Informatyczny kącik olimpijski (33): Inspektor

Tym razem weźmy pod lupę zadanie o nazwie *Inspector is coming*, pochodzące z obozu treningowego rosyjskich drużyn (autor zadania: Aleksey Tolstikov).

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.

Naturalnym podejściem jest przeglądanie, dla każdego zarządzenia j, wszystkich połączeń, które leżą na ścieżce pomiędzy u_j a v_j. Niestety, może być ich nawet $N-1$. Stąd złożoność takiego rozwiązania wyniesie $\Omega(NQ)$. W takim razie możemy dla każdego zarządzenia po prostu przeszukać całe drzewo i zaznaczyć, że połączenia na odpowiedniej ścieżce i o długościach z przedziału $[a_j, b_j]$ muszą zostać przebudowane. Daje to rozwiązanie, które rzeczywiście ma złożoność czasową $O(NQ)$, natomiast pamięciową $O(N)$.

Jak zwykle, problemem jest przejście od takiego rozwiązania do jakiegokolwiek szybszego. Istnieją co najmniej trzy różne podejścia, z których każde daje niezłe wyniki. Przedstawione poniżej jest, zdaniem autora, najefektowniejsze.

Ukorzeńmy drzewo w dowolnym wierzchołku. Zacznijmy od obserwacji, że każda ścieżka (pomiędzy u_j a v_j) rozkłada się na dwie ścieżki idące w drzewie wyłącznie w górę, odpowiednio z u_j i v_j, do $\mathrm{LCA}(u_j, v_j)$. Przez $\mathrm{LCA}(p, q)$ oznaczamy tu najniższego wspólnego przodka wierzchołków p i q (więcej o problemie LCA można poczytać w *Delcie* 9/2007, „O dwóch równoważnych problemach", oraz w IKO w *Delcie* 7/2009). Dokonujemy takiego rozkładu dla każdego zarządzenia i zapamiętujemy w wierzchołkach u_j oraz v_j, że tam rozpoczynają się ścieżki o ograniczeniu $[a_j, b_j]$ prowadzące do $\mathrm{LCA}(u_j, v_j)$.

Zastanówmy się, jak teraz sformułować warunek na to, że połączenie musi zostać przebudowane. Rozważmy połączenie o długości w z wierzchołka p do jego ojca q. Musi ono zostać przebudowane wtedy i tylko wtedy, gdy w poddrzewie wierzchołka p rozpoczyna się jakaś ścieżka z ograniczeniem takim, że $w \in [a_j, b_j]$, oraz ta ścieżka nie kończy się w poddrzewie p.

Będziemy przeszukiwać drzewo w głąb i pamiętać połączenia na ścieżce z aktualnie odwiedzanego wierzchołka do korzenia. Gdy napotykamy wierzchołek, z którego wychodzi ścieżka w górę, to musimy przejrzeć nasze połączenia i stwierdzić, które z nich mają długości z określonego przedziału i są poniżej drugiego końca tej ścieżki. Wykorzystamy do tego statyczne drzewo przedziałowe. Dla każdego przedziału bazowego postaci $[m \cdot 2^k, (m+1) \cdot 2^k)$ będziemy pamiętali strukturę danych S zawierającą połączenia, które są na ścieżce z aktualnie odwiedzanego wierzchołka do korzenia i mają długości z tego przedziału, poza połączeniami, o których wiemy już, że trzeba je przebudować. Strukturę S zaimplementujemy jako stos, z połączeniami uporządkowanymi od korzenia (dno stosu) w dół drzewa (wierzchołek stosu). Gdy przechodzimy jakimś połączeniem w dół drzewa, to dodajemy je na szczyt stosu dla $O(\log W)$ przedziałów zawierających jego długość, a gdy po pewnym czasie cofamy się tym połączeniem, dla tych samych przedziałów usuwamy je z wierzchołków stosów (chyba że z niektórych zostało już ono wcześniej usunięte). Kiedy natomiast napotykamy dolny koniec ścieżki, rozbijamy przedział $[a_j, b_j]$ na sumę $O(\log W)$ rozłącznych przedziałów bazowych i dla każdego z nich zaznaczamy, że połączenia, które są poniżej drugiego końca ścieżki (a więc kilka początkowych z wierzchu stosu), muszą zostać przebudowane, oraz zdejmujemy je ze stosów dla tych przedziałów. Takie przetworzenie jednej ścieżki może pochłonąć wiele operacji, ale co najwyżej $O(\log W + X)$, gdzie X jest liczbą zdjętych ze stosów połączeń. Wszystko, co wrzucimy na stos, zostanie jednak zdjęte co najwyżej raz, więc całość się amortyzuje.

Ostatecznie, złożoność czasowa takiego rozwiązania to: $O((N+Q)\log N)$ – wstępne przetworzenie drzewa oraz wyznaczenie wszystkich najniższych wspólnych przodków, plus $O(N \log W)$ – wrzucanie połączeń do struktury i zamortyzowane ich usuwanie oraz próby usuwania tych, które wcześniej już usunięto, plus $O(Q \log W)$ – przetwarzanie ścieżek poza usuwaniem połączeń, co łącznie daje $O((N+Q)\log NW)$, przy zużyciu pamięci $O(N \log N + W + N \log W)$. Można pozbyć się czynników $\log N$, jeśli zastosuje się optymalne algorytmy znajdowania LCA, jednak jest to raczej niepraktyczne.

Tomasz KULCZYŃSKI