LISOVSKYI STANISLAV 55106

**Dokumentacja użytkownika programu**

Uruchomiamy program, widzimy pulpit

**Text

Description automatically generated**

Klikając prawy przycisk przechodzimy do formularzu z projektem Nr 3. Widok naszego projektu:

**Graphical user interface, application, Word

Description automatically generated**

Wprowadzamy dane wejściowe: liczba wierzchołków, liczba krawędzi oraz lista sąsiedztwa

Graphical user interface, application, Word

Description automatically generated

Graphical user interface, application, Word

Description automatically generated

Po wprowadzeniu listy sąsiedztwa klikamy przycisk **Wizualizacja grafu**

Graphical user interface, application, Word

Description automatically generated

Zatem podajemy wierzchołek startu i naciskamy przycisk **Wykonaj**

Graphical user interface, application, Word

Description automatically generated

Zamiast wprowadzenia listy sąsiedztwa, możemy skorzystać z przykładowego grafu, naciskając przycisk **Przykładowa lista sąsiedztwa**, a zatem **Wizualizacja grafu**

Graphical user interface, application, Word

Description automatically generated

Podając wierzchołek startu i naciskając przycisk **Wykonaj** widzimy, jak działa nasz algorytm

Graphical user interface, application, Word

Description automatically generated

**Opis algorytmu przeszukiwania wszerz (BFS)**

**Definicja.** Graf (skierowany) to para (V,E), gdzie V jest zbiorem, a E relacją na nim.

* Zbiór V nazywamy zbiorem wierzchołków (lub węzłów) grafu.
* Parę (u,v)∈E nazywamy krawędzią z u do v.

W naszym programie reprezentujemy graf za pomocą **listy sąsiedztwa**. W tym sposobie, dla każdego wierzchołka i=1,…,n pamiętamy listę (lub zbiór, słownik) złożony z tych wierzchołków, które są sąsiadami i . Złożoność pamięciowa tego rozwiązania to Θ(|E|+|V|) , gdyż list jest tyle, ile wierzchołków, a elementów list w sumie dokładnie tyle, co krawędzi.

**Kroki algorytmu BFS**, (gdzie start - wierzchołek źródłowy):

1. Stwórz pustą kolejkę q .
2. Oznacz start jako odkryty i umieść go na końcu q .
3. Nadaj start odległość 0 .
4. Dopóki q jest niepusta:
   1. Weź wierzchołek t z początku kolejki.
   2. Niech n = odległość nadana t .
   3. Oznacz nieodwiedzone, nieodkryte sąsiady t jako odkryte i umieść je na końcu q . Nadaj im odległość n+1 . Zapamiętaj, że są sąsiadami t .
   4. Oznacz t jako odwiedzony.

**Złożoność algorytmu BFS:**

Niech G=(V,E) będzie grafem, na którym przeprowadzamy BFS (druga wersja) począwszy w t0 o którym zakładamy, że da się z niego poprowadzić ścieżkę do każdego wierzchołka V (czyli BFS odwiedzi wszystkie wierzchołki V ). Aby podać złożoność algorytmu BFS, rozważmy operacje, jakie wykonujemy na wierzchołkach. Wszystkie operacje polegające na zapamiętaniu ich poprzedników, odległości i kolorów można wykonać w czasie stałym (używając np. słowników).

Kroki 1-3 wykonują się raz: czas stały.

Kroki 4.1, 4.2 i 4.4 zajmują czas stały i wykonują się raz dla każdego odwiedzonego wierzchołka, czyli |V| razy.

Kroki podjęte dla sąsiadów wierzchołków w 4c wymagają stałego czasu. Zwracamy przy tym uwagę, że krok podjęty dla sąsiada prowadzi do jego odkrycia tylko raz. Sąsiad taki będzie jednak przetestowany (gdy sprawdzamy, czy jest odwiedzony, czy nie) za każdym razem, gdy odwiedzamy wierzchołek, z którego prowadzi do niego krawędź. Tych testów będzie więc tyle, ile krawędzi w grafie, |E| .

Stąd złożoność algorytmu to Θ(|V|+|E|) . Ponieważ zakładamy, że wszystkie wierzchołki grafu są połączone z t0 , konieczne jest, że |E|≥|V|−1 . Możemy więc - w tym przypadku - wyrazić czas działania jako Θ(|V|) .

Jeżeli nie założymy, że każdy wierzchołek w grafie jest połączony ze źródłem, podobna analiza prowadzi do ograniczenia O(|V|+|E|) .