**Algorytmy i Struktury Danych – Laboratoria**

Dr. Hab. Inż. Małgorzata Sterna

Informatyka, semestr 2, grupa i2, wtorki godz. 13.30

25.04.2017r.

**Ćwiczenie nr.3 – Algorytmy Grafowe**

Krzysztof Pasiewicz, 132302

Mikołaj Frankowski, 132220

**Zadanie 2**

|  |  |  |
| --- | --- | --- |
| Quant | d=0,2 | d=0,4 |
| 600 | 0,001 | 0,002 |
| 1200 | 0,003 | 0,006 |
| 1800 | 0,006 | 0,011 |
| 2400 | 0,011 | 0,019 |
| 3000 | 0,016 | 0,028 |
| 3600 | 0,022 | 0,042 |
| 4200 | 0,029 | 0,056 |
| 4800 | 0,037 | 0,087 |
| 5400 | 0,048 | 0,095 |
| 6000 | 0,062 | 0,114 |

Tab. 1 – Obliczanie etykiet w zależności od gęstości grafu

Metoda sortowania topologicznego opiera się o algorytm przeszukiwania grafu „w głąb” (DFS). Algorytm przeszukiwania DFS cechuje się złożonością O(n+m).Jego efektywność jest zależna od reprezentacji grafu na której operuje. Jako, że najczęstszą czynnością wykonywaną w trakcie pracy tego algorytmu jest poszukiwanie i przechodzenie do następników danego wierzchołka, najlepszą możliwą reprezentacją grafu jest lista następników. Najgorszą byłaby lista poprzedników, ponieważ w poszukiwaniu następników musiałaby zostać przeszukana cała struktura. Spośród analizowanych struktur najgorsza jest lista łuków, a lepsza od niej jes macierz sąsiedztwa (wykluczając przypadek w którym *m*<*n*, kiedy to lista łuków okazuje się być lepsza).

Łatwo można zauważyć, żę wraz ze wzrostem gęstości grafu, wzrósł też czas przeszukiwania grafu i obliczania etykiet czasowych wierzchołków. Wynika to z większej ilości łuków wychodzących z danego wierzchołka jakie program musi sprawdzić. Czas obliczania etykiet czasowych zależy też od samej reprezentacji grafu. Różnice w gęstości są najabrdziej widoczne dla listy łuków, ponieważ wraz ze wzrostem gęstości rośnie też rozmiar samej struktury. Podobnie, choć w mniejszym stopniu widać to na przykładzie listy następników, która także wraz z zwiększającą się ilością łuków zwiększa się rozmiar samej struktury (jednak w mnijeszym stopniu niż dla listy łuków, gdzie każdy kolejny łuk zapisywany jest jako para wierzchołków które łączy, natomiast dla listy następników jest dopisywany do danego wierzchołka jedynie dodatkowy element jako jego następnik). Najmniej widoczne byłoby to dla macierzy sąsiedztwa, gdyż jej rozmiar jest stały i nie zależy od ilości łuków (stała macierz *n* x *n*).

**Zadanie 3**

|  |  |  |
| --- | --- | --- |
| Quant | d=0,2 | d=0,4 |
| 600 | 35990 | 71986 |
| 1200 | 143903 | 288269 |
| 1800 | 324013 | 647835 |
| 2400 | 575888 | 1152178 |
| 3000 | 900526 | 1800429 |
| 3600 | 1295964 | 2594084 |
| 4200 | 1764814 | 3528386 |
| 4800 | 2304560 | 4607860 |
| 5400 | 2917516 | 5830117 |
| 6000 | 3600583 | 7199389 |

Tab. 2 – Ilość łuków powrotnych w zależności od gęstości grafu.

Wszystkie grafy ze wzgędu na tworzenie ich w sposób losowy posiadają łuki powrotne, więc żaden z wylosowanych grafów nie jest acykliczny.

Sortowanie topologiczne zostało stworzone na potrzeby skierowanych grafów acyklicznych (DAG – Directed Acyclic Graphs). Jest możliwe posortowanie tylko tych grafów, ponieważ brak cykli w grafie umożliwia ustawienie ich w takiej kolejności, że każdy wierzchołek posiadający sąsiadów znajduje się na liście przed nimi (porządek topologiczny).

**Zadanie 4**

Zależność czasu trwania etapu zliczania liczby łuków powrotnych od liczby wierzchołków n.

|  |  |  |  |
| --- | --- | --- | --- |
| **Zależność dla gęstości d=0,2** | | | |
| **Liczba wierzchołków n** | **Lista łuków(AL)** | **Macierz sąsiedztwa(M)** | **Lista następników(SL)** |
| **600** | 0,001 | 0,002 | 0,001 |
| **1200** | 0,001 | 0,009 | 0,002 |
| **1800** | 0,004 | 0,02 | 0,006 |
| **2400** | 0,007 | 0,035 | 0,01 |
| **3000** | 0,011 | 0,056 | 0,015 |
| **3600** | 0,014 | 0,079 | 0,02 |
| **4200** | 0,02 | 0,109 | 0,03 |
| **4800** | 0,026 | 0,141 | 0,037 |
| **5400** | 0,037 | 0,184 | 0,048 |
| **6000** | 0,046 | 0,239 | 0,062 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Zależność dla gęstości d=0,4** | | | |
| **Liczba wierzchołków n** | **Lista łuków(AL)** | **Macierz sąsiedztwa(M)** | **Lista następników(SL)** |
| **600** | 0,001 | 0,003 | 0,001 |
| **1200** | 0,005 | 0,017 | 0,006 |
| **1800** | 0,007 | 0,031 | 0,01 |
| **2400** | 0,013 | 0,055 | 0,018 |
| **3000** | 0,024 | 0,089 | 0,029 |
| **3600** | 0,045 | 0,131 | 0,048 |
| **4200** | 0,047 | 0173 | 0,054 |
| **4800** | 0,072 | 0,221 | 0,079 |
| **5400** | 0,072 | 0,279 | 0,096 |
| **6000** | 0,094 | 0,345 | 0,119 |

Procedura zliczania liczby łuków powrotnych analizuje zbiór wszystkich łuków (w postaci (u,v)) w danym grafie i sprawdza czy spełniają warunek begin[v]<begin[u]<end[u]<end[v], gdzie begin oznacza krok, w którym rozpoczęto analizę danego wierzchołka według algorytmu DFS a end, w którym ją zakończono.

Najlepszą ze struktur do tej procedury okazuje się lista łuków, jej złożoność obliczeniowa wynosi zaledwie O(m). Dzieje się tak, ponieważ w tej strukturze znajdują się wyłącznie istniejące łuki dzięki czemu nie trzeba analizować żadnych dodatkowych, niepotrzebnych informacji. Niewiele większą złożonością charakteryzuje się lista następników, wynosi ona O(n+m). Różnica wynika z faktu, że w tej strukturze trzeba przeanalizować wszystkie wierzchołki, niezależnie od tego czy wychodzi z nich łuk, czy nie. Zdecydowanie najwolniejszym czasem wykazuje się macierz sąsiedztwa, której złożoność wynosi aż O(). Struktura ta musi przeanalizować wszystkie swoje elementy, których jest właśnie aby wydobyć informację o tym jakie łuki znajdują się w grafie.

Gęstość grafu ma bardzo duże znaczenie dla czasu działania procedury. Im gęstszy graf tym więcej łuków do sprawdzenia a to powoduje wydłużenie czasu działania algorytmu. Nie spowoduje to jednak zmiany czasu jeżeli jako reprezentację grafu wybierzemy macierz sąsiedztwa, ponieważ niezależnie od liczby łuków trzeba w niej przejrzeć tyle samo elementów. Najbardziej wraz ze wzrostem gęstości wydłuży się czas działania dla listy łuków. Pod wpływem jej zwiększenia lista następników również znacząco zwiększy swoją złożoność.

**Zadanie 5.**

Reprezentacje grafu

Analizując różne reprezentacje grafu należy zwrócić uwagę na kilka aspektów. Pierwszym z nich jest złożoność pamięciowa. Najlepszą strukturą pod tym względem jest zdecydowanie lista łuków, jej złożoność wynosi O(m). Lista następników i poprzedników ma bardzo podobną złożoność, która wynosi O(n+m). Zdecydowanie najgorsza pod tym względem jest macierz sąsiedztwa, której złożoność wynosi O(). Różnicę między tą a pozostałymi strukturami można szczególnie odczuć kiedy graf jest rzadki. Przy grafie pełnym złożoność wszystkich reprezentacji wynosi O(). Macierz incydencji ma złożoność pamięciową O(n\*m).

Największą zaletą macierzy sąsiedztwa jest możliwość sprawdzenia czy pomiędzy dwoma wierzchołkami znajduje się łuk w liniowym czasie, dzięki bezpośredniemu odwołaniu się do konkretnego jej elementu. Pozostałe struktury są w stanie wykonać tą operację w czasie rzędu O(m) dla listy łuków i macierzy incydencji oraz O(m/n) dla list następników i poprzedników.

Wyświetlanie wszystkich łuków najlepiej sprawdza się dla listy łuków. Czas przy tej strukturze jest rzędu O(m). Dla listy poprzedników i następników złożoność wynosi O(n+m), dla macierzy sąsiedztwa O() a dla macierzy incydencji O(n\*m).

Najlepszą reprezentacją do sprawdzania następników jest, jak można się domyślić, lista następników. Jej złożoność obliczeniowa wynosi O(n). Najgorzej radzi sobie z kolei lista poprzedników, w której czas jest rzędu O(n+m). W macierzy sąsiedztwa złożoność wynosi O(n). Wynika to z konieczności przejrzenia n komórek czyli jednego wiersza. Czas działania listy łuków jest rzędu O(m) a macierzy incydencji O(m\*n). Podobna sytuacja pojawia się przy wyświetlaniu poprzedników, tym razem jednak najlepszą strukturą, o złożoności O(n), jest lista poprzedników a najgorszą, o czasie rzędu O(n+m), lista następników.