Projekt AAL – dokumentacja końcowa

Autor: Agata Kłoss

Temat projektu

W przestrzeni rozmieszczono n punktów w taki sposób, że żadne trzy z nich nie są współliniowe. Następnie każdej parze tych punktów przyporządkowano liczbę 0 lub 1. Mamy daną listę wszystkich par punktów z przyporządkowaną liczbą 0. Znaleźć liczbę wszystkich trójek punktów, dla których każda z tworzących je trzech par punktów ma przyporządkowaną tą samą liczbę. Porównać czasy obliczeń i wyniki różnych metod.

Interpretacja problemu

Problem sprowadza się do znalezienia liczby trójkątów w grafie gdzie wierzchołki to punkty w przestrzeni, a między wierzchołkami jest krawędź jeśli punkty mają przypisaną liczbę 1 oraz w odwrotności tego grafu (krawędź, jeśli punkty mają przypisaną liczbę 0). Z tego powodu metody rozwiązania są algorytmami grafowymi.

Metody rozwiązania, struktury danych i spodziewane złożoności

Wszystkie złożoności są podane względem ilości wierzchołków grafu zwanej dalej po prostu „n”.

naiwna

Algorytm: Dla każdej trójki wierzchołków sprawdź czy każda ich para ma krawędź

Struktury danych: listy

Złożoność: O(n^5)

Sprawdzenie istnienia krawędzi jest liniowe względem ilości krawędzi która maksymalnie wynosi n*(n-1)/2, a zatem ma złożoność O(n^2) – łączna złożoność O(n^2*n^3) = O(n^5)

listowa

Algorytm: Dla każdego wierzchołka utwórz zbiór jego sąsiadów – sprawdź każdą parę sąsiadów z jego zbioru – jeśli sąsiedzi mają krawędź, to cała trójka tworzy trójkąt. Po sprawdzeniu wierzchołka usuń go ze zbiorów jego sąsiadów.

Struktury danych: zbiory

Złożoność: O(n^4), ale średnio O(n^3)

Sprawdzenie istnienia krawędzi polega na sprawdzeniu istnienia elementu w zbiorze i jest w najgorszym przypadku liniowe względem ilości sąsiadów (amortyzowany najgorszy przypadek) która maksymalnie wynosi n-1 => złożoność O(n), ale średni przypadek to O(1)

wierzchołkowa

Algorytm: Podobnie jak w metodzie listowej, ale na start tworzona jest lista wierzchołków posortowanych rosnąco względem ich stopni – wybierany jest zawsze wierzchołek o najmniejszym stopniu, a po jego usunięciu lista jest aktualizowana. Aktualizacja listy polega na podmianie wartości stopnia danego wierzchołka (indeks w liście jest znany), oraz co najwyżej jednym przestawieniu dwóch elementów.

Struktury danych: zbiory, słowniki, listy

Złożoność: tak samo jak listowa - O(n^4), ale średnio O(n^3)

macierzowa

Algorytm: Utwórz macierz incydencji grafu A. Oblicz ślad z A^3, podziel przez 6.

Struktury danych: macierz

Złożoność: O(n^3) (implementacja mnożenia macierzy w Numpy)

Pomiary czasów i złożoności:

Do analizy zostały użyte złożoności średnie.

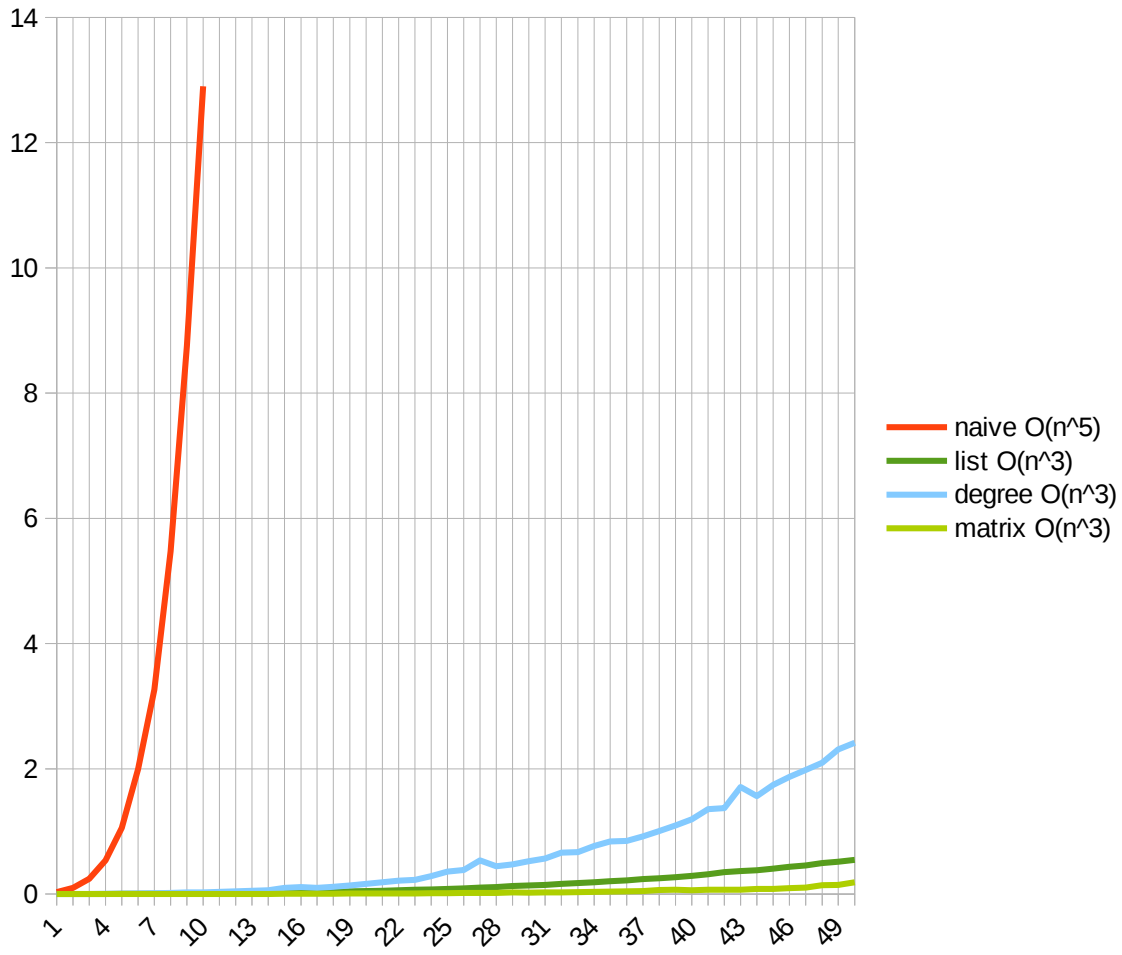
Oprócz metody naiwnej, wszystkie algorytmy mają praktycznie tę samą złożoność średnią, jednak czasy ich wykonania dla tych samych grafów różnią się.

Dla testowanych rozmiarów najlepiej wypada metoda macierzowa – dzieje się tak głównie ponieważ sama implementacja mnożenia jest napisana w C i mocno zoptymalizowana pod nowoczesne procesory, natomiast inne metody są całkowicie napisane w Pythonie.

Drugą najlepszą metodą okazała się metoda listowa, potem wierzchołkowa, a daleko z tyłu naiwna.

Różnice w czasach wykonania metod obrazuje wykres wygenerowany na podstawie testów wykonanych poleceniem ./main.py test -a [algorithm] -t random -k 50 -n 20 -d 0.5 -step 5 -r 5

(losowe grafy, o gęstości 0.5, 50 iteracji począwszy od rozmiaru 20 ze skokiem 5, 5 powtórzeń w każdej iteracji)



Oś X – numer iteracji (rozmiar = (iteracja-1)*5 + 20)

Oś Y – czas wykonania w częściach sekundy (dla systemu Linux – 1/100 sekundy)

Pełne dane testów:

naive O(n^5)

n	t(n)	q(n)
20	0,0319	0,96
25	0,0987	0,98
30	0,2455	0,98
35	0,5381	0,99
40	1,0571	1
45	1,9942	1,05
50	3,262	1,01
55	5,4733	1,05
60	8,7491	1,09
65	12,8982	1,08

list O(n^3)

n	t(n)	q(n)
20	0,0009	3,63
25	0,0015	3,13
30	0,0023	2,77
35	0,0035	2,64
40	0,0044	2,25
45	0,0065	2,36
50	0,0064	1,68
55	0,0064	1,27
60	0,0078	1,19
65	0,0112	1,33
70	0,0129	1,23
75	0,0145	1,13
80	0,0174	1,11
85	0,0203	1,09
90	0,0238	1,07
95	0,0286	1,09
100	0,0319	1,05
105	0,0367	1,04
110	0,042	1,03
115	0,049	1,06
120	0,0534	1,01
125	0,0604	1,01
130	0,0678	1,01
135	0,075	1
140	0,0837	1
145	0,0926	1
150	0,1022	0,99
155	0,1122	0,99
160	0,1275	1,02
165	0,136	0,99
170	0,1484	0,99
175	0,1616	0,99
180	0,1762	0,99
185	0,1893	0,98
190	0,2033	0,97
195	0,218	0,96
200	0,2388	0,98
205	0,2532	0,96
210	0,269	0,95
215	0,2892	0,95
220	0,3152	0,97
225	0,3498	1,01
230	0,3674	0,99
235	0,3797	0,96
240	0,4073	0,97
245	0,4341	0,97
250	0,4585	0,96
255	0,4969	0,98
260	0,5162	0,96
265	0,5477	0,96

degree O(n^3)

n	t(n)	q(n)
20	0,0011	1,06
25	0,0019	0,93
30	0,0032	0,92
35	0,005	0,9
40	0,0069	0,82
45	0,0097	0,82
50	0,0138	0,85
55	0,0175	0,81
60	0,0263	0,94
65	0,0284	0,8
70	0,0363	0,81
75	0,0437	0,8
80	0,0532	0,8
85	0,062	0,78
90	0,0995	1,05
95	0,1121	1
100	0,0995	0,76
105	0,1176	0,78
110	0,1359	0,78
115	0,1652	0,83
120	0,1892	0,84
125	0,2147	0,84
130	0,2255	0,79
135	0,2852	0,89
140	0,3572	1
145	0,3857	0,97
150	0,5368	1,22
155	0,444	0,92
160	0,4741	0,89
165	0,5243	0,9
170	0,5671	0,89
175	0,6632	0,95
180	0,672	0,89
185	0,7678	0,93
190	0,8399	0,94
195	0,8499	0,88
200	0,9217	0,89
205	1,0069	0,9
210	1,0944	0,91
215	1,1959	0,92
220	1,3541	0,98
225	1,3725	0,93
230	1,7078	1,08
235	1,5661	0,93
240	1,7451	0,97
245	1,8728	0,98
250	1,9806	0,97
255	2,0979	0,97
260	2,3095	1,01
265	2,4158	1

matrix O(n^3)

n	t(n)	q(n)
20	0,0003	6,11
25	0,0002	3,05
30	0,0003	2,36
35	0,0005	2,15
40	0,0006	1,68
45	0,0007	1,55
50	0,0009	1,44
55	0,0012	1,35
60	0,0014	1,3
65	0,0018	1,28
70	0,0021	1,18
75	0,0028	1,29
80	0,0032	1,22
85	0,0035	1,1
90	0,004	1,07
95	0,0049	1,12
100	0,0056	1,1
105	0,0065	1,1
110	0,0076	1,12
115	0,0082	1,05
120	0,0093	1,05
125	0,0103	1,03
130	0,0115	1,02
135	0,0129	1,02
140	0,0141	1
145	0,0157	1
150	0,0197	1,14
155	0,0195	1,02
160	0,0249	1,19
165	0,0239	1,04
170	0,0268	1,06
175	0,0285	1,04
180	0,0307	1,03
185	0,0352	1,08
190	0,0395	1,12
195	0,0439	1,15
200	0,0464	1,13
205	0,0643	1,46
210	0,0695	1,46
215	0,0614	1,2
220	0,0682	1,25
225	0,0682	1,17
230	0,0701	1,12
235	0,0803	1,21
240	0,084	1,18
245	0,0933	1,24
250	0,1037	1,29
255	0,14	1,65
260	0,1445	1,6
265	0,1903	1,99

Wyliczone ze wzoru wartości q(n) są dla metody naiwnej i wierzchołkowej bliskie 1. Dla metody listowej i macierzowej widoczne jest lekkie przeszacowanie – jednak porównanie ze złożonością O(n^2) dawało znacznie gorsze odchylenie w stronę niedoszacowania, co oznacza że rzeczywista złożoność jest znacznie bliższa O(n^3).

Algorytmy pomocnicze

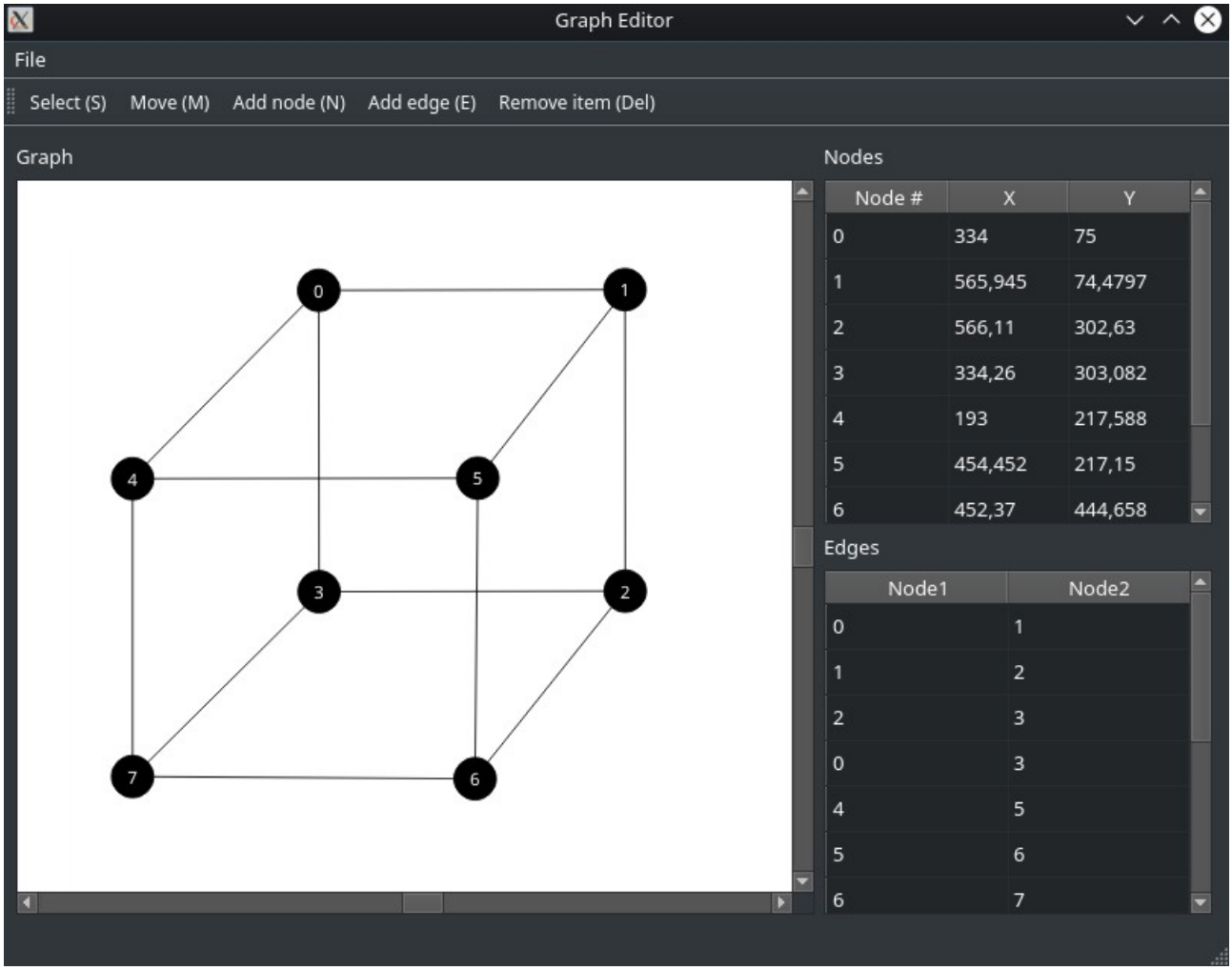
Do generacji grafu zostały napisane proste generatory różnych rodzajów grafów: losowych, pełnych, regularnych, dwudzielnych (równy podział wierzchołków) i drzew binarnych.

Do celów wizualizacji grafów zostały napisane generatory współrzędnych 2D dla wierzchołków: losowych, na okręgu, w kształcie drzewa binarnego.

Wizualizacja

Do wizualizacji generowanych grafów i ręcznego sprawdzenia poprawności wyników służy załączona aplikacja QtGraphs napisana w C++ z użyciem Qt. Pozwala ona na edycję plików grafów w tym samym formacie w jakim program wczytuje/zapisuje grafy i ręczne sprawdzenie poprawności wyników.

Przykładowy graf: 8 wierzchołków, wszystkie o stopniu 3:



Wynik działania metody listowej:

```
0 2 5
0 2 7
0 5 7
1 3 4
1 3 6
1 4 6
2 5 7
3 4 6
```

Są to wszystkie trójkąty znalezione w grafie odwrotnym – w grafie oryginalnym nie ma żadnych trójkątów. Wynik działania jest poprawny.