Marcin Gecow
Robert Krajewski

Sprawozdanie 2 GIS

Temat: Badanie spójności wierzchołkowej grafu

Założenia: Graf G=(V,E) jest prosty, nieskierowany, spójny.

Opis problemu:

Spójnością grafu lub ***spójnością wierzchołkową grafu*** nazywamy taką liczbę k, że usunięcie z grafu pewnych k wierzchołków wraz z incydentnymi krawędziami spowoduje, że graf przestanie być spójny lub zredukuje go do jednego wierzchołka, ale usunięcie dowolnych k-1 wierzchołków zawsze pozostawi graf spójny.

Spójność grafu G oznaczamy κ(G).

Celem projektu jest implementacja i przetestowanie algorytmu obliczającego wartość spójności wierzchołkowej grafu - κ(G).

Metoda rozwiązania:

W algorytmie wykorzystana zostanie metoda Forda Fulkersona wyznaczająca maksymalny przepływ grafu. Zastosowana w sposób rekurencyjny pozwoli na wyznaczenie wartości spójności wierzchołkowej grafu.

Opis algorytmu:

Do rozwiązania problemu został wybrany opisany niżej algorytm A. Aby wyznaczyć maksymalny przepływ pomiędzy dwoma wierzchołkami wykorzystuje opisany algorytm Forda Fulkersona. Algorytm A jest wykonywany w sposób rekurencyjny przez algorytm B.

Algorytm Maxflow – Forda Fulkersona

Zapis metody Forda-Fulkersona w pseudokodzie:

while istnieje pewna ścieżka powiększająca $p \in G_f$ **do**

  for each $(u, v) \in p$ **do**

$$f(u,v) := f(u,v) + c_f(p)$$
$$f(v,u) := f(v,u) - c_f(p)$$

Algorytm A:

Wejście: Graf G = (V,E), oraz para nieprzylegających wierzchołków v oraz w.

Wyjście: Wartość κ(v,w)

1. Zamienić każdą krawędź xy∈E na arcs(x,y) oraz (y,x) i nazwać otrzymany graf digrafem D.
2. Dla każdego wierzchołka u, innego niż v oraz w należących do grafu G, zastąpić u dwoma nowymi wierzchołkami u_1 oraz u_2, a następnie dodać nową krawędź (u_1,u_2).

Połączyć wszystkie krawędzie, wcześniej dochodzące do u w grafie G, do wierzchołka u_1 – analogicznie krawędzie wychodzące do wierzchołka u_2 w grafie D.
3. Oznaczyć v jako wierzchołek źródłowy oraz wierzchołek w jako wierzchołek końcowy.
4. Oznaczyć wagę każdej krawędzi jako 1 i nazwać otrzymaną sieć jako H.
5. Znaleźć funkcję maksymalnego przepływu w H.
6. Ustawić κ(v,w) jako całkowity przepływ f. Koniec.

Algorytm B:
Wejście: Graf G = (V,E).
Wyjście: Wartość κ(G).

1. Przypisać i <-- 1, N <-- n-1, oraz niech $V=\{v_1,v_2,\dots,v_n\}$.
2. Dla każdego J, j=i+1, i+2, …, n,
    a. Jeżeli i > N idź do punktu 4.
    b. Jeżeli v_i oraz v_j nie są przylegające w G, wtedy oblicz $\kappa(v_i,v_j)$ używając **Algorytmu A**, i oznacz N <-- min $\{N, \kappa(v_i,v_j)\}$. Koniec pętli.
3. Przypisać i <-- i+1, później idź do punktu 2.
4. Przypisać κ(G) <-- N. Koniec.

Złożoność: O((n-δ-1)κ) wywołań max-flow.

Struktury danych:

Poszczególne elementy zostaną zrealizowane jako:
- Graf - dwuwymiarowa tablica incydencji.
- Ścieżka - wektor kolejnych wierzchołków.
- Przepustowość oraz przepływ między wierzchołkami - dwuwymiarowe macierze.

Projekt testów:

Testy zostaną podzielone na dwie kategorie: poprawnościowe oraz wydajnościowe.

Pierwszy rodzaj posłuży do weryfikacji poprawności rozwiązania. Będą one na tyle małe, żeby w prosty sposób móc się przekonać o poprawności wyniku przeprowadzając obliczenia niezależnie od algorytmu. Najprostsze z nich zostaną wygenerowane ręcznie.

Drugim rodzajem testów będą testy wydajnościowe. Ich celem będzie weryfikacja przewidywanej złożoności czasowej oraz pamięciowej. Składać się będą z generowanych automatycznie przypadków, z których część będzie losowana.

Założenia programu:

założenia wstępne:

- Graf prosty, nieskierowany, spójny.
- Dla grafu podanego na wejście istnieje rozwiązanie.

złożoność obliczeniowa:

- Algorytm Forda-Fulkersona:
  - Złożoność pamięciowa: O(V^2)
  - Złożoność czasowa: O(E* maxflow) gdzie maxflow to maksymalny przepływ
- Metoda główna (Algorytm A):
  - Złożoność pamięciowa: O(V^2)
  - Złożoność czasowa: $O(mn^{2/3})$, wymaga n(n-1)/2-m wywołań algorytmu forda fukersona

wejście:

Pierwszą linię wejścia stanowić będą dwie liczby n i m, gdzie 0 <= n <= 1000 oznacza liczbę wierzchołków a 0 <=m <= 1000 liczbę krawędzi. W każdej z kolejnych m linii wejścia będą znajdować się dwie liczby a i b takie, że 1 <= a,b <= 1000 oznaczające istnienie krawędzi od wierzchołka o numerze a do wierzchołka o numerze b.

wyjście:

Wyjście programu będzie stanowić pojedyncza liczba: wartość κ(G).

kryteria stopu:

Zostały określone w poszczególnych algorytmach.

sytuacje awaryjne:

Sytuacje awaryjne nie są przewidziane ponieważ dane wejściowe są poprawne z założenia.

Bibliografia

1. **Cormen, Thomas H., Leiserson, Charles E. i Rivest, Ronald L.** *Wprowadzenie do algorytmów.* Warszawa : WNT, 2004. 83-204-2879-3.

2. **Lipski, Witold.** *Kombinatoryka dla programistów.* Warszawa : WNT, 2004. 82-204-2968-4.

3. **Esfahanian, Abdol–Hossein.** Connectivity Algorithms. http://www.cse.msu.edu/~esfahani/book_chapter/Graph_connectivity_chapter.pdf.