Lista 1, zadanie 5

Założenia i notacja

Dany jest skierowany graf acykliczny $G = (V, E)$, $|V| = n$, $|E| = m$. Długość ścieżki liczona jest w liczbie krawędzi. Zakładamy, że graf jest przedstawiony listami następników.

1 Liczenie *długości* najdłuższej ścieżki

Idea

W DAG-u istnieje uporządkowanie topologiczne $T = (v_1, \ldots, v_n)$, w którym każda krawędź idzie „w prawo”. Przechodząc w tej kolejności można dynamicznie uzupełniać wartość

$$\mathrm{dp}[v] = \text{długość najdłuższej ścieżki kończącej się w } v.$$

Algorytm

Algorithm 1 LongestPathLength

Require: $G = (V, E)$ — DAG
Ensure: długość najdłuższej ścieżki w G

$T \leftarrow$ topologiczne uporządkowanie (np. algorytm Kahna)
for $v \in V$: $\mathrm{dp}[v] \leftarrow 0$
for v w kolejności T **do** ▷ każdy wierzchołek raz
    for all w następniki v **do**
        if $\mathrm{dp}[v] + 1 > \mathrm{dp}[w]$ **then**
            $\mathrm{dp}[w] \leftarrow \mathrm{dp}[v] + 1$
        end if
    end for
end for
return $\max_{v \in V} \mathrm{dp}[v]$

Złożoność. Uporządkowanie topologiczne $O(n + m)$, pętla również $O(n + m)$. Pamięć $O(n)$.

2 Wypisywanie jednej maksymalnej ścieżki

Modyfikacja

Wystarczy zapamiętywać poprzednik, z którego uzyskano najlepszą wartość `dp`.

$$\mathrm{parent}[w] = \begin{cases} v, & \text{gdy } \mathrm{dp}[v] + 1 > \mathrm{dp}[w], \\ \texttt{nil}, & \text{jeśli brak aktualizacji.} \end{cases}$$

Algorithm 2 LongestPath

Require: $G = (V, E)$
Ensure: jedna najdłuższa ścieżka w kolejności źródło→cel

$T \leftarrow$ topologiczne uporządkowanie
for $v \in V$ **do** $\mathrm{dp}[v] \leftarrow 0$, $\mathrm{parent}[v] \leftarrow$ `nil`
end for
for v w T **do**
    for all $w \in \mathrm{succ}(v)$ **do**
        if $\mathrm{dp}[v] + 1 > \mathrm{dp}[w]$ **then**
            $\mathrm{dp}[w] \leftarrow \mathrm{dp}[v] + 1$
            $\mathrm{parent}[w] \leftarrow v$
        end if
    end for
end for
$u \leftarrow \arg\max_v \mathrm{dp}[v]$ ▷ koniec ścieżki
ścieżka $\leftarrow [\,]$
while $u \neq$ `nil` **do**
    wstaw u na początek ścieżki
    $u \leftarrow \mathrm{parent}[u]$
end while
return ścieżka

Złożoność. Niezmieniona: $O(n + m)$ czasu, $O(n)$ pamięci (dodatkowa tablica `parent`).

Poprawność (krótko)

Przetwarzając wierzchołki w kolejności topologicznej mamy pewność, że dla każdej krawędzi $v \rightarrow w$ wartość $\mathrm{dp}[v]$ jest finalna w momencie aktualizowania $\mathrm{dp}[w]$; otrzymujemy największą możliwą długość ścieżki kończącej się w w. Indukcja po kolejnych wierzchołkach dowodzi poprawności tablicy `dp`, a śledzenie `parent` od elementu o maksymalnym `dp` odtwarza ścieżkę o tej właśnie długości. □