \rightarrow$ 3-spójność wierzchołkowa (Hopcroft i Tarjan)
  redukcja k-spójności krawędziowej do wierzchołkowej (Galil i Italiano)
- $k = 4 \rightarrow$ utrzymywanie spójnych w grafach inkrementacyjnych
  $O(m + n\alpha(n))$ (Kanevsky et al.)
  $O(q + m + nlogn)$ (Dinitz i Westbrook)

Dotychczasowe wyniki

- $k = 1 \to$ dfs
- $k = 2 \to$ funkcja low (Hopcroft i Tarjan)
- $k = 3 \to$ 3-spójność wierzchołkowa (Hopcroft i Tarjan)
  redukcja k-spójności krawędziowej do wierzchołkowej (Galil i Italiano)
- $k = 4 \to$ utrzymywanie spójnych w grafach inkrementacyjnych
  $O(m + n\alpha(n))$ (Kanevsky et al.)
  $O(q + m + nlogn)$ (Dinitz i Westbrook)
- $k >= 5 \to O(q + m + k^2 nlog(n/k))$ (Dinitz i Westbrook)

Nowy wynik

Deterministyczny algorytm znajdujący 4-spójne krawędziowo składowe w czasie liniowym.

Plan

Plan

- Redukcja do problemu znalezienia 4-spójnych w grafie 3-spójnym krawędziowo.

Plan

- Redukcja do problemu znalezienia 4-spójnych w grafie 3-spójnym krawędziowo.
- Dodatkowa struktura danych i algorytm z twierdzenia 4.3.

Plan

- Redukcja do problemu znalezienia 4-spójnych w grafie 3-spójnym krawędziowo.
- Dodatkowa struktura danych i algorytm z twierdzenia 4.3.
- Randomizowane znajdywanie 3-cięć w grafie 3-spójnym.

Plan

- Redukcja do problemu znalezienia 4-spójnych w grafie 3-spójnym krawędziowo.
- Dodatkowa struktura danych i algorytm z twierdzenia 4.3.
- Randomizowane znajdywanie 3-cięć w grafie 3-spójnym.
- Deterministyczne znajdywanie 3-cięć w grafie 3-spójnym.

Plan

- Redukcja do problemu znalezienia 4-spójnych w grafie 3-spójnym krawędziowo.
- Dodatkowa struktura danych i algorytm z twierdzenia 4.3.
- Randomizowane znajdywanie 3-cięć w grafie 3-spójnym.
- Deterministyczne znajdywanie 3-cięć w grafie 3-spójnym.
- Rekonstrukcja 4-spójnych na podstawie 3-cięć.

Definicje

$\mathcal{T}(G, u)$

drzewo przeszukiwania DFS grafu G ukorzenione w u

Definicje

$\mathcal{T}(G, u)$

drzewo przeszukiwania DFS grafu G ukorzenione w u

krawędź drzewowa

krawędź G należąca do drzewa $\mathcal{T}$, narzucamy jej umowne skierowanie "w dół"(od korzenia)

Definicje

$\mathcal{T}(G, u)$

drzewo przeszukiwania DFS grafu G ukorzenione w u

krawędź drzewowa

krawędź G należąca do drzewa $\mathcal{T}$, narzucamy jej umowne skierowanie "w dół"(od korzenia)

$\mathcal{T}_e$

poddrzewo $\mathcal{T}$ ukorzenione w "głowie"(patrząc na umowne skierowanie) krawędzi $e \in E(\mathcal{T})$, analogicznie $\mathcal{T}_v : v \in V(\mathcal{T})$

Definicje

$\mathcal{T}(G, u)$

drzewo przeszukiwania DFS grafu G ukorzenione w u

krawędź drzewowa

krawędź G należąca do drzewa $\mathcal{T}$, narzucamy jej umowne skierowanie "w dół"(od korzenia)

$\mathcal{T}_e$

poddrzewo $\mathcal{T}$ ukorzenione w "głowie"(patrząc na umowne skierowanie) krawędzi $e \in E(\mathcal{T})$, analogicznie $\mathcal{T}_v : v \in V(\mathcal{T})$

krawędź wsteczna

krawędź G nie będąca drzewową, narzucamy jej umowne skierowanie "w górę"(do korzenia)

Definicje

głowa/ogon krawędzi

wierzchołek będący końcem/początkiem krawędzi w powyżej przyjętym skierowaniu

Definicje

głowa/ogon krawędzi

wierzchołek będący końcem/początkiem krawędzi w powyżej przyjętym skierowaniu

cykl fundamentalny C_e

dla $e \in E(G)$ będącej krawędzią wsteczną, łączącą wierzchołki u, v jest to cykl zbudowany z ścieżki pomiędzy u i v w $\mathcal{T}$ i e

Definicje

głowa/ogon krawędzi

wierzchołek będący końcem/początkiem krawędzi w powyżej przyjętym skierowaniu

cykl fundamentalny C_e

dla $e \in E(G)$ będącej krawędzią wsteczną, łączącą wierzchołki u, v jest to cykl zbudowany z ścieżki pomiędzy u i v w $\mathcal{T}$ i e

$\leq_{\mathcal{T}}$

porządek częściowy na drzewie, $u, v \in V(\mathcal{T})$, gdzie $u \leq_{\mathcal{T}} v \iff u$ leży na ścieżce do v z korzenia $\mathcal{T}$. Analogicznie dla krawędzi drzewowych, porównując ich "głowy".

Definicje

krawędź przeskakująca

krawędź $f = pq$ przeskakuje wierzchołek v jeśli $p \in \mathcal{T}_v \wedge q \notin \mathcal{T}_v$, analogicznie definiujemy przeskakiwanie nad krawędzią

Definicje

krawędź przeskakująca

krawędź $f = pq$ przeskakuje wierzchołek v jeśli $p \in \mathcal{T}_v \wedge q \notin \mathcal{T}_v$, analogicznie definiujemy przeskakiwanie nad krawędzią

low()

funkcja $low()$ zdefiniowana przez Tarjana, ale dla naszych potrzeb zdefiniujemy $low(e)$ jako krawędź przeskakującą e minimalizującą preorder swojej głowy. $low(e) = \perp$ jeśli takowa nie istnieje.

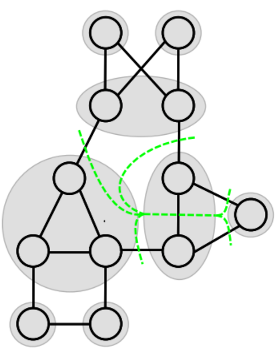
Redukcja

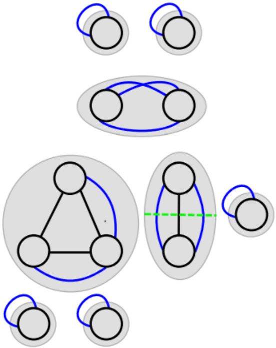
Twierdzenie 1

Dla podanej 2-spójnej grafu G, istnieje graf H = (U, F), że:

- każda krawędź $e \in F$ leży na dokładnie 1 cyklu H
- U jest rodziną wszystkich 3-spójnych w G
- istnieje bijekcja $\Phi : E' \to F$, że $E' \subset E$ jest zbiorem wszystkich krawędzi należących do pewnego 2-cięcia G, że $\forall e = uv \in E'$ jej obraz $\Phi(e)$ jest krawędzią w H łączące pewną 3-spójne zawierającą u i 3-spójną zawierającą v
- para krawędzi $e, f \in E'$ tworzy 2-cięcie $\iff$ $\Phi(e)$ i $\Phi(f)$ leżą na tym samym cyklu H.

Redukcja





Redukcja

Lemat 1.

Graf S ma następujące właściwości:

- jest 3-spójny
- dla każdego 3-cięcia c w G dzielącego X na niepuste części istnieje 3-cięcie c' w S, dzielące X w ten sam sposób
- dla każdego 3-cięcia c' w S istnieje niepusty zbiór 3-cięć w G, każde dzielące X w ten sam sposób co c'

Dodatkowa struktura danych i algorytm z twierdzenia 4.3.

Twierdzenie 2.

Istnieje struktura danych dla problemu łączenia zbiorów rozłącznych, która zainicjalizowana nieskierowanym drzewem $\mathcal{T}$ ("union tree") o n wierzchołkach, tworzy n singletonów. Po inicjalizacji struktura ta obsługuje następujące operacje w dowolnej kolejności:

- find(x): zwraca indeks zbioru zawierającego x,
- union(x, y): jeśli x i y są w różnych zbiorach to któryś z nich jest zastąpiony ich sumą, podczas gdy ten drugi jest zastąpiony zbiorem pustym. Operacja ta jest dozwolona tylko jeśli xy jest krawędzią $\mathcal{T}$.

Struktura obsługuje dowolny ciąg q operacji w łącznym czasie $O(n+q)$.

Dodatkowa struktura danych i algorytm z twierdzenia 4.3.

Lemat 2.

Strukturę z poprzedniego twierdzenia można wzbogacić tak, aby po ukorzenieniu $\mathcal{T}$ w dowolnym wierzchołku być w stanie obsłużyć poniższą operacje w czasie stałym:

- lowest(x): zwraca najmniejszy w sensie $\leq_{\mathcal{T}}$ wierzchołek z S, gdzie S to zbiór zawierający x.

Dodatkowa struktura danych i algorytm z twierdzenia 4.3.

Twierdzenie 3.

Istnieje deterministyczny algorytm przyjmujący jako wejście:

- nieskierowane, nieukorzenione n-wierzchołkowe drzewo $\mathcal{T}$,
- p ważonych ścieżek $P_1, P_2, \ldots, P_p$ w drzewie $\mathcal{T}$, gdzie ścieżka P_i $(1 \leq i \leq p)$ jest najkrótszą ścieżką pomiędzy wierzchołkami u_i, v_i o wadze $w_i \in \{0, 1, \ldots, C\}$,
- i dodatnią liczbę całkowitą k,

i dla każdej krawędzi e zwraca indeksy k najlżejszych ścieżek zawierających e, remisy rozstrzygając dowolnie; jeśli e występuje w mniej niż k ścieżkach, wszystkie takie ścieżki zostają zwrócone. Złożoność czasowa tego algorytmu wynosi $O(nk + p + C)$.

Algorytm randomizowany

funkcja haszująca

Funkcja $H : E \to \mathcal{P}(E)$ taka, że:

$$H(e) = \begin{cases} \{e\}, e \notin \mathcal{T} \\ \text{zbiór krawędzi niedrzewowych przeskakujących } e, e \in \mathcal{T} \end{cases}$$

Algorytm randomizowany

funkcja haszująca

Funkcja $H : E \to \mathcal{P}(E)$ taka, że:

$$H(e) = \begin{cases} \{e\}, e \notin \mathcal{T} \\ \text{zbiór krawędzi niedrzewowych przeskakujących } e, e \in \mathcal{T} \end{cases}$$

Lemat 3.

Niech $G = (V, E)$ to spójny graf, a A to podzbiór jego krawędzi. Graf $G' := G - A$ jest niespójny wtw. istnieje niepusty podzbiór $B \subseteq A$ taki, że xor haszy krawędzi w B jest zbiorem pustym.

Algorytm randomizowany

funkcja haszująca

Funkcja $H : E \to \mathcal{P}(E)$ taka, że:

$$H(e) = \begin{cases} \{e\}, e \notin \mathcal{T} \\ \text{zbiór krawędzi niedrzewowych przeskakujących } e, e \in \mathcal{T} \end{cases}$$

Lemat 3.

Niech $G = (V, E)$ to spójny graf, a A to podzbiór jego krawędzi. Graf $G' := G - A$ jest niespójny wtw. istnieje niepusty podzbiór $B \subseteq A$ taki, że xor haszy krawędzi w B jest zbiorem pustym.

Lemat 4.

Mając dwie krawędzie e i f należące do pewnego 3-cięcia w 3-spójnym grafie, pozostałą krawędź 3-cięcia wyznaczamy jednoznacznie po jej haszu: $H(e) \oplus H(f)$.

Algorytm deterministyczny

MaxDn()

Używamy struktury z TW 3. inicjalizując
$k = 1, C = 2n, \{P_e | e$ jest krawędzią wsteczną w $\mathcal{T}\}$ gdzie dla P_e i
$e = (x, y), y \leq_{\mathcal{T}} x$ waga $w_e = 2n - preorder(x)$.
Inaczej jest to krawędź przeskakująca e z ogonem o maksymalnym preorderze.

MinDn()

Używamy struktury z TW 3. inicjalizując
$k = 1, C = 2n, \{P_e | e$ jest krawędzią wsteczną w $\mathcal{T}\}$ gdzie dla P_e i
$e = (x, y), y \leq_{\mathcal{T}} x$ waga $w_e = preorder(x)$.
Inaczej jest to krawędź przeskakująca e z ogonem o minimalnym preorderze.

Algorytm deterministyczny

MaxUp()

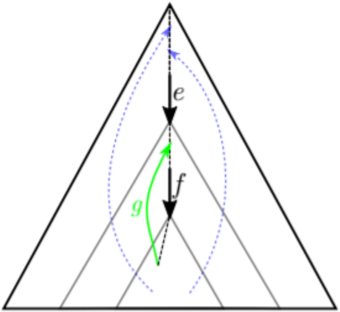
Używamy struktury z TW 3. inicjalizując $k = 1, C = 2n, \{P_e | e$ jest krawędzią wsteczną w $\mathcal{T}\}$ gdzie dla P_e i $e = (x, y), y \leq_{\mathcal{T}} x$ waga $w_e = 2n - preorder(y)$.
Inaczej jest to krawędź przeskakująca e z głową o maksymalnym preorderze.

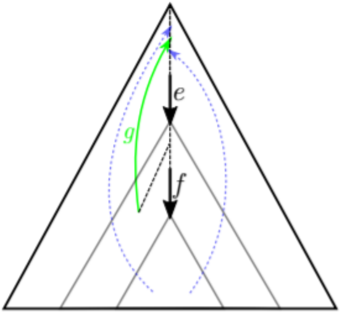
Analogicznie do $MaxUp()$, $MinDn()$, $MaxDn()$ definiujemy $MaxUp1()$, $MinDn1()$, $MaxDn1()$, $MaxUp2()$, $MinDn2()$, $MaxDn2()$.
Podobnie definiujemy $low1()$, $low2()$, $low3()$, jako 3 krawędzie przeskakujące pewne $e \in \mathcal{T}$ o najmniejszych wartościach preorder swoich głów, w kolejności rosnącej.

Algorytm deterministyczny – 1 krawędź drzewowa

Zestaw $\{e, low1(e), low2(e)\}$ jest 3-cięciem G $\iff$ $low3(e) = \perp$

Algorytm deterministyczny – 2 krawędzie drzewowe





Lemat 5.

Jeśli $e, f \in \mathcal{T}$ należą do 3-cięcia, to są porównywalne względem $\leq_{\mathcal{T}}$

Algorytm deterministyczny – 2 krawędzie drzewowe

lower case

DeepestDnCut(e): $e \in E(\mathcal{T})$

Najgłębsza $f \in E(\mathcal{T})$, że wszystkie krawędzie przeskakujące e mają ogony w $\mathcal{T}_f$. Jest poprawnie zdefiniowana, bo jej głowa jest LCA dla rozpatrywanych ogonów.

Algorytm deterministyczny – 2 krawędzie drzewowe

lower case

DeepestDnCut(e): $e \in E(\mathcal{T})$

Najgłębsza $f \in E(\mathcal{T})$, że wszystkie krawędzie przeskakujące e mają ogony w $\mathcal{T}_f$. Jest poprawnie zdefiniowana, bo jej głowa jest LCA dla rozpatrywanych ogonów.

Lemat 6.

$\forall_{e \in E(\mathcal{T})} DDC(e) = uv, u \leq_{\mathcal{T}} v$ to $v = LCA(MinDn1(e), MaxDn1(e))$

DeepestDnCut(e): $e \in E(\mathcal{T})$

Najgłębsza $f \in E(\mathcal{T})$, że wszystkie krawędzie przeskakujące e mają ogony w $\mathcal{T}_f$. Jest poprawnie zdefiniowana, bo jej głowa jest LCA dla rozpatrywanych ogonów.

Lemat 6.

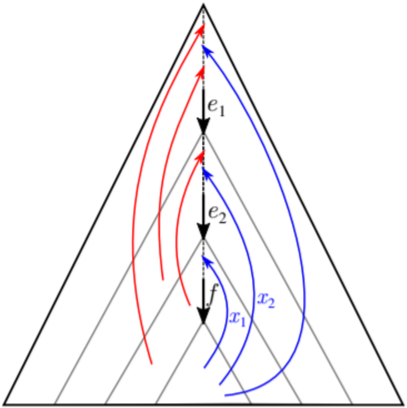
$\forall_{e \in E(\mathcal{T})} DDC(e) = uv, u \leq_{\mathcal{T}} v$ to $v = LCA(MinDn1(e), MaxDn1(e))$

Lemat 7.

Jeśli $\{e, f, g\}$ to 3-cięcie i $e, f \in \mathcal{T}, f >_{\mathcal{T}} e$ to e jest najgłębszą krawędzią spełniającą $DDC(e) \geq_{\mathcal{T}} f$

Algorytm deterministyczny – 2 krawędzie drzewowe

lower case



Algorytm deterministyczny – 2 krawędzie drzewowe

upper case

Niech g ma ogon v, zauważmy, że $\forall u \in V(\mathcal{T}_f)$ $preorder(v) < preorder(u)$, albo $\forall u \in V(\mathcal{T}_f)$ $preorder(v) > preorder(u)$.

Algorytm deterministyczny – 2 krawędzie drzewowe

upper case

Niech g ma ogon v, zauważmy, że $\forall u \in V(\mathcal{T}_f)$ $preorder(v) < preorder(u)$, albo $\forall u \in V(\mathcal{T}_f)$ $preorder(v) > preorder(u)$.

DeepsetDnCutNoMin(e)

najgłębsza krawędź $f \in \mathcal{T}$, że wszystkie krawędzie przeskakujące e, poza $g = MinDn1(e)$ mają ogon w $\mathcal{T}_f$.

Niech g ma ogon v, zauważmy, że $\forall u \in V(\mathcal{T}_f)$ $preorder(v) < preorder(u)$, albo $\forall u \in V(\mathcal{T}_f)$ $preorder(v) > preorder(u)$.

DeepsetDnCutNoMin(e)

najgłębsza krawędź $f \in \mathcal{T}$, że wszystkie krawędzie przeskakujące e, poza $g = MinDn1(e)$ mają ogon w $\mathcal{T}_f$.

Analogicznie do lematu 6. $DDCNM(e) = LCA(MinDn2(e), MaxDn1(e))$ (symetrycznie gdy $g = MaxDn1(e)$).
Podobnie dla $f_0 = DDCNM(e)$ jeśli pewne $f > e$ jest w 3-cięciu $\{e, f, g\}$ to $f \leq f_0$

Algorytm deterministyczny – 2 krawędzie drzewowe

upper case

Lemat 8.

$\{f_i, g, e\}$ jest 3-cięciem wtw $MaxUp1(f_i) = h_i <_{\mathcal{T}} e$. Jeśli nie jest, to $f_{i+1} \in \mathcal{T}$ takie, że $head(h_i) = head(f_{i+1})$ będzie nowym lepszym kandydatem.
$f_0 = DDCNM(e)$

Algorytm deterministyczny – 2 krawędzie drzewowe

upper case

Lemat 8.

$\{f_i, g, e\}$ jest 3-cięciem wtw $MaxUp1(f_i) = h_i <_{\mathcal{T}} e$. Jeśli nie jest, to $f_{i+1} \in \mathcal{T}$ takie, że $head(h_i) = head(f_{i+1})$ będzie nowym lepszym kandydatem.
$f_0 = DDCNM(e)$

Drzewo U

Drzewo ukorzenione w $\perp$, że $V(U) = E(\mathcal{T}) \bigcup \{\perp\}$ w którym będzie zachodzić, że dla $\perp \neq e \in V(U)$ jego rodzicem będzie krawędź z $\mathcal{T}$ o tej samej głowie co $MaxUp1(e)$ w G, a jeśli głowa byłaby korzeniem, to rodzic e w U to $\perp$. U można zbudować w czasie linowym.

Niech F_U to niepołączona struktura Union z Lematu 3. na drzewie U

Algorytm deterministyczny – 2 krawędzie drzewowe

upper case

Algorithm 3. Efficient deterministic solution for "two tree edges, upper case"

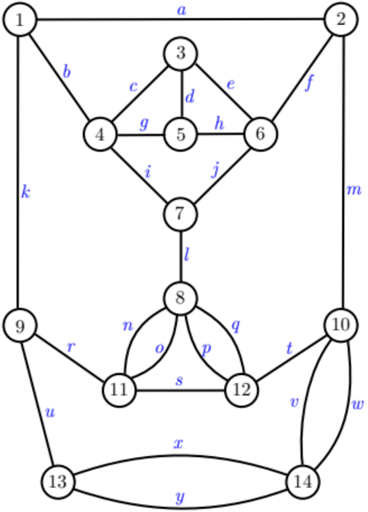
```
function SolveTwoTreeEdgesUpper
    𝒰 ← the rooted tree on E(𝒯) ∪ {⊥} defined above
    F_𝒰 ← the disjoint set union data structure built on 𝒰 (Lemma 4.2)
    for e ∈ E(𝒯), in order from the deepest to the shallowest do
        for c – child of e in 𝒰 do
            F_𝒰.union(e, c)
        g ← MinDn1(e)
        f_0 ← DeepestDnCutNoMin(e)
        f' ← F_𝒰.lowest(f_0)
        if f' ≠ e then
            add {e, f', g} to the list of 3-edge-cuts
        ▷ Run the analogous algorithm for DeepestDnCutNoMax instead of DeepestDnCutNoMin.
```

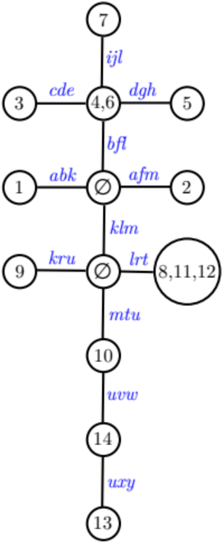
Rekonstrukcja 4-spójnych krawędziowo składowych

Twierdzenie 4.

Dla każdego 3-spójnego krawędziowo grafu $G = (V, E)$, istnieje drzewo $H = (U, F)$ wraz z funkcjami $\phi : C \to F$ i $\psi : V \to U$, takimi że ϕ jest bijekcją z 3-cięć G do krawędzi H, a ψ mapuje (niekoniecznie surjektywnie) wierzchołki z G na wierzchołki z H tak że, całe 4-spójne krawędziowo składowe są zmapowane na ten sam wierzchołek. Ponadto, jeśli 3-cięcie c dzieli wierzchołki G na dwa zbiory V_1 i V_2, to $\phi(c)$ dzieli wierzchołki H na dwa zbiory U_1 i U_2 tak, że $\psi^{-1}(U_1) = V_1$ i $\psi^{-1}(U_2) = V_2$.

Rekonstrukcja 4-spójnych krawędziowo składowych





Rekonstrukcja 4-spójnych krawędziowo składowych

Lemat 9.

Istnieje algorytm, który dla grafu G i listy C wszystkich 3-cięć w G konstruuje drzewo H wraz z funkcjami ϕ i ψ i działa w czasie liniowym.