Odległość edycyjna na drzewach

Maciej Brzeski

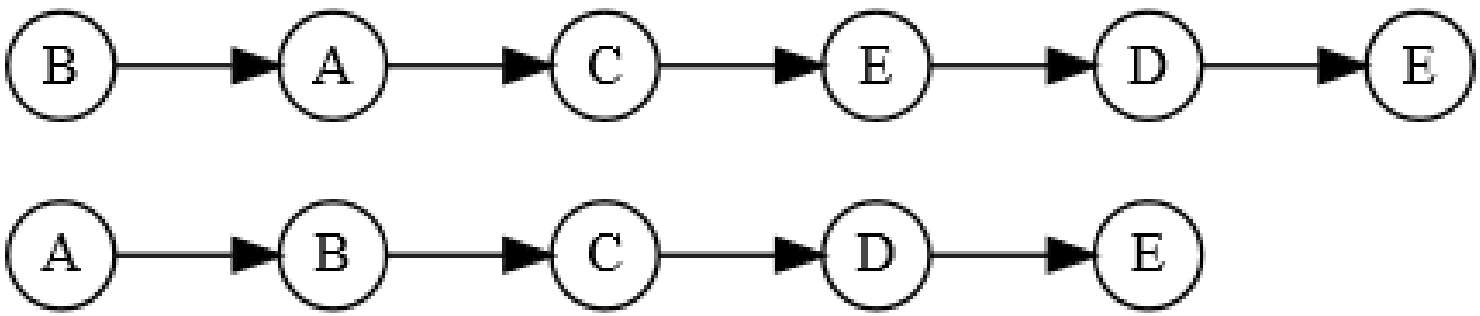
Seminarium: Matematyka dyskretna

5 listopada 2020

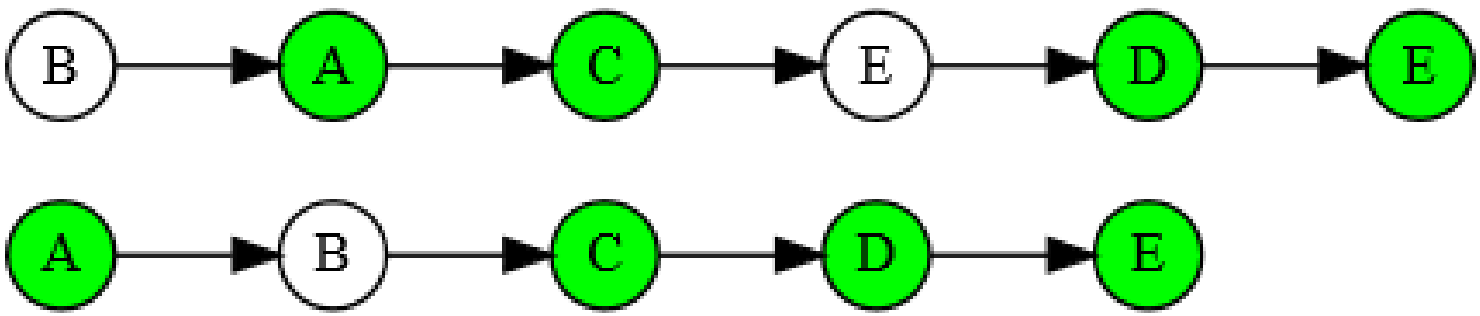
Agenda

Longest Common Subsequence

Longest Common Subsequence



Longest Common Subsequence



Graph Matching

Chcemy uzyskać podobne narzędzie, ale dla drzew (grafów):

Graph Matching

Chcemy uzyskać podobne narzędzie, ale dla drzew (grafów):

- sparować odpowiadające sobie wierzchołki

Graph Matching

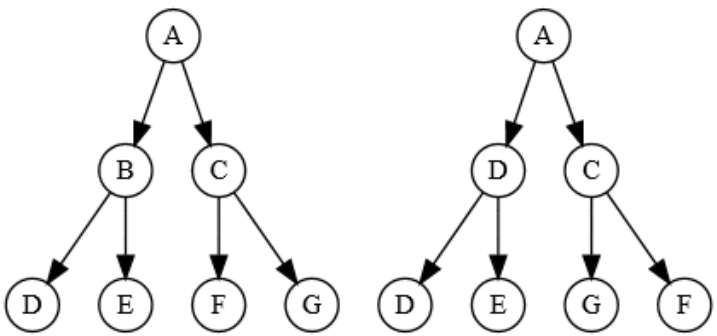
Chcemy uzyskać podobne narzędzie, ale dla drzew (grafów):

- sparować odpowiadające sobie wierzchołki
- "zachować strukturę"

Graph Matching

Chcemy uzyskać podobne narzędzie, ale dla drzew (grafów):

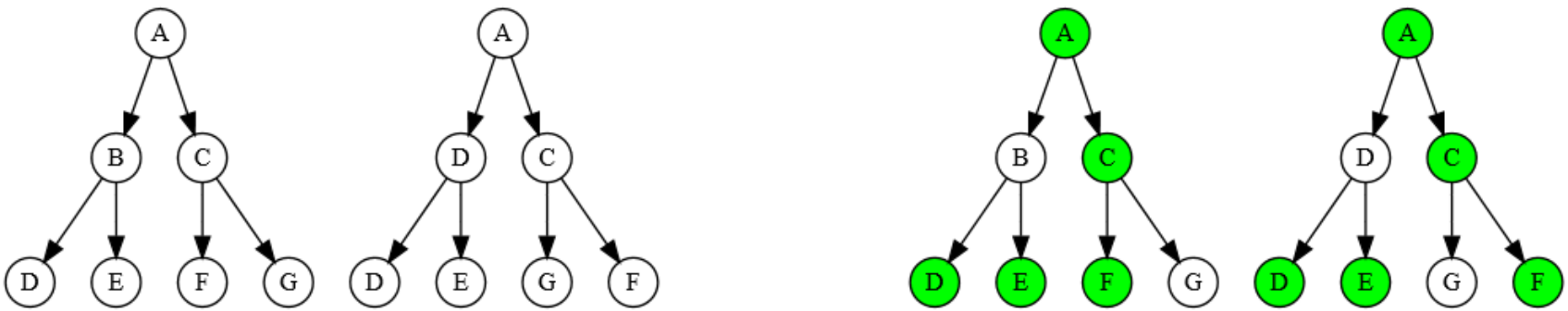
- sparować odpowiadające sobie wierzchołki
- "zachować strukturę"



Graph Matching

Chcemy uzyskać podobne narzędzie, ale dla drzew (grafów):

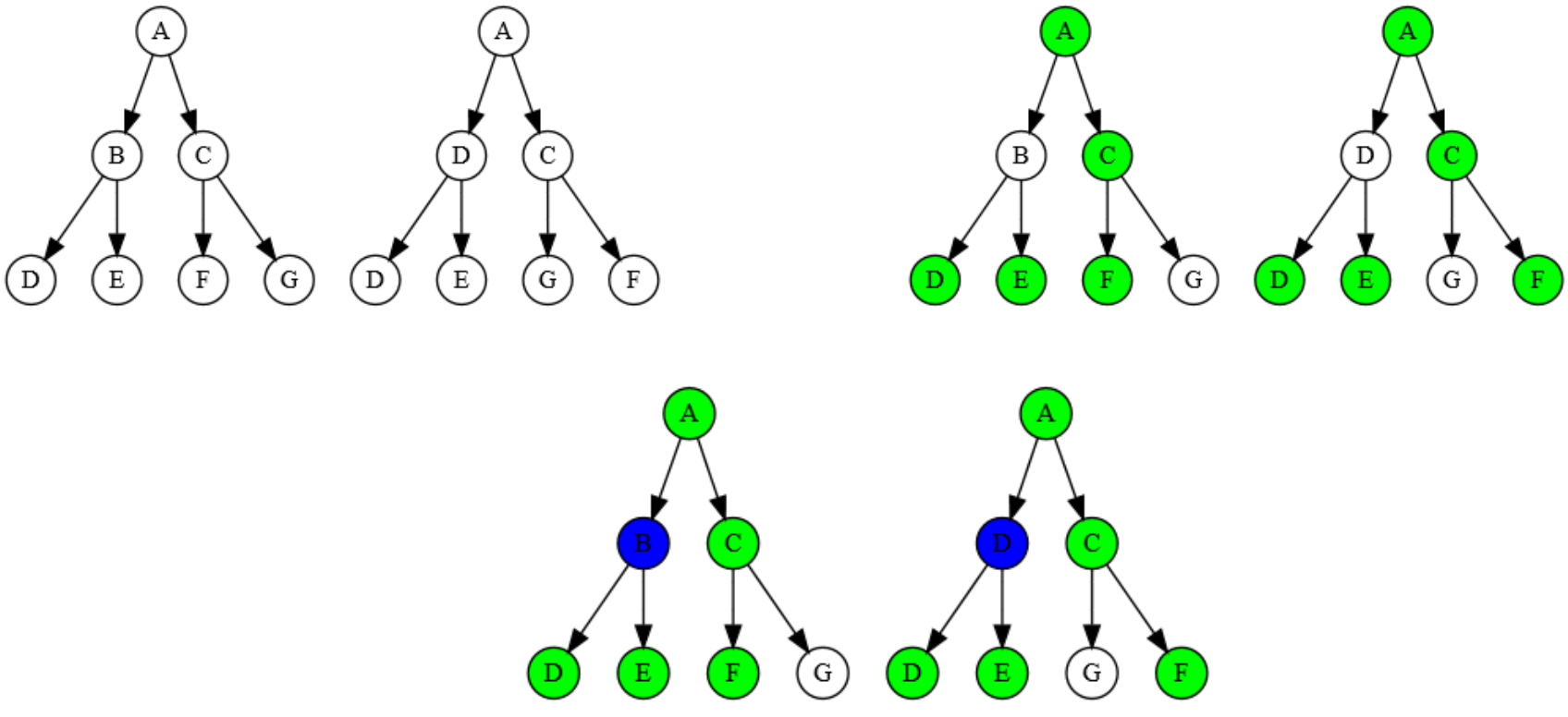
- sparować odpowiadające sobie wierzchołki
- "zachować strukturę"



Graph Matching

Chcemy uzyskać podobne narzędzie, ale dla drzew (grafów):

- sparować odpowiadające sobie wierzchołki
- "zachować strukturę"



Ordered Tree Matching

Ordered Tree Matching

Niech T_1 i T_2 będą drzewami z N_1 i N_2 wierzchołkami. Przypuścmy także, że mamy jakiś porządek na tych drzewach i $T[i]$ oznacza i-ty wierzchołek drzewa T.

Ordered Tree Matching

Niech T_1 i T_2 będą drzewami z N_1 i N_2 wierzchołkami. Przypuścmy także, że mamy jakiś porządek na tych drzewach i $T[i]$ oznacza i-ty wierzchołek drzewa T.
Dopasowaniem nazywamy trójkę (M, T_1, T_2), gdzie M jest zbiorem par takich liczb naturalnych (i, j), że:

1. $1 \leqslant i \leqslant N_1$, $1 \leqslant j \leqslant N_2$
2. Dla każdej pary (i_1, j_1), (i_2, j_2) należącej do M:
   - $i_1 = i_2$ wtedy i tylko wtedy, gdy $j_1 = j_2$,
   - $T_1[i_1]$ jest przodkiem $T_1[i_2]$ wtedy i tylko wtedy, gdy $T_2[j_1]$ jest przodkiem $T_2[j_2]$,
   - $T_1[i_1]$ jest na lewo od $T_1[i_2]$ wtedy i tylko wtedy, gdy $T_2[j_1]$ jest na lewo od $T_2[j_2]$.

Ordered Tree Matching

Niech T_1 i T_2 będą drzewami z N_1 i N_2 wierzchołkami. Przypuścmy także, że mamy jakiś porządek na tych drzewach i $T[i]$ oznacza i-ty wierzchołek drzewa T.
Dopasowaniem nazywamy trójkę (M, T_1, T_2), gdzie M jest zbiorem par takich liczb naturalnych (i, j), że:

1. $1 \leqslant i \leqslant N_1$, $1 \leqslant j \leqslant N_2$
2. Dla każdej pary (i_1, j_1), (i_2, j_2) należącej do M:
   - $i_1 = i_2$ wtedy i tylko wtedy, gdy $j_1 = j_2$,
   - $T_1[i_1]$ jest przodkiem $T_1[i_2]$ wtedy i tylko wtedy, gdy $T_2[j_1]$ jest przodkiem $T_2[j_2]$,
   - $T_1[i_1]$ jest na lewo od $T_1[i_2]$ wtedy i tylko wtedy, gdy $T_2[j_1]$ jest na lewo od $T_2[j_2]$.

$\gamma(M) = \sum_{(i,j)\in M} \gamma(T_1[i] \to T_2[j]) + \sum_{i\in I} \gamma(T_1[i] \to \Lambda) + \sum_{j\in J} \gamma(\Lambda \to T_2[j])$
Naszym celem jest znalezienie dopasowania minimalizującego $\gamma(M)$

Odległość edycyjna na uporządkowanych drzewach

Odległość edycyjna na uporządkowanych drzewach

Możliwe operacje:

- zmiana etykiety

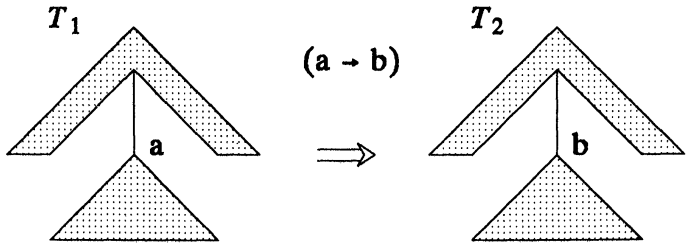


FIG. 1

- usunięcie wierzchołka

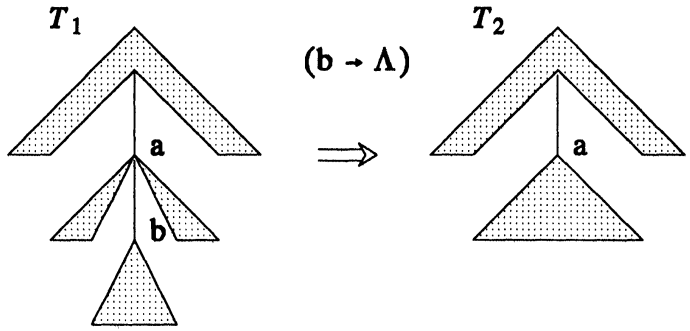


FIG. 2

- dodanie wierzchołka

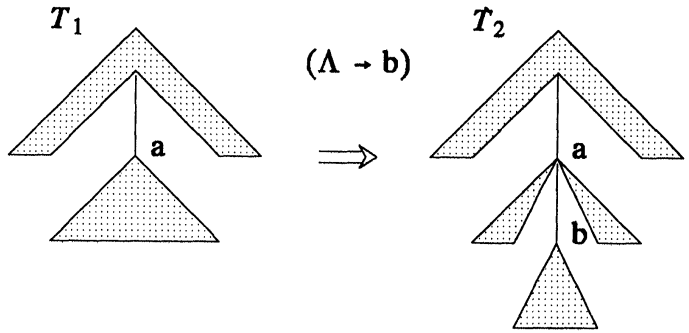


FIG. 3

Odległość edycyjna na uporządkowanych drzewach

Operacją edycyjną definiujemy parę $(a, b) \neq (\Lambda, \Lambda)$. Będziemy ją oznaczali $a \rightarrow b$.

Niech S będzie ciągiem $s_1, \ldots, s_k$. Będziemy mówili, że S przekształca T_1 w T_2, jeśli po zaaplikowaniu operacji $s_1, \ldots, s_k$ na drzewie T_1 otrzymamy drzewo T_2.

Niech $\gamma(a, b)$ będzie kosztem operacji edycyjnej. γ musi być metryką. γ rozszerzamy na S: $\gamma(S) = \sum_{i=1}^{k} \lambda(s_i)$. Ostatecznie odległość edycyjną pomiędzy drzewami T_1 i T_2 definiujemy:

$$\delta(T_1, T_2) = \min(\gamma(S), \text{dla wszystkich ciągów } S \text{ przekształcających } T_1 \text{ w } T_2)$$

Odległość edycyjna na uporządkowanych drzewach

Theorem

Dla każdego ciągu przekształcającego S istnieje dopasowanie M, dla którego $\gamma(S) \geqslant \gamma(M)$. Z kolei dla każdego dopasowania istnieje ciąg przekształcający o takim samym koszcie.

Minimalna odległość edycyjna

Oznaczenia:

- Niech $T[i]$ będzie i-tym wierzchołkiem w porządku postorder,
- $l(i)$ jest najbardziej lewym liściem dla wierzchołka $T[i]$,
- $p(i)$ jest rodzicem $T[i]$,
- $anc(i)$ jest zbiorem przodków $T[i]$ (włącznie z nim),
- $T[i \ldots j]$ jest uporządkowanym lasem zawierającym wierzchołki od i do j,
- $forrest(i) = T[1 \ldots i]$,
- $tree(i) = T[l(i) \ldots i]$,
- $size(i)$ jest liczbą wierzchołków dla $tree(i)$.
- $forestdist(i, j)$ jest odległością edycyjną dla lasów $T[1 \ldots i]$ i $T_2[1 \ldots j]$
- $treedist(i, j)$ jest odległością edycyjną dla drzew ukorzenionych w wierzchołkach w $T_1[i]$ i $T_2[j]$

Minimalna odległość edycyjna

LEMMA 4. *Let* $i_1 \in anc(i)$ *and* $j_1 \in anc(j)$. *Then*

$$forestdist(l(i_1)..i, l(j_1)..j) = \min \begin{cases} forestdist(l(i_1)..i-1, l(j_1)..j) + \gamma(T_1[i] \to \Lambda), \\ forestdist(l(i_1)..i, l(j_1)..j-1) + \gamma(\Lambda \to T_2[j]), \\ forestdist(l(i_1)..l(i)-1, l(j_1)..l(j)-1) \\ \quad + forestdist(l(i)..i-1, l(j)..j-1) \\ \quad + \gamma(T_1[i] \to T_2[j]). \end{cases}$$

Minimalna odległość edycyjna

LEMMA 5. *Let* $i_1 \in anc(i)$ *and* $j_1 \in anc(j)$. *Then*

(1) *If* $l(i) = l(i_1)$ *and* $l(j) = l(j_1)$

$$forestdist(l(i_1)..i, l(j_1)..j) = \min \begin{cases} forestdist(l(i_1)..i-1, l(j_1)..j) + \gamma(T_1[i] \to \Lambda), \\ forestdist(l(i_1)..i, l(j_1)..j-1) + \gamma(\Lambda \to T_2[j]), \\ forestdist(l(i_1)..i-1, l(j_1)..j-1) + \gamma(T_1[i] \to T_2[j]). \end{cases}$$

(2) *If* $l(i) \neq l(i_1)$ *or* $l(j) \neq l(j_1)$ (i.e., *otherwise*)

$$forestdist(l(i_1)..i, l(j_1)..j) = \min \begin{cases} forestdist(l(i_1)..i-1, l(j_1)..j) + \gamma(T_1[i] \to \Lambda), \\ forestdist(l(i_1)..i, l(j_1)..j-1) + \gamma(\Lambda \to T_2[j]), \\ forestdist(l(i_1)..l(i)-1, l(j_1)..l(j)-1) \\ \quad + treedist(i, j). \end{cases}$$

Minimalna odległość edycyjna

$LR\_keyroots(T) = \{k|$ nie istnieje takie $k' > k$, że $l(k) = l(k')\}$

Input: Tree T_1 and T_2.
Output: $Tree\_dist(i, j)$, where $1 \leqq i \leqq |T_1|$ and $1 \leqq j \leqq |T_2|$.
Preprocessing
(To compute $l(\ )$, $LR\_keyroots1$ and $LR\_keyroots2$)
Main loop
  for $i' := 1$ to $|LR\_keyroots(T_1)|$
    for $j' := 1$ to $|LR\_keyroots(T_2)|$
      $i = LR\_keyroots1[i']$;
      $j = LR\_keyroots2[j']$;
      Compute $treedist(i, j)$;

Minimalna odległość edycyjna

treedist(i, j):

$forestdist(\emptyset, \emptyset) = 0;$
for $i_1 := l(i)$ to i
  $forestdist(T_1[l(i)..i_1], \emptyset) = forestdist(T_1[l(i)..i_1 - 1], \emptyset) + \gamma(T_1[i_1] \to \Lambda)$
for $j_1 := l(j)$ to j
  $forestdist(\emptyset, T_2[l(j)..j_1]) = forestdist(\emptyset, T_2[l(j)..j_1 - 1]) + \gamma(\Lambda \to T_2[j_1])$
for $i_1 := l(i)$ to i
  for $j_1 := l(j)$ to j
    if $l(i_1) = l(i)$ and $l(j_1) = l(j)$ then
      $forestdist(T_1[l(i)..i_1], T_2[l(j)..j_1)] = \min\{$
        $forestdist(T_1[l(i)..i_1 - 1], T_2[l(j)..j_1]) + \gamma(T_1[i_1] \to \Lambda),$
        $forestdist(T_1[l(i)..i_1], T_2[l(j)..j_1 - 1]) + \gamma(\Lambda \to T_2[j_1]),$
        $forestdist(T_1[l(i)..i_1 - 1], T_2[l(j)..j_1 - 1]) + \gamma(T_1[i_1] \to T_2[j_1])\}$
      $treedist(i_1, j_1) = forestdist(T_1[l(i)..i_1], T_2[l(j)..j_1])$/* put in permanent array */
    else
      $forestdist(T_1[l(i)..i_1], T_2[l(j)..j_1]) = \min\{$
        $forestdist(T_1[l(i)..i_1 - 1], T_2[l(j)..j_1]) + \gamma(T_1[i_1] \to \Lambda),$
        $forestdist(T_1[l(i)..i_1], T_2[l(j)..j_1 - 1]) + \gamma(\Lambda \to T_2[j_1]),$
        $forestdist(T_1[l(i)..l(i_1) - 1], T_2[l(j)..l(j_1) - 1]) + treedist(i_1, j_1)\}$

Złożoność

Złożoność

Zauważmy, że:

$$LR\_keyroots(T) \leqslant leaves(T) \tag{1}$$

Złożoność

Zauważmy, że:

$$LR\_keyroots(T) \leqslant leaves(T) \tag{1}$$

Następnie zdefiniujmy:

$$LR\_colldepth(i) = |anc(i) \cap LR\_keyroots(T)| \tag{2}$$

$$LR\_colldepth(T) = \max_i(LR\_colldepth(i)) \tag{3}$$

Wówczas otrzymujemy:

$$\sum_{i=1}^{LR\_keyroots(T)} Size(i) = \sum_{j=1}^{N} |LR\_colldepth(j)| \tag{4}$$

a następnie:

$$\sum_{i=1}^{|LR\_keyroots(T_1)|} \sum_{j=1}^{|LR\_keyroots(T_2)|} Size(i) \cdot Size(j) =$$

$$= \sum_{i=1}^{|LR\_keyroots(T_1)|} Size(i) \cdot \sum_{j=1}^{|LR\_keyroots(T_2)|} Size(j) =$$

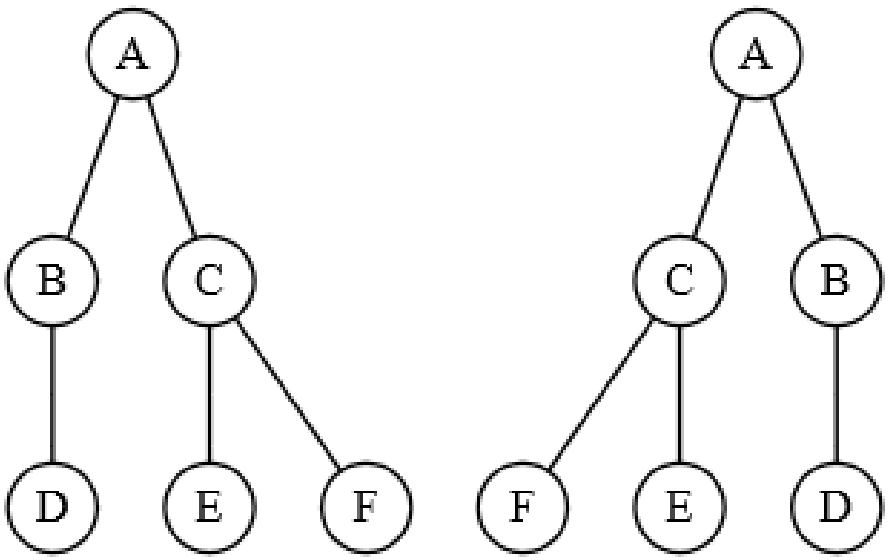
$$= \sum_{i=1}^{N_1} LR\_colldepth(i) \cdot \sum_{j=1}^{N_2} LR\_colldepth(j) \leqslant$$

$$\leqslant N_1 \cdot N_2 \cdot LR\_colldepth(T_1) \cdot LR\_colldepth(T_2) \leqslant$$

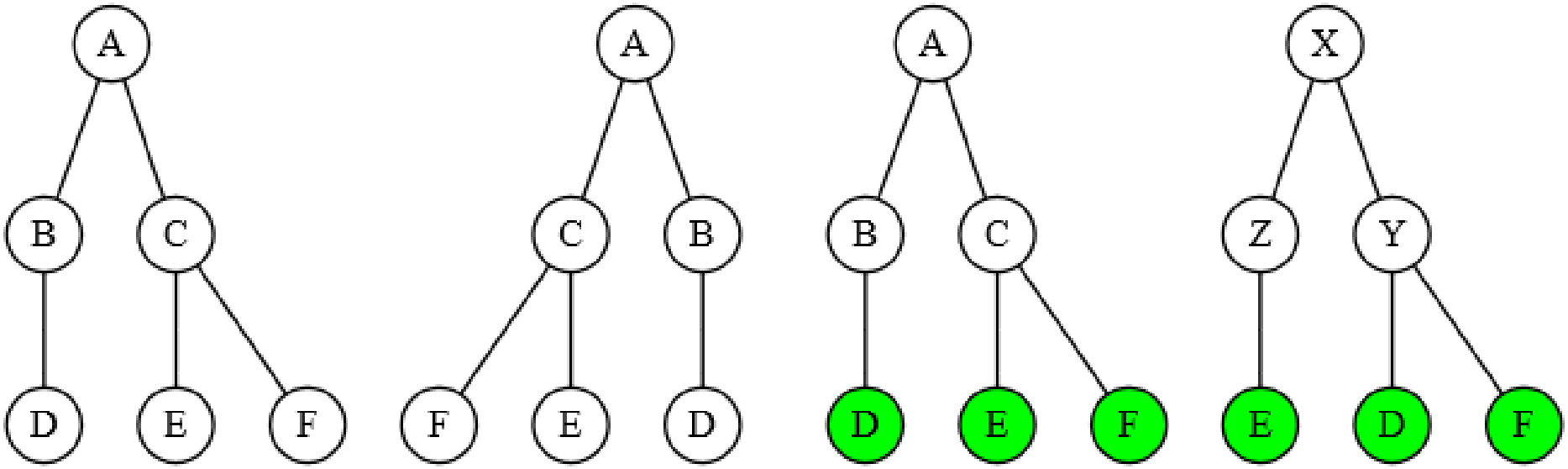
$$\leqslant N_1 \cdot N_2 \cdot \min(depth(T_1), leaves(T_1)) \cdot \min(depth(T_2), leaves(T_2))$$

Drzewa nieuporządkowane

Drzewa nieuporządkowane



Drzewa nieuporządkowane



Drzewa nieuporządkowane

Warunek ze zgodnością porządku musimy zastąpić:

Dla każdej trójki $(i_l, j_l), (i_2, j_2), (i_3, j_3) \in M$ zdefiniujmy:

$$t_1[I] = lca(t_1[i_l], t_1[i_2])$$
$$t_2[J] = lca(t_2[j_l], t_2[j_2])$$

Wówczas $t_1[I]$ jest przodkiem $t_1[i_3]$ wtedy i tylko wtedy, gdy $t_2[J]$ jest przodkiem $t_2[j_3]$.

Drzewa nieuporządkowane

Warunek ze zgodnością porządku musimy zastąpić:

Dla każdej trójki $(i_l, j_l), (i_2, j_2), (i_3, j_3) \in M$ zdefiniujmy:

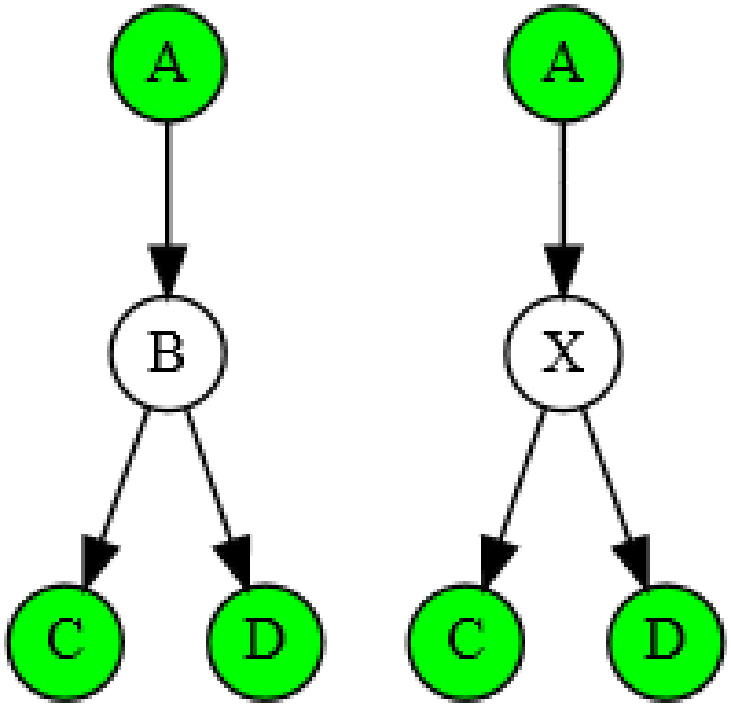
$$t_1[l] = lca(t_1[i_l], t_1[i_2])$$
$$t_2[J] = lca(t_2[j_l], t_2[j_2])$$

Wówczas $t_1[l]$ jest przodkiem $t_1[i_3]$ wtedy i tylko wtedy, gdy $t_2[J]$ jest przodkiem $t_2[j_3]$.

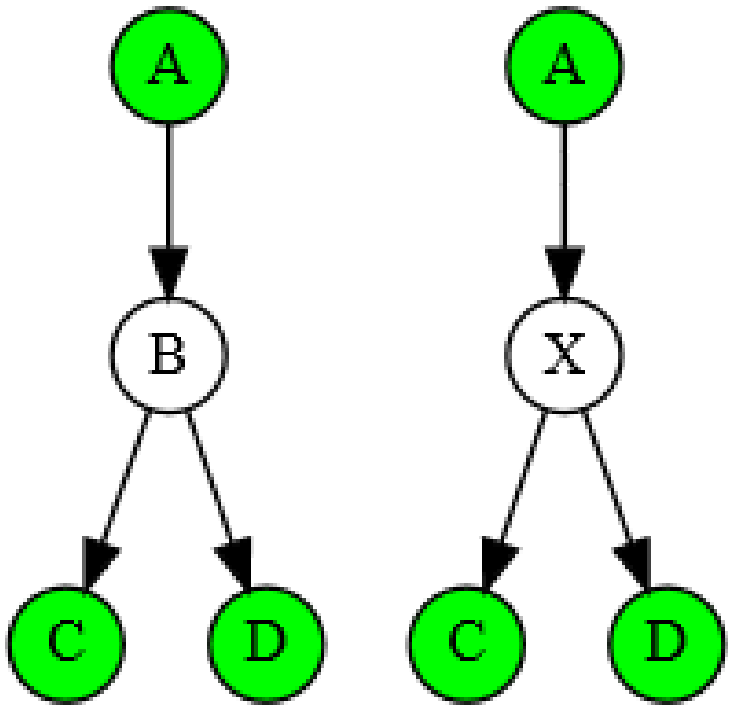
K.Zhang - A constrained edit distance between unordered labeled trees

Skierowany graf acykliczny

Skierowany graf acykliczny

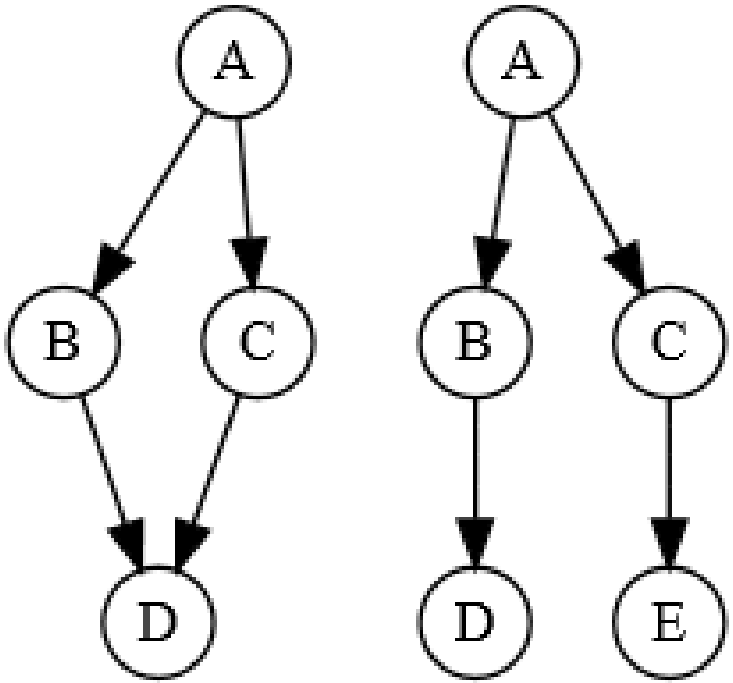


Skierowany graf acykliczny

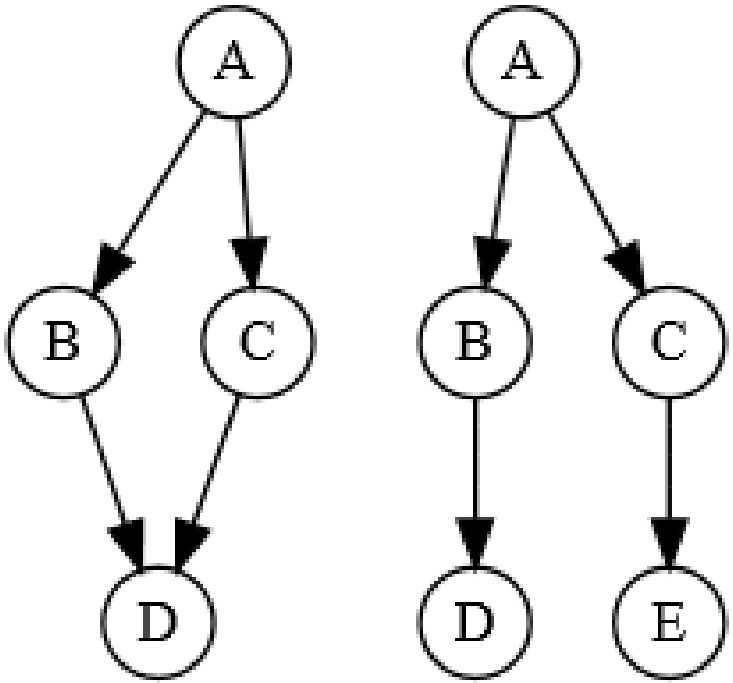


Może się okazać, że takiego dopasowania wierzchołków nie da się uzyskać dla zadanych operacji edycji grafu (np. gdy usuwany jedynie wiechołki o stopniu maksymalnie 2)

Skierowany graf acykliczny

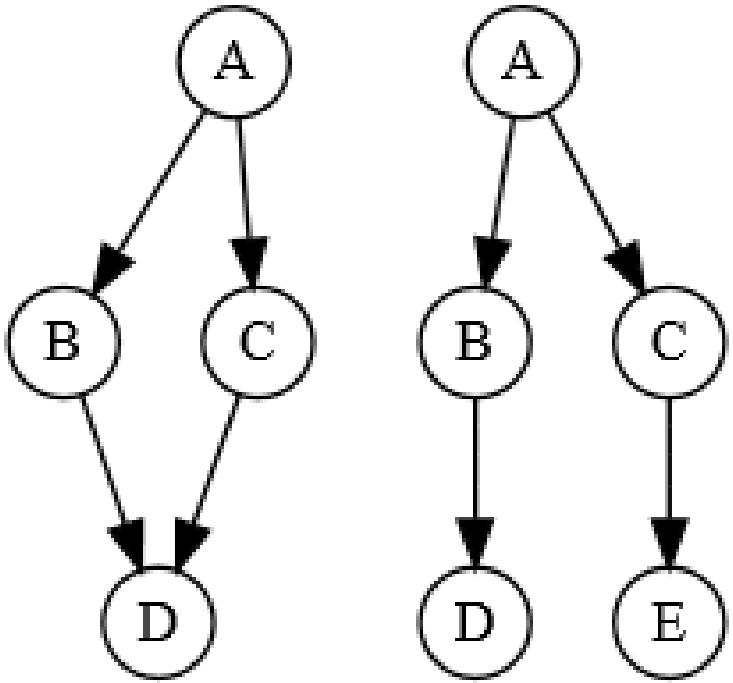


Skierowany graf acykliczny



Czasem dodaje się także operację rozdzielania wierzchołka (i analogicznie - scalania)

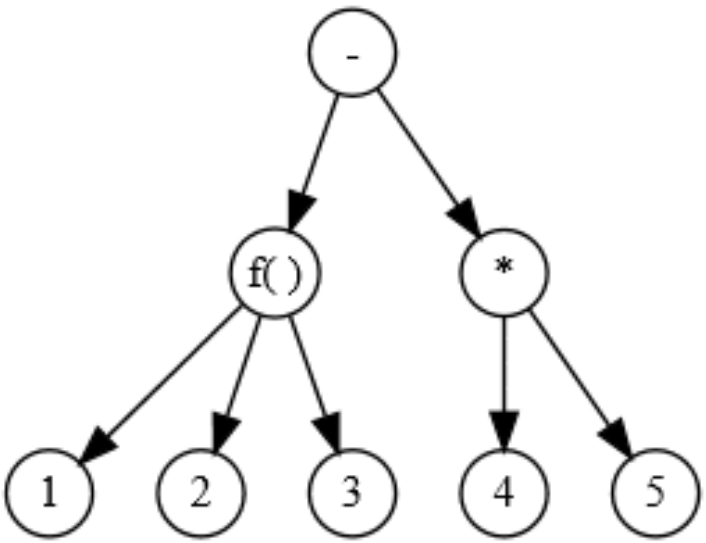
Skierowany graf acykliczny



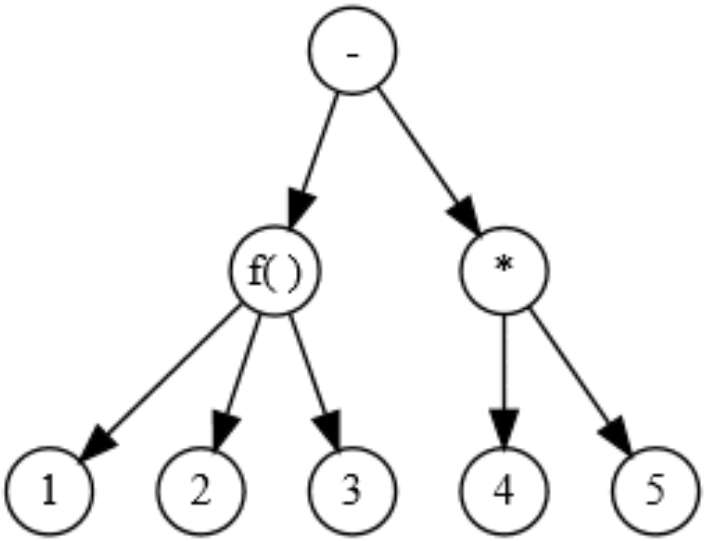
Czasem dodaje się także operację rozdzielania wierzchołka (i analogicznie - scalania)
K.Zhang, J. T. L. Wang, D. Shasha - On the Editing Distance between Undirected Acyclic Graphs and Related Problems

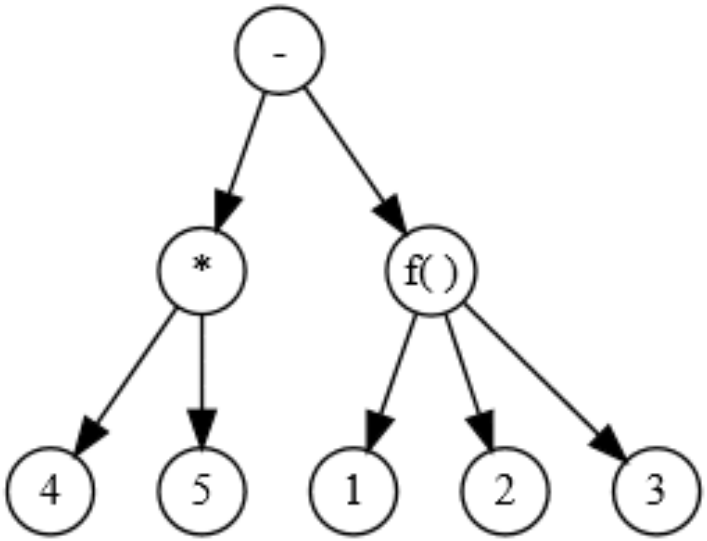
Częściowo uporządkowane drzewa (DAGi)

Częściowo uporządkowane drzewa (DAGi)



Częściowo uporządkowane drzewa (DAGi)





Dziękuję za uwagę.

Literatura:

- K.Zhang, D. Shasha - Simple Fast Algorithms for the Editing Distance Between Trees and Related Problems
- K.Zhang - A constrained edit distance between unordered labeled trees
- K.Zhang, J. T. L. Wang, D. Shasha - On the Editing Distance between Undirected Acyclic Graphs and Related Problems