Informatyczny kącik olimpijski (34): Inspektor po raz drugi

Na powierzchni pewnej planety rozmieszczonych jest N baz, pomiędzy którymi wybudowano $N-1$ dwukierunkowych połączeń o danych całkowitych długościach w_i, tworzących drzewo (pomiędzy każdymi dwiema bazami istnieje dokładnie jedna ścieżka). Inspektor kontrolujący bazy wprowadził Q zarządzeń, z których j-te jest postaci (u_j, v_j, a_j, b_j) i nakazuje przebudować wszystkie połączenia o długościach z przedziału $[a_j, b_j]$, które znajdują się na ścieżce pomiędzy bazami u_j i v_j. Naszym zadaniem jest obliczyć, ile połączeń należy przebudować, aby spełnić wymagania inspektora. Zakładamy, że N i Q, a także W – górne ograniczenie na liczby w_i, a_j i b_j – są tego samego rzędu wielkości.

W tym kąciku zajmiemy się zadaniem, o którym była już mowa poprzednio – *Inspector is coming*. Nie jest to pomyłka, zadanie ma bowiem drugie bardzo interesujące rozwiązanie. Treść zadania przypominamy na marginesie.

Tak jak poprzednio, ukorzeńmy drzewo w dowolnym wierzchołku. Tym razem spróbujmy pomyśleć o długościach krawędzi jak o chwilach, w których są aktywne – krawędź długości w_i jest aktywna w chwili w_i. Wówczas zarządzenie wymagające przebudowania krawędzi na ścieżce pomiędzy u_j i v_j o długościach z przedziału $[a_j, b_j]$ oznacza dokładnie tyle, że należy przebudować krawędzie z tej właśnie ścieżki, które są aktywne kiedykolwiek pomiędzy chwilami a_j i b_j. Rozbijmy takie zarządzenie na początek (w chwili a_j) oraz koniec (w chwili b_j). Zdarzeniami nazwijmy: początki zarządzeń, ich końce oraz momenty aktywności poszczególnych krawędzi. Posortujmy wszystkie zdarzenia względem czasu (sposób rozstrzygania remisów podczas sortowania pozostawiamy do przemyślenia Czytelnikowi) i spróbujmy je obsługiwać właśnie w tej kolejności.

Aktywne, czyli rozpoczęte i jeszcze niezakończone, zarządzenia będziemy zapisywać w pewnej strukturze danych. Dla każdego zdarzenia reprezentującego krawędź chcemy stwierdzić, czy trzeba tę krawędź przebudować, czy też nie. Jak już wspominaliśmy przy pierwszym rozwiązaniu, krawędź długości w prowadzącą z wierzchołka p do jego ojca q w drzewie należy przebudować dokładnie wtedy, gdy w poddrzewie wierzchołka p rozpoczyna się, ale nie kończy, jakaś ścieżka z zarządzeniem zawierającym w. Ponieważ zajmujemy się tylko aktywnymi w danej chwili zarządzeniami, więc wystarczy nam umieć stwierdzać, czy w naszej strukturze danych jest jakiekolwiek zarządzenie wychodzące z poddrzewa p poza to poddrzewo.

Jak zbudować strukturę danych umożliwiającą efektywne wykonywanie opisanych operacji? Nie pierwszy raz będziemy w takiej sytuacji korzystać z drzewa przedziałowego, wcześniej jednak musimy poczynić pewne obserwacje.

Po raz kolejny przypomnę, że o problemie znajdowania najniższego wspólnego przodka dwóch wierzchołków w drzewie, czyli LCA tych wierzchołków, można poczytać w *Delcie* 9/2007, „O dwóch równoważnych problemach", oraz w IKO w *Delcie* 7/2009.

Po pierwsze, zauważmy, że ścieżkę pomiędzy u i v możemy rozłożyć na dwie części idące w drzewie w górę: z u do $\mathrm{LCA}(u, v)$ oraz z v do $\mathrm{LCA}(u, v)$. Będziemy więc zapisywać dla zarządzenia (u_j, v_j, a_j, b_j), w momencie jego rozpoczęcia, liczbę $+1$ w wierzchołkach u_j i v_j oraz -2 w wierzchołku $\mathrm{LCA}(u_j, v_j)$, a przy zakończeniu tego zarządzenia będziemy usuwać te zapisy. Zauważmy, że wystawanie pewnej ścieżki z poddrzewa p jest w takim razie równoważne temu, że suma liczb zapisanych w poddrzewie p jest dodatnia. Cały problem sprowadza się więc do następującego: w jaki sposób szybko obliczać sumę liczb zapisanych w wierzchołkach dowolnego poddrzewa.

W tym miejscu potrzebny jest jeszcze jeden pomysł: przed wykonaniem wszystkich opisanych obliczeń wykonamy pewien wstęp, który pozwoli nam na utworzenie wymarzonego drzewa przedziałowego. Będzie nim przeszukiwanie drzewa w głąb (DFS) startujące z korzenia drzewa, podczas którego zapamiętamy dla każdego wierzchołka v czas wejścia do niego A_v i czas wyjścia z niego B_v. Zauważmy, że za pomocą tych liczb możemy stwierdzać, czy jeden wierzchołek znajduje się w poddrzewie drugiego: w jest w poddrzewie v wtedy i tylko wtedy, gdy $[A_w, B_w] \subseteq [A_v, B_v]$. W takim razie, jeśli posortujemy wierzchołki według wartości A_i, otrzymamy taką kolejność, w której każde poddrzewo jest spójnym fragmentem (przedziałem) tak uporządkowanego ciągu wierzchołków.

Pozostaje już tylko standardowe zastosowanie struktury danych, jaką jest drzewo przedziałowe: liśćmi tego drzewa będą wierzchołki oryginalnego drzewa w ustalonej przed chwilą kolejności. Potrzebne nam operacje to dodanie liczby do wartości zapisanej w liściu symbolizującym odpowiedni wierzchołek oraz sprawdzenie, czy suma wartości w spójnym fragmencie zbioru liści, odpowiadającym odpowiedniemu poddrzewu drzewa z treści zadania, jest dodatnia. Są to dokładnie operacje udostępniane przez drzewo przedziałowe (w razie wątpliwości Czytelników odsyłamy na stronę `http://was.zaa.mimuw.edu.pl/`).

Podsumujmy wykonywane operacje: $O(N \log N)$ – wstępne przetworzenie drzewa, przeszukiwanie DFS oraz sortowanie wierzchołków według wartości A_i, plus $O(Q \log N)$ – znalezienie najniższych wspólnych przodków dla wszystkich par wierzchołków z zarządzeń, plus $N + 2Q$ operacji na drzewie przedziałowym, każda w czasie $O(\log N)$. Łącznie daje to $O((N + Q) \log N)$ operacji przy zużyciu pamięci rzędu $O(N \log N)$.

Tomasz KULCZYŃSKI