# Indukcja na grafach

18 października 2018

## 1 Definicje

**Definicja 1** (Rodzina zbiorów).

$$\mathcal{A} = \{A_i : A_i \subseteq X, i \in \mathbb{N}\}$$

*Będziemy używać jedynie skończonych rodzin.*

**Definicja 2** (Suma uogólniona).

$$\bigcup \mathcal{A} = \bigcup_i A_i = \{x \in X : \exists_i x \in A_i\}$$

**Definicja 3** (Dowód indukcyjny na acyklicznych, skończonych grafach skierowanych). *Dowód przebiega w następujący sposób.*

*Najpierw definiujemy zbiór X, pewną własność C oraz zbiory poprzedników $b(x)$ wszystkich elementów zbioru X. Każdy element x ma własność C lub jej nie ma.*

*Definiujemy również zbiór poprzedników danego zbioru jako.*

$$B(X) = \bigcup_{x \in X} b(x)$$

*Algorytm przebiega następująco. Dzielimy graf na warstwy algorytmem BFS, zaczynając od wierzchołków do których nie wchodzi żadna krawędź (warstwa $W_0$). Wierzchołek v należy do warstwy $W_k$ wtedy i tylko wtedy, gdy jego najkrótsza ścieżka do wierzchołka z warstwy $W_0$ ma długość k.*

*Zakładamy tutaj, że wszyscy poprzednicy należą do poprzednich warstw. (Graf skierowany acykliczny.)*

$$\forall_k \forall_{w \in W_k} b(w) \subseteq \bigcup_{i<k} A_i$$

*Naszym celem jest pokazanie, że jeśli dla wszystkich wcześniejszych wierzchołków zachodzi własność C i potrafimy na tej podstawie udowodnić własność C dla kolejnych wierzchołków, to własność C zachodzi dla wszystkich wierzchołków. Dowodzimy to w następujący sposób.*

1. *Podstawa indukcji*

$$\forall_{x \in W_0} C(x)$$

2. *Krok indukcyjny*

$$(\forall_{w \in W_n} (\forall_{p \in b(w)} C(p))) \implies C(w)$$

*Wówczas zachodzi:*

$$(\forall_{n \in \mathbb{N}} (\forall_{w \in W_n} (\forall_{p \in b(w)} C(p)) \implies C(w))) \implies \forall_{x \in X} C(x)$$

*Uwagi!*

- *Graf nie musi być spójny.*

**Definicja 4.**

$$[x = y] = \begin{cases} 1, & x = y \\ 0, & x \neq y \end{cases} \tag{1}$$

**Definicja 5** (Odległość Levenshteina)**.** *Mamy dwa ciągi znaków* $(a_1, a_2, \ldots a_n)$ *oraz* $(b_1, b_2, \ldots b_m)$ *długości odpowiednio* $n$ *i* $m$*. Należy znaleźć minimalną liczbę operacji potrzebnych do przekształcenia ciągu* $a$ *w ciąg* $b$*.*
*Dopuszczalne są następujące operacje:*

- *Wstaw dowolny znak* $c$ *w dowolne miejsce ciągu* $a$*.*
- *Usuń dowolny znak* $c$ *z ciągu* $a$*.*
- *Zamień dowolny znak* $c$ *ciągu* $a$ *na inny znak.*

*Dla ustalenia uwagi oznaczmy, tą liczbę przez L(a,b).*

**Lemat 1.** $L(a, b) \geqslant max(n, m) - min(n, m) = |n - m|$
*Jeśli dwa ciągi mają różne długości to zawsze musimy przedłużyć lub skrócić ciąg [a] o różnicę ich długości.*

**Lemat 2.** $L(a, b) \leqslant max(n, m)$
*Możemy po prostu wymieniać wszystkie znaki po kolei.*

*Z powyższych lematów możemy wywnioskować kolejne.*

**Lemat 3.** $n = 0 \implies L(a, b) = m$

**Lemat 4.** $m = 0 \implies L(a, b) = n$

## 2 Zadanie

Napisać algorytm obliczający L(a, b) oraz wypisujący ciąg operacji.

**Definicja 6** (Prefiks)**.** *Niech* $a[1 \ldots i]$ *będzie prefikem a długości i.*

**Twierdzenie 1.**

$$d(0,0) = 0$$
$$d(i,j) = min(d(i,j-1)+1, d(i-1,j)+1, d(i-1,j-1)+[a_i \neq b_j])$$

*Wówczas* $d(i,j) = L(a[1,\ldots,i], b[1,\ldots,j])$

*Dowód.* Pokażemy to indukcyjnie.
Zdefiniujmy kolejne warstwy:

$$W_0 = \{(0,0)\}$$
$$W_k = \{(i,j) : i+j = k\}$$
$$W_{n+m} = \{(n,m)\}$$
$$\begin{bmatrix} 0 & 1 & 2 & 3 & \ldots & m \\ 1 & 2 & 3 & 4 & \ldots & m+1 \\ 2 & 3 & 4 & 5 & \ldots & m+2 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ n & n+1 & n+2 & n+3 & \ldots & n+m \end{bmatrix}$$

1. Podstawa indukcji (k = 0).
   Dla pustych ciągów nie trzeba nic robić.

2. Krok indukcyjny

   - k = 1

     $$W_1 = \{(0,1),(1,0)\} \tag{2}$$
     $$B(W_1) = B((0,1)) \cup B((1,0)) = \{(0,0)\} = W_0 \subseteq W_0 \tag{3}$$

   - $k > 1$

     $$\forall_{(i,j) \in W_k} \{(i-1,j),(i,j-1),(i-1,j-1)\} \subseteq W_{k-2} \cup W_{k-1}$$
     $$B(W_k) \subseteq W_{k-2} \cup W_{k-1}$$

   Wybranie wartości minimum to po prostu wybranie odpowiedniego ruchu w danym momencie, takiego dla którego sumaryczny koszt jest najniższy. Sumaryczny koszt obliczamy dynamicznie na podstawie wcześniej obliczonych kosztów.

□

# 3 Zadanie dodatkowe

Należy rozpatrzyć dwa inne przypadki.

- Wszystkie operacje mają różne koszty.
- Nie ma operacji zamiany znaku.
- Inny podział na warstwy. Pokazać, że istnieje algorytm w pamięci O(min(n,m)). Tym razem zaczniemy numerację warstw od 1.

$$W_1 = \{(0,0)\}$$
$$W_k = \{(i,j) : j(n+1) + i + 1 = k\}$$
$$W_{(n+1)(m+1)} = \{(n,m)\}$$
$$W_k = \{(k \mod (n+1), \frac{k}{n+1}\}$$
$$\begin{bmatrix} 1 & n+2 & 2n+3 & \dots & m(n+1)+1 \\ 2 & n+3 & 2n+4 & \dots & m(n+1)+2 \\ 3 & n+4 & 2n+5 & \dots & m(n+1)+3 \\ 4 & n+5 & 2n+6 & \dots & m(n+1)+4 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ n+1 & 2(n+1) & 3(n+1) & \dots & (m+1)(n+1) \end{bmatrix}$$