Informatyczny kącik olimpijski (36): Zagadka

W tym kąciku przyjrzymy się zadaniu *Zagadka*, które pojawiło się na tegorocznych Potyczkach Algorytmicznych.

Rozważamy pewne państwo podzielone na k województw i zawierające n miast. W tym państwie jest m dróg, z których każda łączy jakąś parę miast. Naszym zadaniem jest tak wybrać stolice województw, aby każde województwo miało dokładnie jedną stolicę (będącą jednym z miast tegoż województwa), a każda z dróg miała jakąś stolicę na co najmniej jednym swym końcu. Wiadomo, że każda droga łączy dwa różne miasta, każde województwo zawiera co najmniej jedno miasto, a każde miasto należy do dokładnie jednego województwa.

Przede wszystkim zauważmy, że powyższe wymagania ograniczające możliwości wyboru stolic możemy zamienić na zestaw zdań logicznych. Dla każdego miasta i wprowadźmy zmienną logiczną S_i oznaczającą, czy dane miasto jest stolicą swojego województwa. Wówczas drogę łączącą miasta a i b możemy rozumieć jako alternatywę $S_a \vee S_b$. Województwo interpretujemy natomiast jako zbiór alternatyw postaci $\neg S_a \vee \neg S_b$ dla każdej pary różnych miast a, b z tego województwa.

W ten sposób sprowadziliśmy zadanie do wybrania dla każdego miasta i wartości zmiennej logicznej S_i, tak aby zachodziły wszystkie wymienione wyżej alternatywy. (Jeżeli znajdziemy rozwiązanie, w którym pewne województwo nie ma żadnej stolicy, to w takim przypadku możemy dowolne jego miasto uznać za jego stolicę i nie zepsuje to znalezionego rozwiązania.) Jest to klasyczny problem 2-SAT, dla którego znany jest liniowy (dokładniej: liniowo zależny od liczby zmiennych i liczby alternatyw) algorytm. W przypadku naszego zadania oznacza to rozwiązanie w czasie $O(n^2 + m)$. Nie jest to jeszcze wynik satysfakcjonujący. Aby go poprawić, zagłębimy się w działanie algorytmu rozwiązującego problem 2-SAT, po czym spróbujemy go przyspieszyć dla tego konkretnego zadania.

Zauważmy, że każdą alternatywę $x \vee y$ możemy zamienić na dwie implikacje: $\neg x \to y$ oraz $\neg y \to x$. Zbudujemy zatem graf, którego wierzchołkami są S_i oraz $\neg S_i$ dla wszystkich i, a krawędziami – implikacje powstałe ze wszystkich naszych alternatyw. Chcemy dla każdego miasta i wybrać dokładnie jeden z dwóch odpowiadających mu wierzchołków i uznać za prawdziwy, pamiętając przy tym o implikacjach. Nietrudno spostrzec, że w każdej silnie spójnej składowej takiego grafu wszystkie wierzchołki mają taką samą wartość logiczną. Jeśli utożsamimy wierzchołki należące do tych samych silnie spójnych składowych, otrzymujemy tzw. graf silnie spójnych składowych, w którym nie ma już cykli. Ten graf sortujemy topologicznie. Rozpatrujemy kolejno te silnie spójne składowe, z których wychodzą krawędzie tylko do już rozpatrzonych. Zauważmy, że jeśli x i y są w tej samej silnie spójnej składowej, to $\neg x$ i $\neg y$ także, gdyż jeśli z x można dojść krawędziami do y, to z $\neg y$ można analogicznie dojść do $\neg x$, podobnie jeśli można dojść z y do x, to można z $\neg x$ do $\neg y$ (wynika to ze sposobu konstrukcji krawędzi z alternatyw). W takim razie, podczas rozpatrywania kolejnych składowych, jeśli napotykamy taką, której komplementarna nie została jeszcze rozpatrzona, ustalamy, że wszystkie wierzchołki tej składowej będą prawdziwe, a komplementarnej – fałszywe. W ten sposób dla każdej pary literałów x, $\neg x$ zdecydujemy, który z nich jest prawdziwy, a wszystkie implikacje, a więc także alternatywy, będą spełnione (dlaczego?). Jedynym przypadkiem, gdy ta metoda nie działa, jest sytuacja, gdy dla pewnego x, x i $\neg x$ leżą w jednej silnie spójnej składowej. W takim przypadku w ogóle nie istnieje żadne rozwiązanie, gdyż implikacje są sprzeczne.

Widzimy, że główny koszt czasowy algorytmu jest generowany przez znajdowanie silnie spójnych składowych i grafu tychże oraz przez sortowanie topologiczne tego grafu. Wszystkie te operacje w standardowy sposób wykonujemy za pomocą dwóch przeszukiwań w głąb (DFS). Koszt jednego takiego przeszukiwania to $O(V + E)$, gdzie V to liczba wierzchołków, a E – liczba krawędzi grafu. V jest w naszym przypadku równe $2n$, a więc akceptowalne. Problemem jest $E = O(n^2 + m)$. Krawędzie odpowiadające niewygodnemu składnikowi $O(n^2)$ są jednak rozmieszczone regularnie wewnątrz poszczególnych województw. Możemy zatem pamiętać dla każdego wierzchołka jedynie jego „normalnych” sąsiadów, natomiast krawędzie wynikające z przynależności do województwa wyznaczać na bieżąco, w razie potrzeby. Będąc w dowolnym wierzchołku, możemy pójść w przeszukiwaniu grafu dowolną krawędzią z jego listy krawędzi lub też krawędzią z listy krawędzi województwa. Co do tych ostatnich, to jeśli zużywamy krawędź „wojewódzką” z S_a do $\neg S_b$, możemy spokojnie założyć, że krawędzie „wojewódzkie” z S_c do $\neg S_b$ dla wszystkich c są już niepotrzebne, i o nich zapomnieć. Jest tak, gdyż wierzchołek $\neg S_b$ odwiedzamy raz, idąc z S_a, a wszystkie pozostałe krawędzie do niego prowadzące już nam nie posłużą w przeszukiwaniu. Jeśli rozmiar województwa to w, to zamiast $O(w^2)$ krawędzi dla tego województwa pamiętamy tylko $O(w)$ końców tych krawędzi i każdy koniec wykorzystujemy dokładnie raz, przy pierwszej nadarzającej się okazji.

W ten sposób wykonujemy przeszukiwanie DFS w rozważanym grafie w czasie $O(n + m)$. Całe rozwiązanie ma złożoność czasową i pamięciową $O(n + m + k)$.

Tomasz KULCZYŃSKI