Informatyczny kącik olimpijski (47): Godzilla

W tym miesiącu omówimy zadanie, które rozwiązywali uczestnicy Obozu Naukowo-Treningowego im. A. Kreczmara w 2009 r.

Sieć telewizji kablowej składa się z n węzłów i m jednokierunkowych połączeń. Do każdego węzła sieci jest podłączony co najmniej jeden dom. Niektóre węzły są wyróżnione i do nich bezpośrednio transmitowany jest program. W danym domu można go oglądać, jeśli istnieje połączenie (niekoniecznie bezpośrednie) od wyróżnionego węzła do węzła, do którego podłączony jest dom.

Niestety, sieć jest narażona na ataki złośliwej Godzilli, która, nie wiedzieć czemu, żywi się infrastrukturą telewizji kablowej. Co dzień zjada ona jedno połączenie z sieci. Mając daną listę s połączeń, które kolejno zje Godzilla, należy wyznaczyć, ile minimalnie węzłów należy oznaczyć jako wyróżnione każdego dnia, tak aby każdy dom mógł odbierać program.

Sieć kablową możemy traktować jako graf skierowany G o n wierzchołkach i m krawędziach. Aby znaleźć minimalny zbiór wyróżnionych węzłów, wyznaczamy podział grafu G na silnie spójne składowe. Silnie spójną składową nazwiemy *początkową*, jeśli nie wchodzi do niej żadna krawędź (tzn. do żadnego wierzchołka tej składowej nie wchodzi krawędź z wierzchołka spoza tej składowej). Zauważmy, że w każdej początkowej składowej musimy wyróżnić co najmniej jeden wierzchołek, aby dostarczyć program do wierzchołków tej składowej. A ponieważ do każdej niepoczątkowej składowej istnieje ścieżka ze składowej początkowej, więc taki zbiór wierzchołków jest wystarczający.

Znajdowanie podziału na silnie spójne składowe możemy zrealizować dwukrotnym przejściem grafu w głąb w czasie $O(n+m)$. Jeśli po każdym usunięciu krawędzi będziemy wyznaczali ten podział od nowa, dostaniemy algorytm o koszcie czasowym $O(s \cdot (n+m))$.

Chcielibyśmy umieć szybko uaktualniać strukturę składowych po usunięciu krawędzi. Nie jest jednak jasne, jak to zrobić, gdyż po takiej operacji jedna ze składowych może rozpaść się na mniejsze składowe i wydaje się, że nie da się ich wyznaczyć szybciej niż w czasie liniowym względem rozmiaru oryginalnej składowej. Zastosujemy więc pomysłową sztuczkę i odwrócimy czas: zaczniemy od grafu, w którym usunięto wszystkie s krawędzi, i będziemy je dodawać do grafu w odwrotnej kolejności. Tym sposobem pojedynczą operacją będzie połączenie kilku składowych w jedną, a to już wygląda przyjaźniej.

Niech k będzie aktualną liczbą *początkowych* składowych. Przez cały czas będziemy utrzymywać podział zbioru wierzchołków grafu

$$V = A_1 \cup A_2 \cup \ldots \cup A_k \cup B_1 \cup B_2 \cup \ldots \cup B_k.$$

Zbiór A_i tworzą wierzchołki i-tej początkowej składowej; do zbioru B_i należą zaś wierzchołki, które są osiągalne z A_i (jeśli wierzchołek jest osiągalny z kilku początkowych składowych, to należy do zbioru B_i dla jednej z tych składowych).

Zobaczmy, co się dzieje, gdy dodajemy do grafu nową krawędź $u \to v$. Jeśli $v \in B_i$ dla pewnego i, to dodanie tej krawędzi nie spowoduje żadnych zmian w strukturze początkowych składowych, a więc nie musimy nic robić.

W przeciwnym przypadku $v \in A_i$ dla pewnego i. Będziemy przeszukiwać graf transponowany G^T (czyli przechodzić krawędzie grafu G w odwrotnym kierunku), np. algorytmem DFS, począwszy od wierzchołka u. Dla każdego napotkanego wierzchołka w ze zbioru B_i, przenosimy go do zbioru A_i (wiemy, że jest cykl $w \rightsquigarrow u \to v \rightsquigarrow w$, gdyż $w \in B_i$ jest osiągalny z A_i). Oczywiście, dla każdego napotkanego wierzchołka $w \in A_i$ ucinamy przeszukiwanie, gdyż gdy raz wejdziemy do A_i, to już z niego nie wyjdziemy.

Jeśli zaś napotkamy wierzchołek $w \in A_j \cup B_j$ dla $i \neq j$, to znaczy, że znaleźliśmy ścieżkę, która pokazuje, iż składowa A_i stała się osiągalna ze składowej A_j. Zatem powiększamy B_j, wykonując $B_j := B_j \cup A_i \cup B_i$, usuwamy zbiory A_i, B_i i kończymy przeszukiwanie. Liczba początkowych składowych zmalała o jeden.

Oszacujmy złożoność czasową naszego algorytmu. Obliczenie podziału dla początkowego grafu G bez s zjedzonych krawędzi realizujemy w czasie $O(n+m)$. Podział ten będziemy przechowywać w strukturze danych dla zbiorów rozłącznych (ang. *find-union*). Wtedy każda operacja **1)** zapytania, do którego ze zbiorów należy dany wierzchołek i **2)** złączenia zbiorów będzie miała „prawie stały" koszt zamortyzowany – a dokładniej $O(\log^* n)$, przy czym w praktyce $\log^* n \leqslant 5$. Dodatkowo, sumaryczny czas przeszukiwania grafu G^T wynosi $O(n+m)$, gdyż każdą krawędzią przechodzimy co najwyżej raz.

Zatem całkowity koszt czasowy to $O(n+m+s\log^* n)$, przy zużyciu pamięci rzędu $O(n+m)$.

Tomasz IDZIASZEK