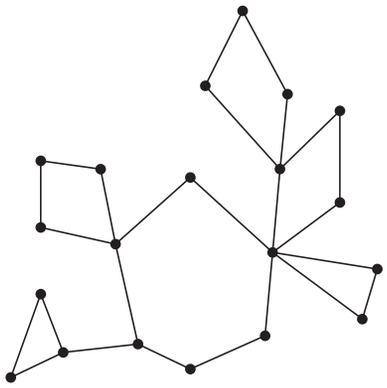
Informatyczny kącik olimpijski (8) – kaktusy

Osoba startująca regularnie w zawodach programistycznych od czasu do czasu natknie się na zadanie podobne do innego, które jakiś czas temu już rozwiązywała. Nie jest to zresztą specyfika jedynie zawodów programistycznych. Czasem jednak można mówić niemalże o modzie na dany typ zadań, gdy określony pomysł okaże się na tyle ciekawy, żeby wielokrotnie go eksploatować.



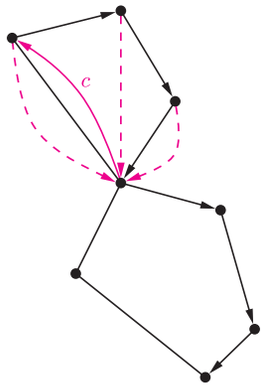
Rys. 1. Przykład kaktusa.

Kaktus jest to graf spójny, w którym każda krawędź leży na co najwyżej jednym cyklu prostym (rys. 1). W tym odcinku zmierzymy się z dwoma zadaniami:

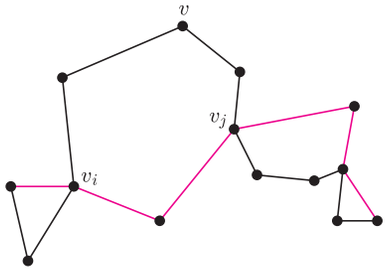
(1) znaleźć średnicę zadanego kaktusa V, czyli $\max_{v,w\in V} d(v,w)$ gdzie $d(v,w)$ to odległość między wierzchołkami v i w.

(2) zakładając, że kaktus ma n wierzchołków, znaleźć liczbę ścieżek o długościach $1, 2, \ldots, n$ w tym kaktusie. Ścieżka nie przechodzi przez żaden wierzchołek więcej niż raz.

Te dwa problemy są bardzo podobne. Wiele dynamicznych zagadnień grafowych wygodnie rozwiązuje się na drzewach – można w nich łatwo poszukiwać najkrótszych ścieżek, najdłuższych ścieżek, ścieżek określonej długości, kolorowań itp. Na naszą klasę grafów wiele z tych rozwiązań się przenosi. W przypadku problemów drzewowych korzystamy zazwyczaj z technik przechodzenia po drzewie, takich jak BFS (wszerz) czy DFS (w głąb). Przydałaby nam się w takim razie technika przechodzenia po kaktusach. Weźmy np. taki pseudokod, opisujący algorytm, który tendencyjnie nazwiemy *Rozszerzony-DFS*:



Rys. 2. W trakcie przeglądania grafu. Strzałki na krawędziach to wartości wskaźników **Ojciec**, strzałki przerywane wskazują **Korzeń**, a strzałki z literą c – **Cykl**.



Rys. 3. Jeden z etapów uaktualniania średnicy kaktusa.

Rozszerzony-DFS(v):
  Odwiedzony[v] = 1
  Dla każdego w będącego sąsiadem v:
    Jeśli **Odwiedzony**[w] = 0:
      Ojciec[w] = v
      Rozszerzony-DFS(w)
      Jeśli **Korzeń**[w] = v:
        Obsłuż cykl Cykl[v], Ojciec[Cykl[v]], Ojciec[Ojciec[Cykl[v]]], ..., v
      W przeciwnym wypadku:
        Obsłuż parę ojciec-dziecko (v, w)
        Jeśli **Korzeń**[w] ustawione:
          Korzeń[v] = **Korzeń**[w]
    Jeśli **Odwiedzony**[w] = 1 i **Ojciec**[w]$\neq v$:
      Cykl[w] = v
      Korzeń[v] = w
  Odwiedzony[v] = 2

W algorytmach drzewowych główne obliczenia odbywają się w miejscu *Obsłuż parę ojciec-dziecko* z powyższego kodu. Jak możemy wykorzystać ten schemat algorytmu do naszych problemów?

Dla szukania średnicy. Niech *Rozszerzony-DFS(v)* oblicza jednocześnie dwie wartości: $s(v)$ – średnicę *pod-kaktusa* zawieszonego w v oraz $d(v)$ – odległość do wierzchołka najbardziej odległego od v znajdującego się w tym pod-kaktusie. W funkcji będziemy mieli trzy zmienne pomocnicze: dotychczas największą znalezioną średnicę (s) pod-kaktusa zaczepionego w v, oraz odległości (d_1, d_2) do dotychczas znalezionego najodleglejszego i drugiego najodleglejszego wierzchołka w tym pod-kaktusie. Wierzchołki te muszą leżeć w różnych składowych tego pod-kaktusa.

Obsługując parę ojciec-dziecko (v, w), łatwo aktualizujemy s oraz d_1, d_2 w czasie stałym. Kiedy obsługujemy cykl $v = v_0,\ v_1,\ v_2, \ldots, v_k = w$, to aktualizując s, musimy rozważyć między innymi wszystkie możliwe ścieżki, takie jak na rysunku 3, czyli musimy wziąć pod uwagę wartość:

$$\max_{1\leqslant i<j\leqslant k} (\min(|i-j|, k-|i-j|+1) + d'(v_i) + d'(v_j)),$$

gdzie d' jest odległością do najdalszego wierzchołka spoza rozpatrywanego cyklu. Podobnie aktualizujemy długości najdłuższych ścieżek wychodzących z v. Na koniec ustalamy $s = \max(s, d_1 + d_2)$. Jeśli się postaramy, rozwiązanie zużywać będzie tylko czas liniowy względem rozmiaru kaktusa.

Po tym wprowadzeniu zadanie **szukania liczby ścieżek różnych długości** pozostawiamy Czytelnikowi.

Filip WOLSKI