Politechnika Krakowska
im. Tadeusza Kościuszki

Matematyka dyskretna

Dokumentacja zadania – zad nr 3)

Gabriela Rączka

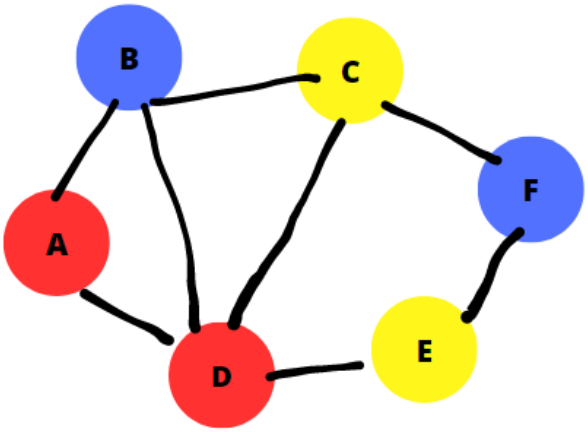
Nr albumu:147085

gr nr 02

1. Algorytm SLF kolorowania grafów.

Algorytm SLF służy do kolorowania grafów, czyli przypisywania kolorów wierzchołkom grafu tak, aby żadne dwa wierzchołki połączone krawędzią nie miały przypisanego tego samego koloru. Algorytm ten działa w następujący sposób:

1. Wybierz wierzchołek o największym stopniu spośród wierzchołków o maksymalnym stopniu nasycenia. Stopień nasycenia to liczba różnych kolorów przypisanych sąsiednim wierzchołkom.
2. Przypisz mu pierwszy wolny kolor.
3. Usuń wybrany wierzchołek z grafu i zaktualizuj stopień nasycenia i stopień wierzchołków sąsiednich.
4. Powtórz kroki 1-3 dla pozostałych wierzchołków grafu, aż wszystkie wierzchołki zostaną pokolorowane.



Na początku algorytmu wierzchołki grafu posortowane są według malejącego stopnia, co oznacza, że wierzchołek D zostaje wybrany jako pierwszy. Następnie wierzchołek C zostaje wybrany jako kolejny, ponieważ ma najwięcej sąsiadów już pokolorowanych. Wierzchołek B zostaje wybrany jako trzeci, ponieważ ma 2 sąsiadów już pokolorowanych, a wierzchołek A jako czwarty, ponieważ ma tylko jednego sąsiada już pokolorowanego. Wierzchołki E i F zostają pokolorowane na końcu, ponieważ mają tylko po jednym sąsiedzie już pokolorowanym.

Ostateczne kolorowanie grafu wygląda następująco:

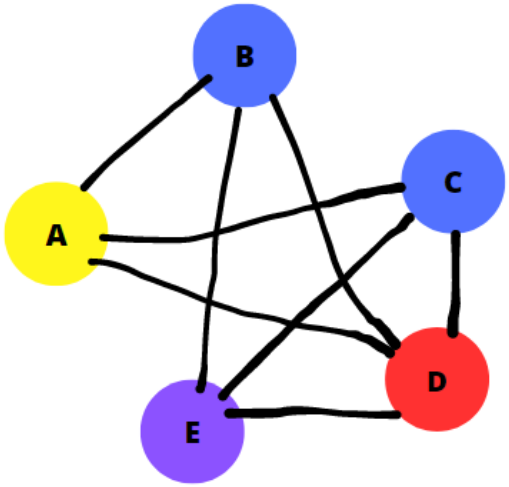
1) Wierzchołek D - kolor czerwony
2) Wierzchołek C - kolor żółty
3) Wierzchołek B - kolor niebieski
4) Wierzchołek A - kolor czerwony
5) Wierzchołek E - kolor żółty
6) Wierzchołek F - kolor niebieski

2. Algorytm heurystyczny LF kolorowania grafów.

Algorytm LF możemy podzielić na dwa, następujące etapy:

1. Uporządkowanie wierzchołków grafu malejąco według ich stopni, przy czym stopień określa liczba krawędzi incydentalnych z danym wierzchołkiem.

2. Pokolorowanie wierzchołków zachłannie, rozpoczynając od wierzchołka o najwyższym stopniu (zgodnie z uporządkowaną kolejnością). Kolorowanie zachłanne polega na przypisaniu rozpatrywanemu wierzchołkowi pierwszego dostępnego koloru.



Algorytm LF na powyższym przykładzie:

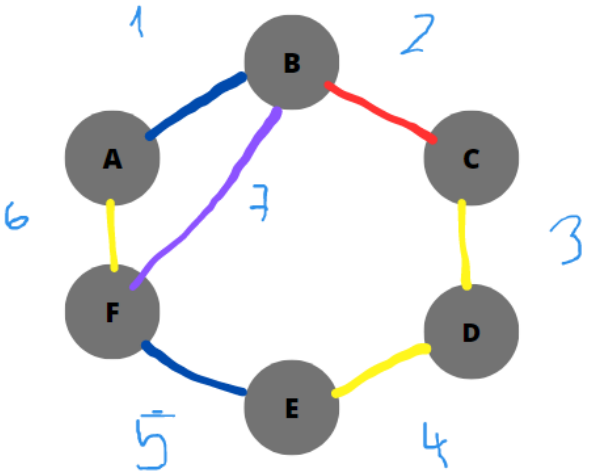
Uporządkowanie wierzchołków według ich stopni malejąco: D (stopień 4), A (stopień 3), B i C (stopień 2), E (stopień 1).

Przypisanie kolorów zachłannie: D otrzyma pierwszy dostępny kolor, czyli kolor czerwony. Następnie, A otrzyma drugi kolor czyli żółty, ponieważ kolor 1 - czerwony jest już zajęty przez wierzchołek D. Kolejno, B i C otrzymają trzeci kolor, czyli niebieski, a E otrzyma czwarty kolor, czyli kolor fioletowy.

3. Algorytm NC kolorowania krawędzi w grafie.

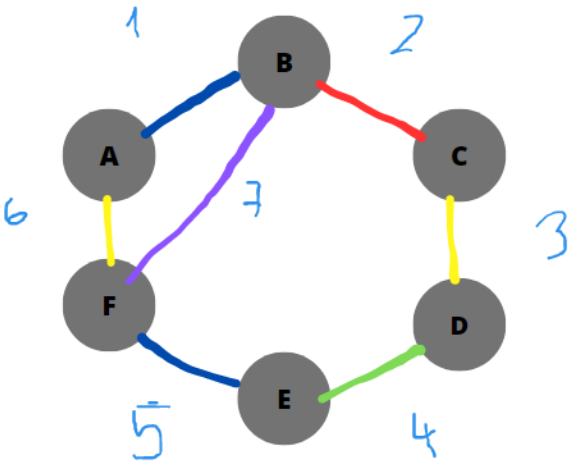
Algorytm NC kolorowania krawędzi w grafie jest algorytmem heurystycznym, który stara się przyporządkować kolory do krawędzi w grafie, tak aby sąsiednie krawędzie miały różne kolory. Algorytm ten działa na zasadzie iteracyjnej modyfikacji kolorów krawędzi w grafie. Na początku każda krawędź otrzymuje losowy kolor, a następnie algorytm sprawdza, czy w grafie nie występują sąsiednie krawędzie o tym samym kolorze. Jeśli tak, to algorytm modyfikuje kolory tych krawędzi, tak aby były one różne. Algorytm kończy pracę, gdy w grafie nie ma już sąsiednich krawędzi o tym samym kolorze.

Kolorowanie grafu przy użyciu algorytmu NC może wyglądać na przykład tak



Graf ma 6 wierzchołków oznaczonych literami A, B, C, D, E i F, oraz 7 krawędzi.

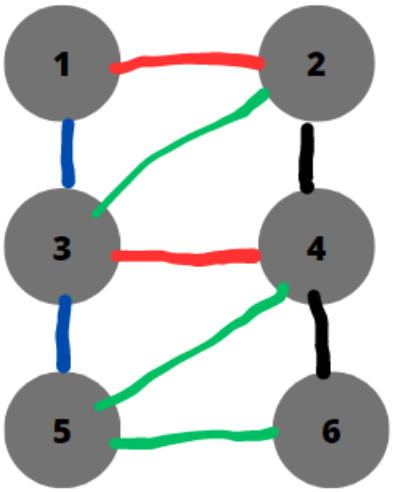
W tym przykładzie, na początku każda krawędź otrzymała losowy kolor, a następnie algorytm NC sprawdził, czy w grafie nie występują sąsiednie krawędzie o tym samym kolorze. Ponieważ w grafie pojawiły się dwie sąsiednie krawędzie o tym samym kolorze (krawędzie 3 i 4), algorytm zmodyfikował kolory tych krawędzi, tak aby były one różne. Po kilku iteracjach, w grafie nie ma już sąsiednich krawędzi o tym samym kolorze, więc algorytm kończy pracę.



4. Algorytm NTL kolorowania krawędzi.

Algorytm NTL kolorowania krawędzi w grafie działa na podobnej zasadzie co algorytm NC, ale stawia dodatkowe ograniczenia. Algorytm stara się przyporządkować kolory do krawędzi w grafie, tak aby trzy krawędzie nie tworzyły współliniowych odcinków. Na początku algorytm losowo przypisuje kolory do krawędzi, a następnie iteracyjnie poprawia kolorowanie, aż spełnione zostaną wszystkie warunki NTL.

Na przykład:



W tym przypadku, algorytm NTL pozwolił nam ustalić kolory wszystkich krawędzi wewnątrz prostokąta, korzystając z informacji o kolorach krawędzi zewnętrznych. Krawędzie zewnętrzne to krawędzie łączące wierzchołki 1-2, 2-4, 4-6 i 5-6.

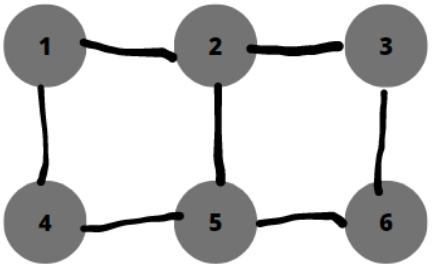
Na tej podstawie, krawędź pomiędzy wierzchołkami 3 i 4 ma kolor czerwony, krawędź pomiędzy wierzchołkami 1 i 3 ma kolor niebieski, krawędź pomiędzy wierzchołkami 4 i 5 ma kolor zielony, a krawędź pomiędzy wierzchołkami 2 i 3 ma kolor zielony.

Jeśli chodzi o zasady NTL, to należy pamiętać, że algorytm może działać tylko dla figur geometrycznych, w których wszystkie krawędzie są różnokolorowe. W tym przypadku, wszystkie krawędzie są różnokolorowe, więc warunek ten jest spełniony. Należy również pamiętać, że algorytm NTL działa tylko dla figur, które mają przynajmniej jedną krawędź boczną. W tym przypadku, prostokąt ma dwie krawędzie boczne, więc warunek ten również jest spełniony.

5. Algorytm Fleury’ego znajdowania cyklu lub ściezki Eulera.

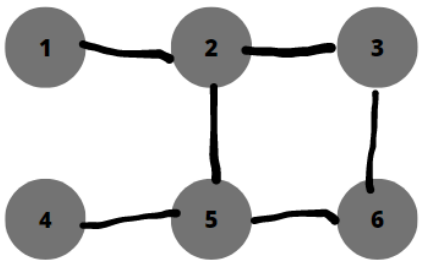
Algorytm Fleury'ego służy do znajdowania cyklu lub ścieżki Eulera w grafie nieskierowanym, czyli takim, w którym każde wierzchołki sąsiadujące z danym wierzchołkiem są połączone krawędzią. Algorytm działa w oparciu o regułę, że jeśli w grafie istnieją wierzchołki o nieparzystych stopniach, to nie istnieje w nim cykl Eulera, ale istnieje ścieżka Eulera, która zaczyna się i kończy w tych wierzchołkach. Algorytm Fleury'ego iteracyjnie usuwa krawędzie grafu, zachowując przy tym spójność grafu, i sprawdza, czy usunięcie danej krawędzi nie spowoduje utworzenia w grafie wierzchołka o nieparzystym stopniu. Jeśli tak, to krawędź ta zostaje usunięta, a jej koniec jest dodawany do ścieżki lub cyklu.

Przykład:

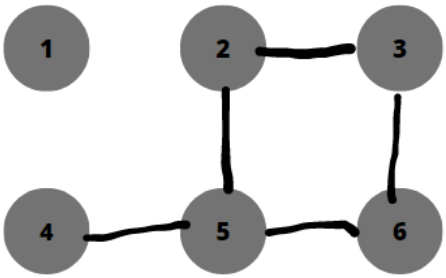


Aby zastosować algorytm Fleury'ego, wybieramy jeden z wierzchołków startowych. W tym przykładzie, wybierzemy wierzchołek 1.

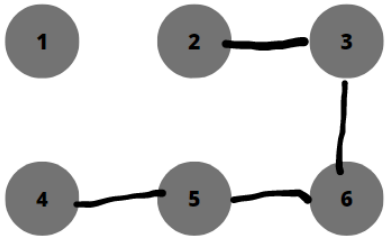
1) Wybieramy krawędź 1-4 i usuwamy ją. Graf pozostaje spójny.



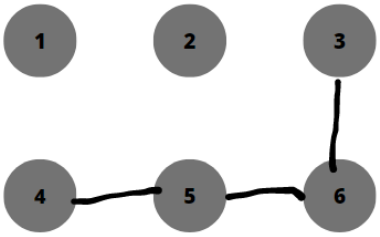
2) Wybieramy krawędź 1-2 i usuwamy ją. Graf pozostaje spójny.



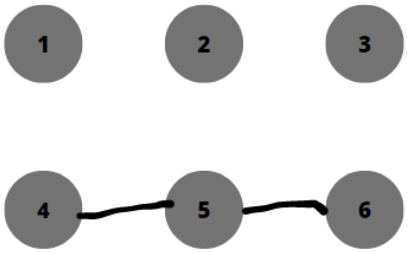
3) Wybieramy krawędź 2-5 i usuwamy ją. Graf pozostaje spójny.



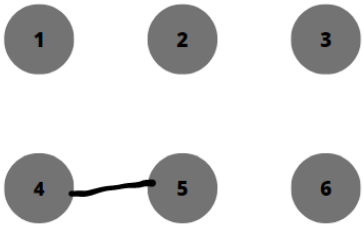
4) Wybieramy krawędź 2-3 i usuwamy ją. Graf pozostaje spójny.



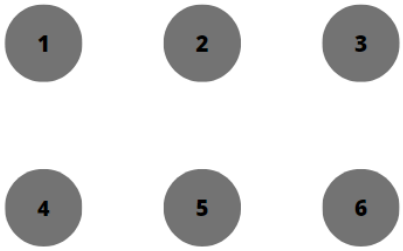
5) Wybieramy krawędź 3-6 i usuwamy ją. Graf pozostaje spójny.



6) Wybieramy krawędź 6-5 i usuwamy ją. Graf pozostaje spójny.



7) Wybieramy krawędź 5-4 i usuwamy ją. Graf pozostaje spójny.

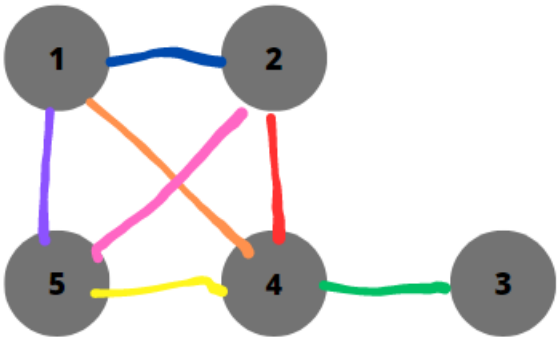


W wyniku powyższych kroków otrzymaliśmy cykl Eulera przechodzący przez wszystkie krawędzie grafu: 1-4-1-2-5-2-3-6-5-6-4-5.

6. Algorytm Hierholzera znajdowania cyklu Eulera.

Algorytm Hierholzera służy do znajdowania cyklu Eulera w grafie skierowanym lub nieskierowanym. Algorytm ten działa iteracyjnie, wybierając wierzchołek startowy i tworząc ścieżkę, która przechodzi przez każdą krawędź grafu dokładnie raz. Następnie algorytm znajduje cykle wychodzące z końcowego wierzchołka ścieżki i dołącza je do niej, tworząc w ten sposób kolejne fragmenty cyklu. Proces ten powtarza się, aż cały graf zostanie pokryty cyklem Eulera.

Przykład:

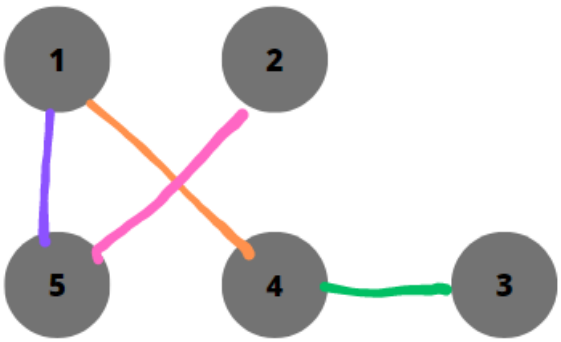


Graf składa się z pięciu wierzchołków i siedmiu krawędzi.

Początkowo, aby zastosować algorytm, wybiera się dowolny wierzchołek startowy. W naszym przypadku, wybierzemy wierzchołek 1.

1) Przechodzimy krawędzią niebieską do wierzchołka 2 i usuwamy tę krawędź.
2) Idziemy krawędzią czerwoną do wierzchołka 4 i usuwamy tę krawędź.
3) Wybieramy krawędź żółtą 4-5, aby połączyć te dwa cykle i usuwamy ją.

Teraz mamy jeden cykl: { 1, 2, 4, 3, 5, 1}.



4) Wybieramy krawędź fioletową 1-5, aby usunąć ostatni most, i usuwamy ją.

Teraz mamy jeden cykl: {1, 5, 4, 3, 2, 1}.

5) Wybieramy krawędź pomarańczową 1-4, aby usunąć ostatni most, i usuwamy ją.

6) Idziemy krawędzią zieloną 4-3 do wierzchołka 3 i usuwamy tę krawędź.

7) Wracamy do punktu 3), ponieważ mamy jeszcze nieodwiedzone krawędzie. Wybieramy krawędź różową 2-5, aby połączyć te dwa cykle i usuwamy ją.

Teraz mamy jeden cykl Eulera: {1, 5, 2, 4, 3, 4, 5, 1}.

Warto zauważyć, że algorytm Hierholzera działa tylko na grafach eulerowskich, czyli takich, w których istnieje cykl Eulera, czyli cykl przechodzący przez każdą krawędź grafu dokładnie raz.

Poniżej przedstawiony jest pseudokod algorytmu Hierholzera:

1. Wybierz dowolny wierzchołek v grafu G i umieść go na stosie S.

2. Utwórz pusty ciąg C.

3. Dopóki stos S nie jest pusty, wykonuj:

a. Pobierz wierzchołek u ze szczytu stosu S, ale nie usuwaj go ze stosu.

b. Jeśli u ma nieodwiedzone sąsiadujące wierzchołki v, to wybierz dowolny taki wierzchołek w i dodaj go na stos S. Usuń krawędź (u, w) z grafu G.

c. Jeśli u nie ma nieodwiedzonych sąsiadujących wierzchołków, to usuń u ze stosu S i dodaj u do ciągu C.

4. Odwróć kolejność elementów w ciągu C i zwróć ten ciąg jako cykl Eulera.

7. Algorytm mrówkowy wyszukiwania ścieżki Hamiltona.

Algorytm mrówkowy to metaheurystyczny algorytm przeszukiwania przestrzeni rozwiązań, który wykorzystuje zachowania kolonii mrówek. Algorytm ten może być stosowany do rozwiązywania różnych problemów optymalizacyjnych, w tym problemu znajdowania ścieżki Hamiltona w grafie. Algorytm mrówkowy polega na symulowaniu zachowań mrówek, które poruszają się po grafie i pozostawiają feromony na odwiedzonych przez siebie krawędziach. Na podstawie poziomu feromonów mrówki wybierają kolejne krawędzie, które chcą odwiedzić. Im większy poziom feromonów na danej krawędzi, tym większa szansa, że mrówka wybierze ją do dalszego przemieszczania się po grafie. Algorytm kończy działanie, gdy wszystkie mrówki zakończą przeszukiwanie grafu, a otrzymany wynik jest optymalizowany poprzez aktualizację poziomów feromonów na odwiedzonych krawędziach.

Pseudokod algorytmu mrówkowego wyszukiwania ścieżki Hamiltona:

- **rho** to współczynnik parowania feromonów
- **delta_T(i, j)** to ilość feromonu pozostawiona przez mrówkę ant na krawędzi (i, j)
- **Q** to stała regulująca ilość feromonu pozostawianego przez mrówkę ant
- **L(P)** to długość ścieżki P mrówki ant

1. Inicjalizacja

   a. Utwórz graf G(V, E).

   b. Zainicjuj macierz feromonową T taką, że T(i, j) = T0 dla każdej pary wierzchołków i, j.

   c. Zainicjuj tablicę mrówek A o długości m.

   d. Dla każdej mrówki a w A: 1. Ustaw pozycję początkową mrówki a na losowy wierzchołek v. 2. Wyczyść ścieżkę P mrówki a.

2. Pętla główna

   a. Dla każdej mrówki a w A wykonaj następujące kroki:
      1). Wybierz kolejny wierzchołek x, który nie został odwiedzony przez mrówkę a i spełnia warunek widoczności.
      2) Zaktualizuj ścieżkę P mrówki a dodając wierzchołek x.
      3) Zaktualizuj feromon na każdej krawędzi w ścieżce P mrówki a według wzoru: T(i, j) = (1 - rho) * T(i, j) + delta_T(i, j) gdzie: - rho to współczynnik parowania feromonów - delta_T(i, j) to ilość feromonu pozostawiona przez mrówkę a na krawędzi (i, j) - Q to stała regulująca ilość feromonu pozostawianego przez mrówkę a - L(P) to długość ścieżki P mrówki a
      4) Jeśli wszystkie wierzchołki zostały odwiedzone, wróć do pozycji początkowej i zakończ ścieżkę.

   b. Zwiększ poziom feromonów na każdej krawędzi grafu G o wartość delta_T(i, j) dla każdej mrówki a.

   c. Zmniejsz poziom feromonów na każdej krawędzi grafu G o wartość rho * T(i, j) dla każdej mrówki a.

   d. Zaktualizuj najlepszą ścieżkę H jako najkrótszą ze wszystkich ścieżek P dla każdej mrówki a.

3. Powtarzaj kroki 2-3 do momentu osiągnięcia warunku stopu (np. określonej liczby iteracji lub wystarczającej jakości rozwiązania).

4. Zwróć najlepszą znalezioną ścieżkę H.