Porównanie dwóch algorytmów kolorowania grafów działających w czasie wielomianowym

Jakub Grzywaczewski Tymoteusz Kwieciński

March 2023

1 Abstrakt

Dokument porusza tematy związane z kolorowaniem grafów i ogólnie pojętą teorią grafów. W naszym rozważaniach będziemy używali definicji grafu G jako pary (V, E), gdzie V oznacza zbiór wierzchołków (najczęściej reprezentowanych przez liczby naturalne), natomiast E będzie zbiorem krawędzi reprezentowanych przez parę wierzchołków. Będziemy pracować nad grafami nieskierowanymi, które pokrywa powyższa definicja.

Kolorowanie grafów jest to proces przypisania do każdego wierzchołka *koloru* w taki sposób, aby dowolny wierzchołek v miał kolor różny od wierzchołków połączonych z v krawędzią (te wierzchołki nazywamy sąsiadami v).

Mówiąc ściśle, kolorowaniem grafu G nazywamy funkcję $c : V \to \mathbb{C}$ ze zbioru wierzchołków w zbiór kolorów, spełniająca tą wyżej opisaną wartość. Najczęściej dla uproszczenia zbiór kolorów jest po prostu zbiorem liczb całkowitych dodatnich.

Głównym celem algorytmów kolorujących grafy jest znalezienie kolorowania, które wykorzystuje jak najmniej kolorów.

Dla każdego grafu G istnieje *liczba chromatyczna*, definiowana jako najmniejszą liczność zbioru $\mathbb{C}$ takiego, że istnieje kolorowanie $c : V(G) \to \mathbb{C}$. Liczbę chromatyczną grafu G oznaczamy jako $\chi(G)$.

Istnieją ograniczenia na liczbę chromatyczną - zarówno z dołu, jak i z góry. Najmniejsza liczba kolorów niezbędna do właściwego pokolorowania grafu G jest nie mniejsza niż *liczba klikowa*, oznaczana jako $\omega(G)$ - czyli rozmiar największej kliki w grafie G. Innymi słowy liczba chromatyczna grafu G jest nie mniejsza niż rozmiar największego podgrafu w grafie G.

Dla grafu G, niebędącym cyklem o nieparzystej długości, ani grafem pełnym, zachodzi nierówność: $\chi(G) \leq \Delta(G)$. W pozostałych przypadkach $\chi(G) = \Delta(G) + 1$. Powyższe stwierdzenie jest treścią twierdzenia Brooksa, którego dowód autorstwa węgierskiego matematyka László Lovász'a [1] wykorzystujemy w jednym z opisanych tutaj algorytmów.

Ostatecznie otrzymujemy ograniczenia:

$$\omega(G) \leq \chi(G) \leq \Delta(G)$$

2 Wstęp

Celem niniejszego dokumentu jest opisanie przygotowania do porównania działania dwóch algorytmów kolorowania grafów - *Connected Sequential* oraz algorytmu pochodzącego z dowodu twierdzenia Brooksa (nazywanego później algorytmem *Lovasz'a*). Oba te algorytmy działają w czasie wielomianowym, co stanowi ich niezwykła zaletę - kolorowanie grafów często jest bardzo czasochłonne.

W ramach planowanego porównania poza wydajnością czasową zaimplementowanych algorytmów, porównamy również jak bardzo efektywne są dane algorytmy dla przykładowych grafów. To znaczy, sprawdzimy jakie grafy są trudne lub dość trudne dla zaimplementowanych algorytmów, a także ile kolorów wymagają poszczególne grafy.

Graf *trudny* dla danego algorytmu A kolorowania to taki graf G, który dla dowolnej implementacji algorytmu A nie zostanie pokolorowany optymalnie (to znaczy na $\chi(G)$ kolorów). Z kolei graf *dość trudny* dla danego algorytmu A to taki graf G dla którego istnieje implementacja (realizacja, czyli na przykład wyniki losowań kolejności wierzchołków itp.) algorytmu A dla którego wierzchołek G nie będzie pokolorowany optymalnie (to znaczy na $\chi(G)$ kolorów).

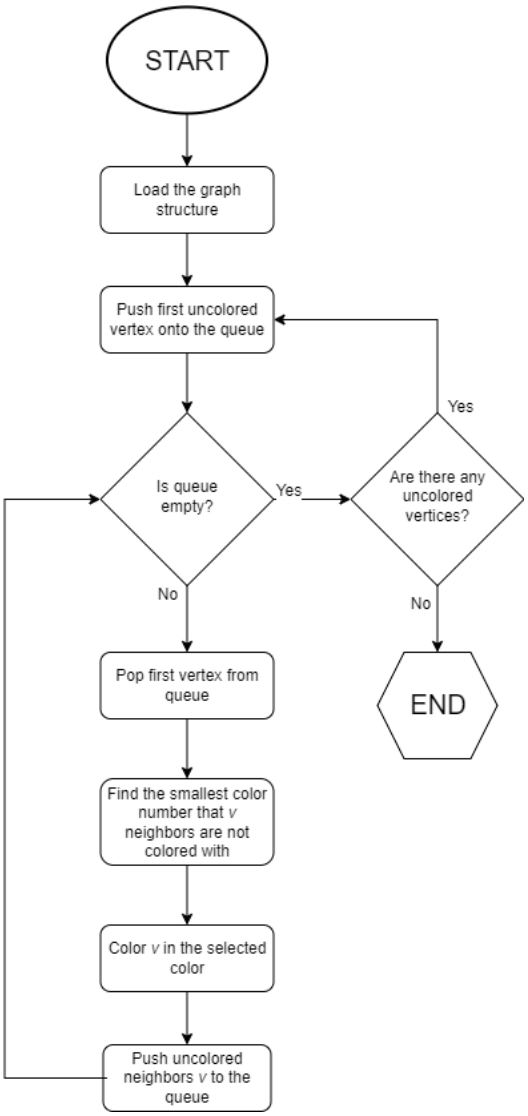
3 Connected Sequentail

3.1 Opis działania

Algorytm *Connected Sequential* jest najprostszym algorytmem jeżeli chodzi o pomysł - w każdym z kroków kolorujemy, jeszcze niepokolorowany, wierzchołek, który jest sąsiadem co najmniej jednego już pokolorowanego wierzchołka. W ten sposób pokolorowany graf jest spójny. Oczywiście pierwszy wierzchołek kolorujemy na dowolny kolor.

Ponieważ kolory interpretowane są jako liczby całkowite dodatnie, to dla każdego kolejnego wierzchołka algorytm wybiera najmniejszą liczbę całkowitą dodatnią na którą nie jest pokolorowany żaden z sąsiadów aktualnie rozważanego wierzchołka.

Przytoczona tutaj jego forma pochodzi z wykładu prowadzonego przez dr inż. Konstantego Junosza-Szaniawskiego



Rysunek 1: Schemat blokowy algorytmu *Connected sequential*

3.2 Pseudokod

Aby lepiej przybliżyć działanie tego prostego algorytmu oraz ułatwić jego późniejszą implementację, poniżej znajduje się jego działanie umieszczone w formie pseudokodu:

Load structure of graph G
Set $n = |V(G)|$
Let V_c be a set of already colored verticies

while $|V_c| \neq n$ **do**
  Create verticies queue Q with randomly choosen uncolored vertex.
  while Q is not empty **do**
    Pop the first element of the queue
    Color the vertex with the lowest avaiable color
    Add all uncolored neightbours of that vertex to the queue
  end while
end while

3.3 Kwestie techniczne

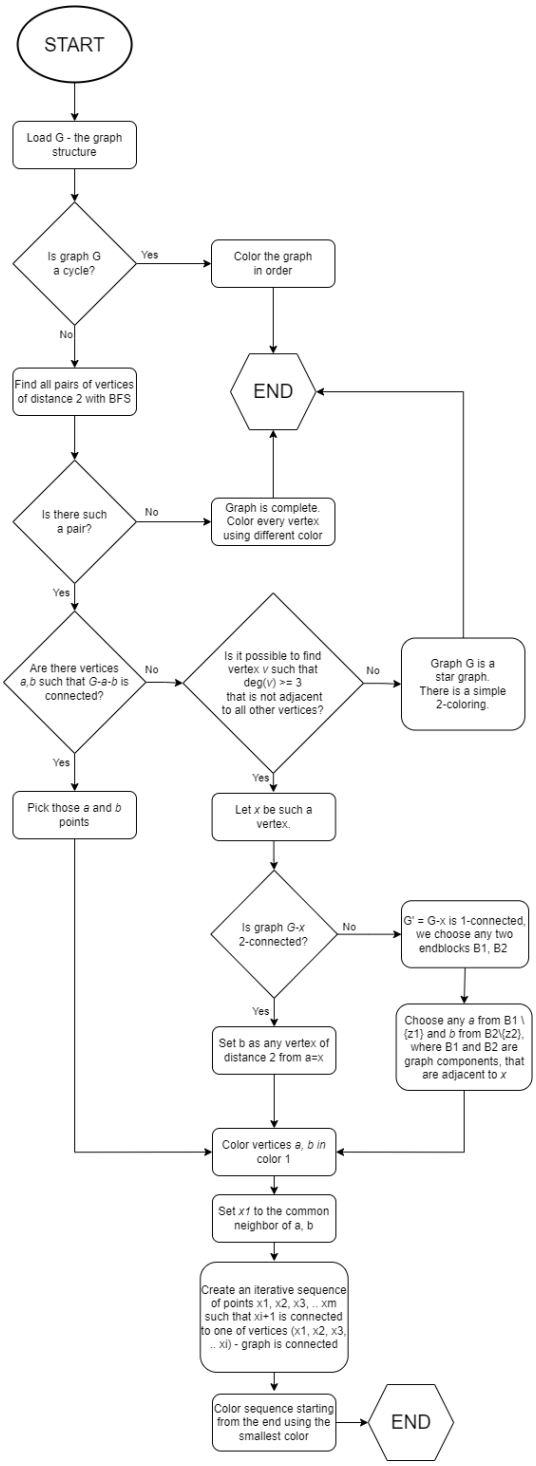
1. V_c będzie zaimplementowany jako tablica, która będzie także przechowywała informacje o kolorach wszystkich wierzchołków jako liczby.

2. Q będzie podstawową kolejką priorytetową.

4 Algorytm Lovász'a pochodzący z dowodu twierdzenia Brooksa

W tym paragrafie opiszemy algorytm kolorowania grafu G, który nie jest nieparzystym cyklem ani grafem pełnym na co najwyżej $\Delta(G)$ kolorów, a w przeciwnym wypadku na $\Delta(G) + 1$ kolorów.

Algorytm ten pochodzi z dowodu twierdzenia Brooksa autorstwa węgierskiego matematyka Lovász'a [1]



Rysunek 2: Schemat blokowy algorytmu pochodzącego z dowodu twierdzenia Brooksa

4.1 Pseudokod

Load structure of graph G
Set $m = |V(G)|$
if G is a cycle **then**
    Color the vertices in order
    Return
end if
Find all pairs of vertices of distance 2 with BFS and put them into the set S
if set S is empty **then**
    Graph G is complete
    Color every vertex in a different color
    Return
end if
if There exists a pair of vertices belonging to the set S a, b such that $G-a-b$ is connected **then**
    Pick those a, b points
else
    if It is possible to find vertex v such that $deg(v) \geq 3$ and v is not adjacent to all other vertices **then**
        Let x be such a vertex v.
        if Graph $G - x$ is connected **then**
            Set $a := x$
            Set b as any vertex of distance 2 from x
        else
            Graph $G' := G - x$ is 1-connected
            Choose any two endblocks B_1, B_2, that are adjacent to x
            Choose any vertex a from B_1 z_1 and choose any vertex b from B_2 z_2
        end if
    else
        Graph G is a star graph and there exists a simple 2-coloring
        Color G using 2 colors, starting from the vertex with greatest degree.
        Return
    end if
end if
Color vertices a, b in a color 1
Set x_1 as any common neighbor of vertices a, b
Create an iterative sequence of points $x_1, x_2, ...x_{m-2}$, from vertices $V(G-a-b)$
for i in $2, ...m-2$ **do**
    Choose any vertex x_{i+1} such that x_{i+1} is connected with any from vertices $P = (x_1, ...x_i)$
end for
# color the sequence with CS algorithm
for i in $m-2, ...1$ **do**
    Color vertex x_i in with the smallest color that any of the neighbors isn't colored in

end for
Return

4.2 Kwestie techniczne

1. Aby sprawdzić czy graf jet cyklem wystarczy policzyć stopnie wierzchołków
2. DFS (Depth First Search) może zostać użyty do sprawdzenia czy graf jest spójny w $O(n)$
3. Możemy użyć BFS (Breth First Search), aby wyznaczyć pary wierzchołków 2 odległe od siebie.
4. Kolorowanie grafu typu *gwiazda* sprowadza się do pokolorowania środka jednym kolorem, a następnie każdą następną ścieżkę naprzemiennie używając dwóch kolorów
5. Wyznaczanie wierzchołków a i b w endblokach może zostać wykonane już w fazie sprawdzania czy graf jest spójny po usunięciu wierzchołka x. Aby tego dokonać wystarczy zacząć DFS od dowolnego z sąsiadów x (nazywamy ten wierzchołek a) i potem sprawdzić czy odwiedziliśmy wszystkich sąsiadów x. Jeżeli nie odwiedziliśmy jakiegoś sąsiada ten wierzchołek staje się b.

5 Porównanie algorytmów

5.1 Oczywiste spostrzeżenia

Łatwo można zauważyć, że algorytmy te są pod pewnymi względami bardzo podobne. Drugi, bardziej zaawansowany algorytm w istocie układa wierzchołki spójnego grafu w odpowiednią sekwencję, a następnie implementuje algorytm *connected sequential* na wierzchołkach grafu w zadanej kolejności, przez co może okazać się wolniejszy.

Algorytm drugi koloruje zadany graf G wykorzystując maksymalnie $\Delta(G)$ kolorów dla grafów niebędących nieparzystymi cyklami i grafów pełnych. Algorytm pierwszy nie ma takiego ograniczenia, oczywistym jest jednak, że nie będzie wykorzystywał więcej niż $\Delta(G) + 1$ kolorów

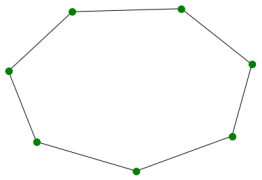
5.2 Hipotezy

1. Algorytm 1 będzie prawdopodobnie szybszy
2. Algorytm 1 będzie wykorzystywał więcej kolorów
3. Algorytm 1 będzie wymagał mniej pamięci

5.3 Pomysły na weryfikacje hipotez

Powyższe algorytmy porównamy, za pomocą ich implementacji w języku *Python*, na przykładowych grafach. Chcemy zweryfikować nasze hipotezy za pomocą testów na kilku przykładowych klasach grafów, których łatwe generowanie oferuje biblioteka *NetworkX*.

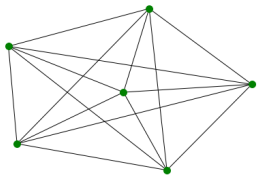
5.4 Przykładowe klasy grafów do testów



Rysunek 3: Cykl długości 7

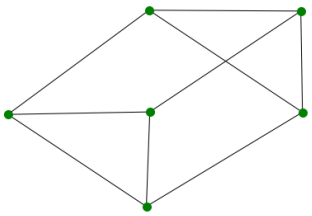
W przypadku cyklu algorytm Lovász'a powinien kolorować go na 2 lub 3 kolory w zależności od parzystości liczby jego wierzchołków.

Algorytm *CS* również powinien osiągać takie rezultaty

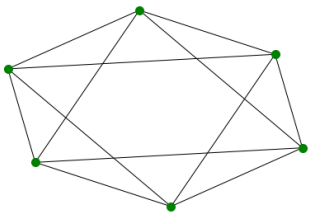


Rysunek 4: Graf pełny o 6 wierzchołkach

Graf pełen o n wierzchołkach powinien być kolorowany przez oba algorytmy na n kolorów w podobnym czasie

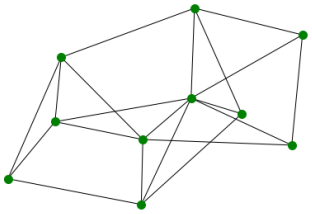


Rysunek 5: Graf regularny

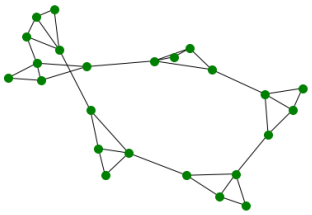


Rysunek 6: Graf Turána

Oba algorytmy zostaną również przetestowane na kilku klasycznych typach grafów, aby sprawdzić różnice w ich działaniu w powyższych przypadkach.



Rysunek 7: Losowy graf wygenerowany według algorytmu Newman Watts Strogatz



Rysunek 8: Graf typu *Caveman* - prawdopodobnie trudniejszy graf do pokolorowania dla algorytmu 2, ze względu na to, że jest 2-spójny

Poza klasycznymi typami grafów planujemy również przetestować je na różnych losowych i trudniejszych typach grafów, tak aby odkryć jakie cechy grafów decydują o różnicach w wydajności oraz liczbach użytych kolorów przez każdy z algorytmów.

Jednak ostateczny wybór przykładowych grafów użytych do testowania zaimplementowanych algorytmów będzie zależał od tego, czy istnieją dostatecznie wyraźne różnice w działaniu porównywanych algorytmów.

6 Bibliografia

Literatura

[1] L Lovász. Three short proofs in graph theory. *Journal of Combinatorial Theory, Series B*, 19(3):269–271, 1975.