

Algorytm szukający minimalnego drzewa rozpinającego

laboratorium - zadanie 3

Reichel Bartosz

reichel@mif.pg.gda.pl

27 listopada 2015

1 Opis

Utworzyć program, który jako dane wejściowe przyjmuje listę krawędzi wraz z wagami oraz wierzchołki grafu, a następnie na tej podstawie poszukuje minimalnego drzewa rozpinającego (patrz protokół STP - Spanning Tree Protocol [1]) wedle dwóch wybranych algorytmów (algorytmu Kruskala, Prima, Borůvky [2]). Jako dane wyjściowe podaje listę krawędzi, które pozostały i tworzą drzewo oraz koszt całego drzewa. Porównać wydajność implementacji algorytmów. Dla zainteresowanych (lub chcących zdobyć więcej punktów) istnieje możliwość zaimplementowania innych algorytmów tego typu (lub większej ich liczby) i ich porównanie. Również wykorzystanie bibliotek, własna implementacja pozwalająca na rysowanie grafów zostanie doceniona.

Zachęcam do wykorzystania biblioteki STL w przypadku osób chcących wykorzystać C++ (ot chociażby sortowanie).

Poniżej przykład.

2 Algorytm

Aby zrozumieć opis algorytmu podam kilka terminów oraz ich wyjaśnień. Po szczegółowe informacje/definicje odsyłam do dowolnej literatury z zakresu teorii grafów jak chociażby [3]

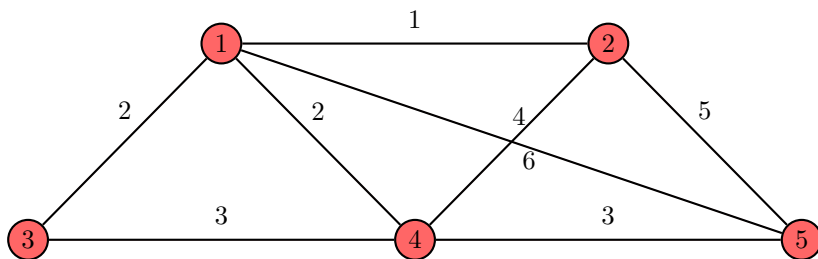
Cykl w grafie to droga zamknięta. Oznacza to, że wierzchołek początkowy jest też końcowym.

Drzewo to graf, który jest acykliczny i spójny. Oznacza to, że z dowolnego jego wierzchołka można dotrzeć do każdego innego (spójność) oraz że droga prowadząca do niego jest tylko jedna (acykliczność).

Drzewo rozpinające grafu G nazywamy drzewo zawierające wszystkie wierzchołki grafu G . Aby skonstruować drzewo rozpinające w grafie G należy usunąć z niego wszystkie krawędzie które należą do cykli.

W przedstawionym tu zadaniu należy znaleźć minimalne drzewo rozpinające. Oznacza to, że w wybranym grafie (którego model może przedstawiać jako np. połączone komputery za pomocą różnych router'ów, switch'y etc.) należy znaleźć połączenie wierzchołków grafu w taki sposób aby drzewo rozpinające było minimalne. W opisanym przypadku należy wykonać to wykorzystując algorytm Kruskala [2].

Rozpatrzmy graf przedstawiony poniżej:



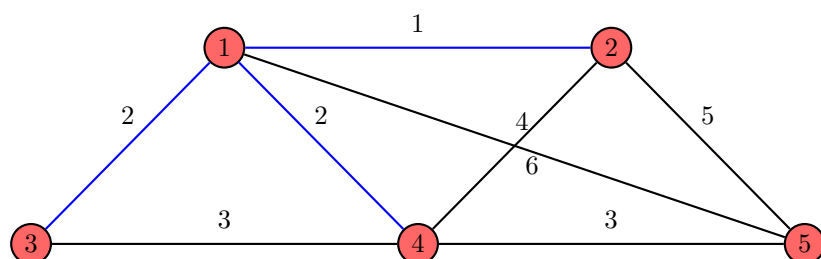
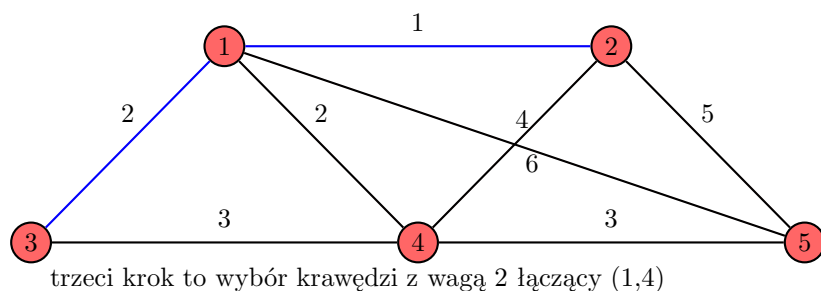
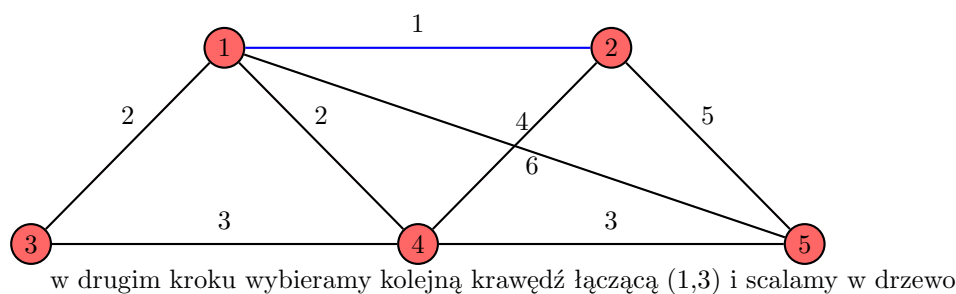
posiada on pięć wierzchołków oraz osiem krawędzi. Numery nad/pod krawędziami wskazują na wagę krawędzi, można ją utożsamiać na przykład z długością połączenia. Poszukiwanie minimalnego drzewa rozpinającego będzie polegało na odszukaniu drzewa rozpinającego, którego wagi będą najmniejsze.

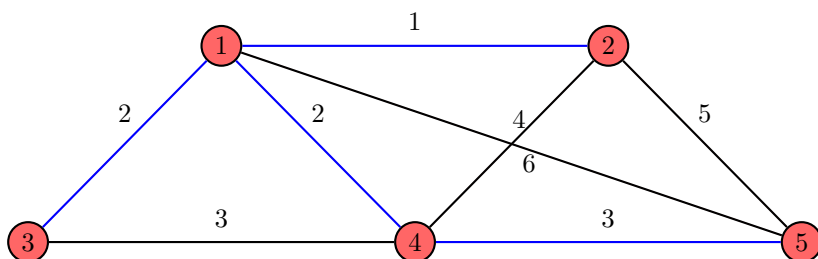
Sam algorytm Kruskala (wraz z uwagami dotyczącymi implementacji) można zapisać w następujących punktach (jako wejście graf G):

1. Utwórz las L z wierzchołków grafu, gdzie każdy wierzchołek tworzy osobne drzewo (w najprostszym zrozumieniu- implementacji jest to lista wierzchołków $1,2,\dots$).
2. Utwórz zbiór K zawierający wszystkie krawędzie grafu G wraz z wagami.
3. Dopóki K nie jest pusty (dla uproszczenia można go przed posortować po wagach):
 - (a) Usuń z K krawędź o najmniejszej wadze (w przypadku posortowanej listy pierwszy element).
 - (b) Jeśli krawędź ta łączyła dwa różne drzewa dodaj ją do lasu L aby połączyła dwa odpowiednie drzewa w jedno (można ją po prostu zachować z informacją co łączy w liście)
 - (c) Jeśli jest inaczej usuń ją.
4. Jako rezultat zwróć las L który jest minimalnym drzewem rozpinającym.

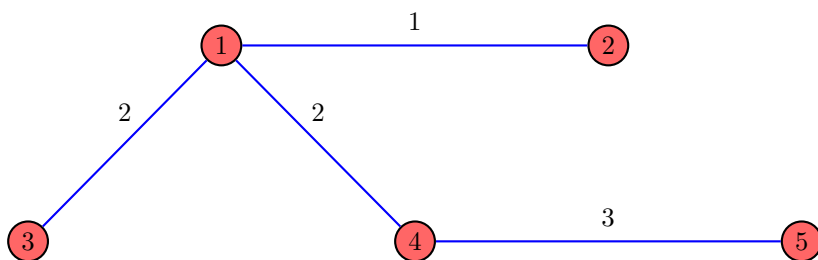
Rozpatrzmy teraz kolejne kroki algorytmu

W pierwszym kroku krawędź łącząca $(1,2)$ jest dodana do lasu, teraz drzewo 1 i 2 jest jednym drzewem.





Pozostałe krawędzie już łączą wierzchołki tego samego drzewa, więc są pomijane. Ostatecznie otrzymujemy minimalne drzewo rozpinające w postaci (z kosztem 8):



Literatura

- [1] E. Decker P. Langille, A. Rijsinghani, K. McCloghrie, Definitions of Managed Objects for Bridges RFC 1493, 1993, <http://www.rfc-editor.org/rfc/rfc1493.txt>
- [2] Cormen T.H., Leiserson Ch.E., Rivest R.L., Stein C. *Wprowadzenie do algorytmów*, WNT, Warszawa, 2001
- [3] R.J. Wilson *Wprowadzenie do teorii grafów* PWN, Warszawa, 2000