AOD sprawozdanie do listy 4

Bohdan Tkachenko 256630

June 8, 2023

1 Zadanie 1: Algorytm Edmondsa-Karpa

1.1 Opis Algorytmu

Algorytm Edmondsa-Karpa jest algorytmem służacym do znajdowania maksymalnego przepływu w sieci przepływowej. Jest to specyficzna implementacja metody Forda-Fulkersona, w której wybór ścieżki powiekszajacej jest dokonywany za pomoca przeszukiwania wszerz (BFS).

Dla sieci przepływowej o V wierzchołkach i E krawedziach, algorytm Edmondsa-Karpa działa w czasie $O(VE^2)$.

1.2 Wyniki Testów

Wyniki przeprowadzonych testów dla różnych wielkości problemu przedstawiono w poniższej tabeli.

Rozmiar	Maksymalny Przepływ	Czas Działania (s)
1	2	6.8e-05
2	5	8e-05
3	12	7.9e-05
4	29	9.9e-05
5	63	0.000158
6	216	0.000447
7	534	0.001509
8	903	0.003278
9	1985	0.011991
10	3215	0.024621
11	9546	0.13024
12	23536	0.440377
13	44421	1.35005
14	89573	4.07946
15	247753	18.9795
16	379519	55.1091

Table 1: Wyniki testów dla algorytmu Edmondsa-Karpa

1.3 Wnioski

Algorytm Edmondsa-Karpa jest skuteczny w znajdowaniu maksymalnego przepływu w sieciach przepływowych. Jego złożoność obliczeniowa sprawia, że jest on odpowiedni dla problemów o średniej wielkości. Dla bardzo dużych instancji problemu, czas działania algorytmu może stać sie nieakceptowalny, co wymaga zastosowania bardziej wydajnych algorytmów.

2 Zadanie 2: Algorytm Hopcrofta-Karpa w znajdowaniu maksymalnych skojarzeń

W zadaniu 2 zaimplementowano algorytm Hopcrofta-Karpa, który służy do znajdowania maksymalnego skojarzenia w grafie dwudzielnym. Program testuje działanie algorytmu dla różnych wielkości grafu i stopni wierzchołków.

2.1 Analiza wyników

Rozmiar	Stopień	Maksymalne skojarzenie	Czas działania (s)
3	1	12	5e-06
3	2	14	8e-06
4	1	22	8e-06
5	3	56	3.1e-05
6	6	126	0.0001
7	7	256	0.000147
8	8	512	0.000386
9	9	1024	0.000655
10	10	2048	0.001592

Table 2: Wyniki algorytmu Hopcrofta-Karpa dla różnych rozmiarów i stopni wierzchołków

Analizujac wyniki, można zauważyć, że maksymalne skojarzenie oraz czas działania rosna wraz ze wzrostem rozmiaru i stopnia wierzchołków w grafie. W szczególności, dla wiekszych grafów i wyższych stopni wierzchołków, liczba maksymalnych skojarzeń rośnie znaczaco.

Jednak warto zauważyć, że czas działania algorytmu, mimo że rośnie, pozostaje stosunkowo niski nawet dla dużych grafów. Oznacza to, że algorytm Hopcrofta-Karpa jest wydajny i skalowalny w praktycznych zastosowaniach.

2.2 Wnioski

Algorytm Hopcrofta-Karpa jest efektywnym narzedziem do znajdowania maksymalnych skojarzeń w grafach dwudzielnych. Jego wydajność i skalowalność sprawiaja, że jest on odpowiedni dla różnych wielkości problemów, w tym także dla dużych instancji. Dzieki temu, algorytm ten znajduje zastosowanie w wielu dziedzinach, takich jak dopasowywanie wzorców, optymalizacja i analiza sieci.

3 Zadanie 3

W zadaniu 3, analizujemy dwa modele programowania liniowego generowane przez odpowiednie funkcje. Jedna funkcja generuje model dla problemu maksymalnego przepływu w grafie, natomiast druga dla problemu maksymalnego skojarzenia.

3.1 Model dla problemu maksymalnego przepływu

Pierwsza funkcja generuje model programowania liniowego majacy na celu rozwiazanie problemu maksymalnego przepływu w grafie. Elementy modelu to:

- **Zmienne decyzyjne**: Dla każdej krawedzi (u, v) w grafie, definiujemy zmienna $x_{u,v}$ reprezentujaca przepływ w tej krawedzi. Zmienna jest ograniczona przez 0 i pojemność krawedzi $C_{u,v}$.
- **Funkcja celu**: Naszym celem jest maksymalizacja sumy przepływów wychodzacych z wierzchołka źródłowego (wierzchołek 0), czyli max $\sum_{(0,v)\in E} x_{0,v}$.
- **Ograniczenia**: Dla każdego wierzchołka v (oprócz źródła i ujścia), suma przepływów wchodzacych musi być równa sumie przepływów wychodzacych. Dla każdego v, $\sum_{(u,v)\in E} x_{u,v} = \sum_{(v,w)\in E} x_{v,w}$.

Nastepnie model jest rozwiazany za pomoca solvera GLPK, a wynik jest wyświetlany na konsoli.

3.2 Model dla problemu maksymalnego skojarzenia

Druga funkcja generuje model programowania liniowego majacy na celu rozwiazanie problemu maksymalnego skojarzenia w grafie. Elementy modelu to:

- **Zmienne decyzyjne**: Dla każdej krawedzi (u, v) w grafie, definiujemy zmienna $x_{u,v}$ reprezentujaca czy ta krawedź jest wybrana do skojarzenia. Zmienna jest ograniczona przez 0 i 1 i jest liczba całkowita.
- **Funkcja celu**: Naszym celem jest maksymalizacja sumy zmiennych decyzyjnych, czyli max $\sum_{(u,v)\in E} x_{u,v}$.
- **Ograniczenia**: Dla każdego wierzchołka u, suma zmiennych decyzyjnych dla krawedzi przyłaczonych do tego wierzchołka musi być mniejsza lub równa 1. Dla każdego u, $\sum_{(u,v)\in E} x_{u,v} \leq 1$.

Podobnie jak w pierwszym modelu, używamy solvera GLPK do rozwiazania modelu i wyświetlamy wynik.

3.3 Porównanie z algorytmami dedykowanymi

Choć modele programowania liniowego umożliwiaja rozwiazanie tych problemów, sa one zazwyczaj mniej wydajne niż dedykowane algorytmy.

Dla problemu maksymalnego przepływu, algorytmy specjalizowane sa czesto znacznie szybsze w praktyce niż rozwiazanie przez programowanie liniowe, zwłaszcza dla dużych grafów.

Dla problemu maksymalnego skojarzenia, algorytmy, takie jak algorytm Hopcrofta-Karpa, sa również bardziej efektywne.

Przyczyna jest to, że programowanie liniowe jest technika ogólna i nie wykorzystuje specjalnych własności problemu.

4 Zadanie 4: Algorytm Dinica

Algorytm Dinica jest jednym z algorytmów służacych do znajdowania maksymalnego przepływu w sieci przepływowej. Jest to algorytm iteracyjny, który działa w fazach, każda faza składa sie z dwóch kroków: budowania sieci warstwowej i wyszukiwania ścieżek powiekszajacych.

4.1 Opis algorytmu

Algorytm Dinica opiera sie na dwóch głównych krokach:

1. **Budowanie sieci warstwowej**: W tym kroku, używajac przeszukiwania wszerz (BFS) od wierzchołka źródłowego, algorytm buduje sieć warstwowa, która jest acyklicznym podgrafiem oryginalnej sieci przepływowej. Wierzchołki sa rozdzielane na poziomy, tak że każda krawedź prowadzi od poziomu i do poziomu $i + 1$.

2. **Wyszukiwanie ścieżek powiekszajacych**: W tym kroku, algorytm używa przeszukiwania w głab (DFS) do znalezienia ścieżek powiekszajacych w sieci warstwowej. Przepływ jest zwiekszany wzdłuż tych ścieżek.

Algorytm kontynuuje te dwa kroki, dopóki nie można znaleźć wiecej ścieżek powiekszajacych w sieci warstwowej. W tym momencie algorytm kończy działanie, a maksymalny przepływ jest suma przepływów wzdłuż ścieżek powiekszajacych.

4.2 Analiza złożoności

Złożoność czasowa algorytmu Dinica jest $O(V^2E)$, gdzie V to liczba wierzchołków, a E to liczba krawedzi w sieci przepływowej. Jest to jednak pesymistyczny przypadek i algorytm czesto działa znacznie szybciej na praktycznych danych.

Jedna z zalet algorytmu Dinica jest jego wydajność, szczególnie w sieciach o małej średnicy. Dla sieci o małej średnicy, liczba iteracji (faz) jest stosunkowo mała, co pozwala algorytmowi szybko znaleźć maksymalny przepływ.

4.3 Wyniki i obserwacje

Poniżej przedstawiam wyniki eksperymentu z algorytmem Dinica dla różnych wielkości sieci przepływowych:

Rozmiar	Maksymalny przepływ	Czas działania (s)
1	2	7×10^{-6}
2	5	6×10^{-6}
3	12	1.3×10^{-5}
4	29	2.1×10^{-5}
5	63	5.1×10^{-5}
6	216	0.000118
7	534	0.000366
8	903	0.000567
9	1985	0.000768
10	3215	0.001066
11	9546	0.002607
12	23536	0.005932
13	44421	0.01078
14	89573	0.013343
15	247753	0.02783
16	379519	0.077518

Z wyników można zaobserwować, że dla dużych sieci przepływowych czas działania algorytmu jest istotnie mniejszy w porownaniu do algorytma zaimplementowanego w zadaniu 1