Grzegorz Biełokozowicz WCY22IB1S4

Problem kolorowania wierzchołków grafu – algorytm LF largest first

1. **Opis teoretyczny problemu**

Problem kolorowania grafu jest jednym z fundamentalnych zagadnień teorii grafów. Polega on na przypisaniu kolorów wierzchołkom grafu w taki sposób, aby żadne dwa sąsiednie wierzchołki nie miały tego samego koloru. Optymalnym rozwiązaniem jest minimalizacja liczby użytych kolorów. Problem kolorowania grafu jest problemem NP-trudnym. Oznacza to, że nie istnieje skuteczny algorytm kolorowania grafu, który rozwiązuje problem w czasie wielomianowym, dla grafu o dowolnym rozmiarze. W związku z tym wykorzystuje się różne algorytmy heurystyczne, takie jak largest first, które nie zapewniają znalezienia optymalnego rozwiązania, ale zapewniają znalezienie dobrego rozwiązania w rozsądnym czasie.

Problem można sformułować w poniższy sposób:

Niech , będzie grafem nieskierowanym, gdzie to zbiór wierzchołków, a to zbiór krawędzi. Przypisanie koloru do wierzchołku grafu, można przedstawić jako funkcję , gdzie to zbiór dostępnych kolorów. Każdemu wierzchołkowi , przypisywany jest kolor . Warunek poprawnego pokolorowania grafu jest następujący:

Czyli żadne dwa sąsiednie wierzchołki nie mogą mieć tego samego koloru.

Liczbą chromatyczną grafu nazywana jest najmniejsza liczba , taka, że istnieje pokolorowanie grafu za pomocą kolorów i jest oznaczana symbolem . Dla dowolnego grafu o maksymalnym stopniu wierzchołka zachodzi oszacowanie

Problem kolorowania grafu jest NP.-trudny, co oznacza, że nie istnieje efektywny algorytm, znajdujący optymalne rozwiązanie w czasie wielomianowym dla wszystkich przypadków.

Przykład pokolorowania grafu, z wykorzystaniem 4 kolorów:

Obraz zawierający krąg, Wielobarwność, zrzut ekranu, sztuka

Opis wygenerowany automatycznie

Przykład optymalnego pokolorowania grafu, z wykorzystaniem 3 kolorów:

Obraz zawierający krąg, zrzut ekranu, Wielobarwność, Grafika

Opis wygenerowany automatycznie

Przykład niepoprawnego pokolorowania grafu, dwa sąsiadujące wierzchołki mają ten sam kolor.

**Obraz zawierający krąg, Grafika, Wielobarwność, zrzut ekranu

Opis wygenerowany automatycznie**

1. **Opis dokładnego algorytmu rozwiązującego problem**

Algorytm dokładny kolorowania grafu polega na rekurencyjnym przypisywaniu kolorów do wierzchołków, tak aby żaden sąsiad nie miał przypisanego tego samego koloru. Przechodząc przez każdy wierzchołek w grafie, przypisywany jest mu jeden z dostępnych kolorów i sprawdzane jest czy jest to poprawne pokolorowanie. Jeżeli tak to algorytm przechodzi do następnego wierzchołka. Jeżeli nie to algorytm przypisuje nowy kolor i sprawdza ponownie. Algorytm kończy się w momencie przypisania wszystkim wierzchołkom koloru.

Ze względu na swoją złożoność wykładniczą, używany jest tylko do małych grafów.

Opis krokowy:

Dane wejściowe:

- graf, gdzie to zbiór wierzchołków, a to zbiór krawędzi.

- liczba wierzchołków w grafie .

- liczba dostępnych kolorów.

* 1. Inicjalizacja:

Niech oznacza kolor przypisany wierzchołkowi , gdzie . Początkowo wszystkie są ustawione na "nieokreślony" lub 0.

* 1. Kolorowanie wierzchołków:

Dla każdego wierzchołka , wykonaj:

Przypisz wierzchołkowi pierwszy dostępny kolor, który nie jest używany przez żadnego sąsiada i który nie przekracza liczby dostępnych kolorów. Jeżeli nie istnieje taki kolor, zwiększ k i dodaj nowy kolor.

* 1. Zakończenie:

Po zakończeniu przypisywania kolorów dla wszystkich wierzchołków, otrzymamy wynik kolorowania grafu.

1. **Opis algorytmu heurystycznego largest first**

Algorytm heurystyczny largest first polega na przypisywaniu wierzchołkom kolorów, w kolejności stopni wierzchołków, czyli liczby sąsiadów danego wierzchołka. Zmniejsza to prawdopodobieństwo wystąpienia konfliktów.

Opis krokowy:

3.1 Inicjalizacja:

Niech oznacza kolor przypisany wierzchołkowi , gdzie Początkowo wszystkie są ustawione na "nieokreślony" lub 0.

3.2 Sortowanie wierzchołków:

* Posortuj wierzchołki grafu malejąco według ich stopni.
* Niech będzie posortowaną listą wierzchołków.

3.3 Przypisanie kolorów:

Przypisz pierwszemu wierzchołkowi z listy największy kolor

Dla każdego wierzchołka , wykonaj:

* Niech będzie zbiorem kolorów użytych przez sąsiadów wierzchołka .
* Niech kolor k będzie najmniejszym dostępnym kolorem dla , który nie należy do zbioru .
* Jeśli żaden kolor nie spełnia warunku, utwórz nowy kolor i przypisz go wierzchołkowi .
* W przeciwnym razie, przypisz kolor wierzchołkowi .

3.4 Zakończenie:

Po zakończeniu przypisywania kolorów dla wszystkich wierzchołków, otrzymamy wynik kolorowania grafu.

Różnice pomiędzy algorytmem dokładnym i heurystycznym:

1. W algorytmie dokładnym stosuje się rekurencyjne podejście, które rozważa wszystkie możliwości przypisania kolorów, aż do znalezienia poprawnego kolorowania. W przypadku algorytmu largest first, nie ma rekurencji ani wycofywania się z przypisań kolorów. Algorytm largest first dokonuje jednokrotnego przypisania kolorów, zgodnie z heurystyką wybierania największego stopnia wierzchołka.
2. Algorytm dokładny rozważa wszystkie możliwe przypisania kolorów, aż do znalezienia poprawnego kolorowania lub wyczerpania wszystkich możliwości. Algorytm heurystyczny largest first zatrzymuje się po jednym przypisaniu koloru do każdego wierzchołka, nie sprawdzając, czy istnieje lepsze rozwiązanie.
3. Ze względu na swoje dokładne podejście, algorytm kolorowania grafu może być bardziej czasochłonny dla dużych grafów. Z drugiej strony, algorytm heurystyczny largest first jest znacznie szybszy, ponieważ wykonuje tylko jedno przypisanie koloru do każdego wierzchołka.

W skrócie, algorytm dokładny stara się znaleźć optymalne kolorowanie grafu, rozważając wszystkie możliwości, podczas gdy algorytm heurystyczny largest first stosuje prostą heurystykę, która może prowadzić do suboptymalnych rozwiązań, ale działa znacznie szybciej.

1. **Oszacowanie złożoności**
   1. **Dla algorytmu dokładnego**

4.1 a) Teoretyczna pesymistyczna złożoność pamięciowa i obliczeniowa w zależności od rozmiaru zadania

Złożoność pamięciowa będzie równa .

Złożoność czasowa zależy od liczby wierzchołków, krawędzi i kolorów. Często zakłada się, że liczba krawędzi jest jakaś funkcją liczby wierzchołków i nie jest ona uwzględniania w notacji.

Złożoność czasowa będzie równa , dla najgorszego przypadku w grafie. Taki najgorszy przypadek jest wtedy, gdy każdy wierzchołek jest połączony z każdym wierzchołkiem i liczba krawędzi może wynosić .

4.1 b) Teoretyczną złożoność oczekiwaną pamięciową i obliczeniową w zależności od rozmiaru zadania

Złożoność pamięciowa nadal będzie równa .

Złożoność oczekiwana będzie się różnić w zależności od tego jak wygląda graf. Dla przeciętnych grafów, będzie ona równa .

4.1 c) Teoretyczną wrażliwość pesymistyczną w zależności od rozmiaru zadania

Pesymistyczna wrażliwość dla tego algorytmu, oznacza maksymalną liczbę zmian kolorów w tym grafie, która może wystąpić w najgorszym przypadku. Będzie ona równa .

4.1 d) Teoretyczną wrażliwość oczekiwaną w zależności od rozmiaru zadania

Zakładając, że w przeciętnym grafie, tylko połowa wierzchołków jest ze sobą połączona, to oczekiwana wrażliwość będzie wynosić .

4.1 e) Teoretyczną dokładność w zależności od rozmiaru zadania

Dokładność tego algorytmu w teorii jest równa 100%, ale niestety wiele implementacji znalezionych w internecie, a także napisanych przeze mnie, czasami zwracała nieoptymalne wyniki.

**4.2 Dla algorytmu heurystycznego largest first**

4.2 a) Teoretyczna pesymistyczna złożoność pamięciowa i obliczeniowa w zależności od rozmiaru zadania

Złożoność pesymistyczna pamięciowa będzie równa . Zależy to jednak od sposobu implementacji grafu.

W najgorszym przypadku dla każdego wierzchołka będzie trzeba sprawdzić kolor wszystkich jego sąsiadów i będzie to trzeba zrobić dla wszystkich wierzchołków. W związku z tym pesymistyczna złożoność obliczeniowa będzie równa .

4.2 b) Teoretyczną złożoność oczekiwaną pamięciową i obliczeniową w zależności od rozmiaru zadania

Złożoność pamięciowa nadal będzie równa .

Dla przeciętnych grafów, w których liczba krawędzi nie jest duża, złożoność obliczeniowa jest równa ponieważ dla każdego wierzchołka, należy przejrzeć wszystkich jego sąsiadów.

4.2 c) Teoretyczną wrażliwość pesymistyczną w zależności od rozmiaru zadania

Wrażliwość zależy głównie od liczby krawędzi i dlatego wrażliwość można określić jako . Przypisanie kolorów nie zależy wprost od liczby wierzchołków, więc nie ma ona dużego wpływu na wrażliwość algorytmu.

4.2 d) Teoretyczną wrażliwość oczekiwaną w zależności od rozmiaru zadania

Dla przeciętnych grafów wrażliwość będzie wynosić

4.2 e) Teoretyczną dokładność w zależności od rozmiaru zadania

Oszacowanie dokładności tego algorytmu jest dosyć trudne. Algorytm ten może znaleźć optymalne rozwiązanie, ale istnieją grafy, dla których przypisze on więcej kolorów niż powinien.

1. **Eksperymenty i wnioski**

Graf jest zapisywany jako słownik sąsiedztwa, gdzie klucz to wierzchołek grafu, a wartość to lista wierzchołków z którymi wybrany wierzchołek jest połączony. Na przykład taki graf:

Obraz zawierający linia, krąg

Opis wygenerowany automatycznie

Jest zapisany w poniższy sposób:

Obraz zawierający zrzut ekranu, typografia, design

Opis wygenerowany automatycznie

Warto zauważyć, że krawędzie wierzchołków są zaznaczone przy obu wierzchołkach. Na przykład dla krawędzi 1-5, dla wierzchołka 1 jest napisana krawędź do 5, a dla wierzchołka 5 jest napisana krawędź do 1.

Kolorowania także są zapisane używając słownika, gdzie kluczem jest wierzchołek, a wartością numer koloru przypisany do tego wierzchołka.

Graf jest losowo generowany przy każdym uruchomieniu. Można ustawić liczbę wierzchołków w grafie, liczbę maksymalnych krawędzi wychodzących od wierzchołka oraz to czy graf ma być pełny, czy nie.

**5.1 Testowanie złożoności pamięciowej**

Dla obu algorytmów, złożoność pamięciowa powinna zależeć tylko od liczby wierzchołków.

Testy dla 50 wierzchołków i maksimum 5 krawędzi.

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Testy dla 50 wierzchołków i maksimum 25 krawędzi.

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Testy dla 50 wierzchołków i maksimum 50 krawędzi.

**Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie**

Test dla 50 wierzchołków i grafu pełnego.

**Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie**

Jak widać, liczba bajtów się nie zmienia.

Testy dla 60 wierzchołków.

**Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie**

Testy dla 70 wierzchołków.

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Testy dla 80 wierzchołków.

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Testy dla 90 wierzchołków.

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Testy dla 100 wierzchołków.

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Jak widać, objętość pamięciowa nie rośnie liniowo. Trzeba jednak pamiętać, że inne implementacje oraz inne języki mogą mieć inne złożoności pamięciowe.

**5.2 Testowanie złożoności obliczeniowej**

**5.2.1 Testy dla grafu pełnego**

Ponieważ testy są wykonywana dla dużych grafów, przedstawienie tych grafów w sprawozdaniu nie jest możliwe.

Liczba wierzchołków równa 500. Dla mniejszej liczby wierzchołków oba algorytmu wykonują się praktycznie natychmiastowo.

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Liczba wierzchołków równa 700:

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Liczba wierzchołków równa 900:

Obraz zawierający tekst, zrzut ekranu, Czcionka, informacja

Opis wygenerowany automatycznie

Liczba wierzchołków równa 1100. Dla więcej niż 100 wierzchołków, python rzuca RecursionError

****

Żeby tego uniknąć trzeba zwiększyć limit rekurencji

sys.setrecursionlimit(3000)

**Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie**

Liczba wierzchołków równa 1300:

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Liczba wierzchołków równa 1500:

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Liczba wierzchołków równa 2000:

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Jak widać, czas potrzebny na pokolorowanie grafu algorytmem dokładnym, rośnie bardzo szybko i staje się praktycznie nieużywalny. Czas potrzebny na pokolorowanie grafu algorytmem heurystycznym zostaje jednak bardzo mały.

Testy dla większych grafów będą wykonywane tylko dla algorytmu heurystycznego.

Liczba wierzchołków równa 3000.

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Liczba wierzchołków równa 4000.

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Liczba wierzchołków równa 5000.

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Liczba wierzchołków równa 6000.

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Liczba wierzchołków równa 7000.

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Liczba wierzchołków równa 8000.

**Obraz zawierający tekst, zrzut ekranu, Czcionka, informacja

Opis wygenerowany automatycznie**

Liczba wierzchołków równa 9000.

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Liczba wierzchołków równa 10000.

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Jak widać nawet dla bardzo dużych grafów, algorytm heurystyczny wykonuje się dosyć szybko.

Dla porównania, dla 10000 wierzchołków algorytm dokładny nie był w stanie się wykonać nawet w ciągu 20 minut.

**5.3 Testowanie dokładności**

Dla większości grafów, oba algorytmu użyły takiej samej liczby kolorów. Czasami udało wygenerować się taki graf, gdzie algorytm heurystyczny wymagał więcej kolorów



Kolorowanie grafu algorytmem dokładnym:

Obraz zawierający linia, krąg

Opis wygenerowany automatycznie

Kolorowanie grafu algorytmem heurystycznym:

Obraz zawierający krąg, linia

Opis wygenerowany automatycznie

**Bibliografia**

<https://en.wikipedia.org/wiki/Graph_coloring>

<https://en.wikipedia.org/wiki/DSatur>

<https://www.geeksforgeeks.org/graph-coloring-set-2-greedy-algorithm/>

<https://www.interviewbit.com/blog/graph-coloring-problem/>

<https://www.scaler.com/topics/graph-coloring-problem/>

<https://inf.ug.edu.pl/~hanna/grafy/14_kolorowanie.pdf>

<https://eti.pg.edu.pl/documents/174618/23783336/Modele%20i%20metody%20kolorowania%20graf%C3%B3w.%20Cz%C4%99%C5%9B%C4%87%20II.pdf>

<https://bibliotekanauki.pl/articles/748571.pdf>