**Algorytm Kruskala** jest to algorytm, który służy do znajdowania najkrótszego drzewa rozpinającego dla grafu nieskierowanego, ważonego. Składa się on z trzech etapów:

* utworzenia listy Kruskala o następującym formacie: wierzchołek pierwszy, wierzchołek drugi, waga krawędzi pomiędzy nimi. Lista ma tyle pozycji ile jest krawędzi czyli (n-1)\*n/2 gdzie n to liczba wierzchołków
* posortowanie listy rosnąco względem kolumny trzeciej, która przechowuje wagi krawędzi
* dołączeniu kolejnych pozycji listy Kruskala do macierzy wynikowej pod warunkiem, że dodanie danej krawędzi nie spowoduje utworzenia cyklu. Macierz wynikowa ma n-1 pozycji.

Algorytm Kruskala jest algorytmem zachłannym, ponieważ wybór krawędzi do dodania odbywa się bez weryfikacji pozostałych połączeń, czyli zostaje dodane optimum lokalne. Graniczna złożoność obliczeniowa czasowa tego algorytmu to θ .

**Pseudokod:**

Dane wejściowe:

n - liczba wierzchołków w sieci

lista – lista krawędzi sieci, ma 3 kolumny i (n-1) \* n /2 wierszy, pierwsza i druga kolumna to połączone ze sobą wierzchołki, a trzecia to waga krawędzi między nim

Wykonywanie Algorytmu Kruskala:

dlListy:= (n – 1) \* n / 2

lista = posortować listę rosnąco względem trzeciej kolumny

listaWynikowa[n - 1][3]

roots[n]

roots[1]:= wypełnij każdą komórkę numerem kolumny

iListy:= 1

iWynikowy:= 1

WHILE iWynikowy <= n - 1 && iListy <= dlListy DO

If roots [lista[iListy][1]] <> roots [lista[iListy][2]] THEN   
 listaWynikowa[iWynikowy] = lista[iListy]  
 root2 = roots[lista[ilisty][2]]

FOR i:= 1 TO n DO

IF roots [i] == root2 THEN  
 roots[i] = roots[lista[iListy][1]]

i:= i + 1

iWynikowy:= iWynikowy + 1

iListy:= iListy + 1

if iWynikowy < n - 1 THEN

wypisz("Siec jest niespójna")

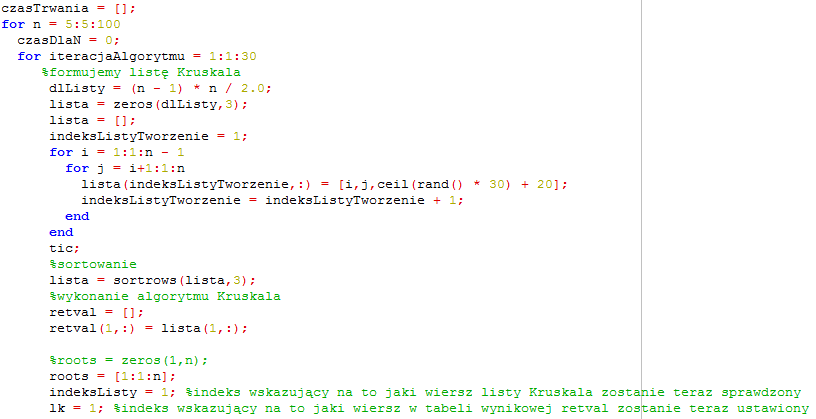
**Złożoność obliczeniowa algorytmu:**

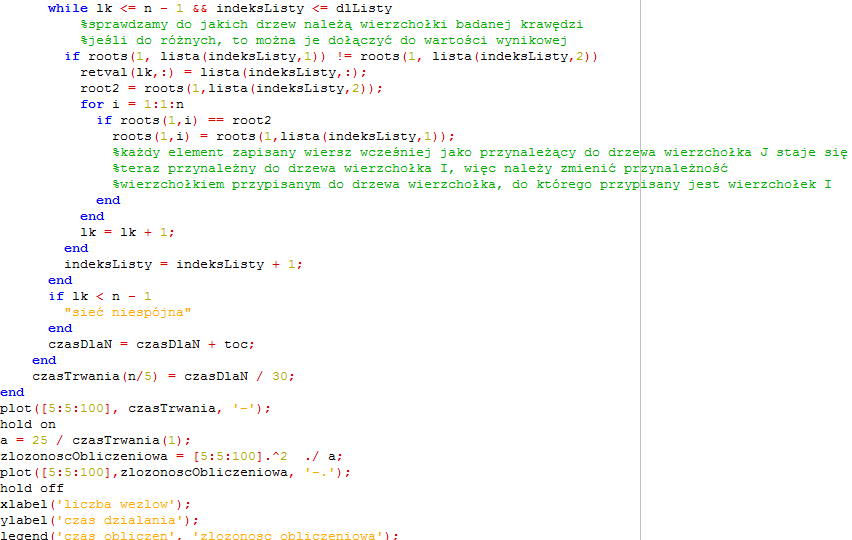
Sieć o ilości wierzchołków n może mieć liczbę krawędzi od do , więc moc zbioru krawędzi Zakładamy, że sieć jest grafem prostym, spójnym i nieskierowanym. Dla najmniejszej ilości krawędzi , główna pętla programu wykona się n – 1 razy, gdyż spójna, prosta, nieskierowana sieć o minimalnej wartości krawędzi jest już swym najkrótszym drzewem rozpinającym. Z tego powodu przy każdej iteracji pętli zostanie spełniony warunek niewprowadzenia cykliczności do drzewa. Spełnienie tego warunku sprawia, że zostanie wywołana pętla odpowiadająca za unię drzew chwilowych, która to jest iterowana n razy. Z tego wynika, że optymistyczna złożoność czasowa wynosi:

Zakładając teoretycznie, że macierz ma maksymalną ilość krawędzi i przez niefortunny układ wag ostatnia krawędź dołączona do drzewa rozpinającego znajduje się na ostatniej pozycji posortowanej listy Kruskala, pętla główna wykona się razy. Warunek o niewprowadzeniu cykliczności do drzewa zostanie jednak spełniony tylko razy, gdyż tylko tyle krawędzi może mieć drzewo rozpinające. Przy spełnieniu tego warunku zostanie wywołana pętla odpowiadająca za unię drzew chwilowych, która to jest iterowana n razy. To oznacza, że razy zostanie wywoła pętla główna pociągając za sobą wywołanie pętli odpowiadającej za unię drzew i pozostałą ilość razy, czyli: razy zostanie wywołana pętla główna bez wywoływania pętli do wykonywania unii drzew chwilowych. Z tego wynika, że teoretyczna pesymistyczna złożoność obliczeniowa czasowa wynosi:

Zarówno złożoność obliczeniowa czasowa optymistyczna, jak i pesymistyczna jest wielomianem stopnia drugiego, dlatego też złożoność obliczeniowa algorytmu Kruskala wynosi θ .

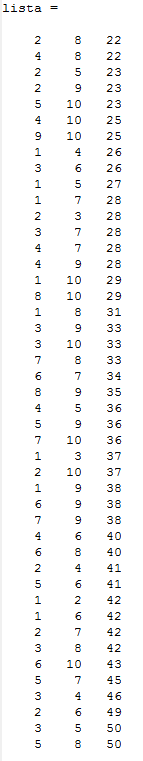
**Skrypt generujący funkcję złożoności obliczeniowej:**



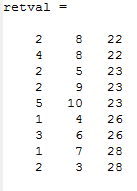


**Sprawdzenie poprawności działania algorytmu:**

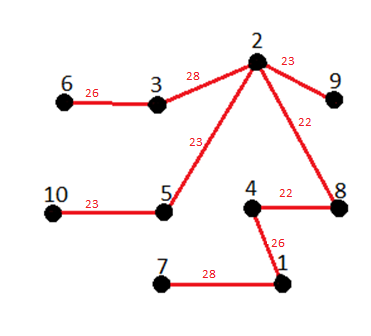
W zmiennej „lista” jest przechowywana wygenerowana sieć w formie listy Kruskala. Kolumna pierwsza i druga przechowują wierzchołki połączone ze sobą krawędzią, natomiast kolumna trzecia przechowuje wagę tych krawędzi. Sieć ta zawiera w sobie 10 wierzchołków i, ponieważ każdy wierzchołek jest połączony z każdym i jest połączony tylko raz, posiada 45 krawędzi zgodnie ze wzorem: gdzie n, to ilość wierzchołków.



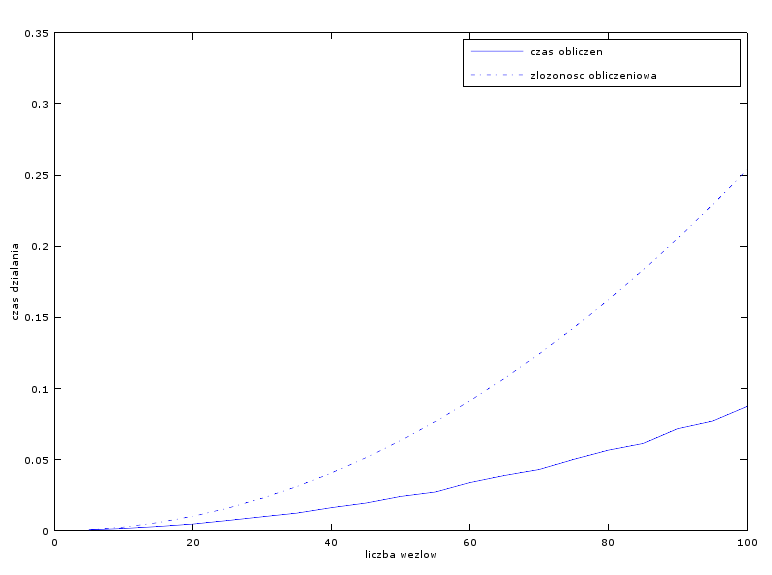
Wynik działania algorytmu jest przechowywany w zmiennej o nazwie „retval”. Ponieważ sieć, dla której zostały przeprowadzone operacje, jest spójna, macierz wynikowa posiada w sobie 9 wierszy zgodnie ze wzorem: gdzie n to ilość krawędzi. Pierwsza i druga kolumna macierzy wynikowej zawierają w sobie wierzchołki połączone krawędzią, natomiast trzecia kolumna przechowuje wagi poszczególnych krawędzi.



Reprezentacja graficzna macierzy wynikowej znajduje się poniżej:



**Wykres:**

****

**Wnioski:**

- W celu osiągnięcia najlepszych czasów wykonywania algorytmu nie należy dzielić skryptu na funkcje, lecz umieścić wszystkie instrukcje kodu w jednym miejscu. MATLab jest językiem interpretowanym, dlatego każde wywołanie podprogramu wiąże się z kosztem czasowym.

- Algorytm Kruskala jest algorytmem zachłannym, ponieważ decyzja o dołączeniu krawędzi do minimalnego drzewa rozpinającego jest podejmowana bez weryfikacji pozostałych krawędzi. Algorytm ten jest jednak pozbawiony ryzyka, że znalezione rozwiązanie nie jest w pełni optymalne, lecz tylko zbliżone do optymalnego. To ryzyko jest dość częste w przypadku algorytmów zachłannych, ponieważ bazują one na poszukiwaniu optimów lokalnych. Ze względu na posortowanie krawędzi względem wag rosnąco, każda bieżąca krawędź niewprowadzająca cykliczności do drzewa jest najbardziej optymalna, dlatego też optimum lokalne znalezione przez algorytm jest optimum globalnym. Nie ma więc ryzyka, że znaleziony wynik jest tylko zbliżony do optymalnego.

- W sieci może być więcej niż jedno minimalne drzewo rozpinające. Zdarza się to, gdy najmniejsze wagi krawędzi pomiędzy jednym punktem mają taką samą wartość. Algorytm Kruskala znajduje tylko jedno drzewo. Nie jest to jednak jego wada, ponieważ do funkcjonowania systemów potrzeba tylko jednego minimalnego drzewa rozpinającego

- Algorytm ten może być stosowany do minimalizacji opóźnień w sieciach komputerowych, długości połączeń w układach elektronicznych, długości tras w ruchu lądowym i powietrznym pomiędzy miastami, w systemach nawigacyjnych i wielu innych

- Algorytm ten do poprawnego wykonywania operacji potrzebuje wolnej pamięci dla trzech macierzy: listy Kruskala, wektora korzeni „roots” i macierzy wynikowej. Wektor roots posiada w sobie n elementów, a macierz wynikowa n – 1 wierszy, każdy po trzy kolumny. Natomiast lista Kruskala może posiadać od do wierszy, każdy po trzy kolumny. Z tego powodu pamięciowa złożoność obliczeniowa algorytmu wynosi , ponieważ graniczne zajmowanie pamięci przez dominującą strukturę, jaką jest lista Kruskala, jest wyrażone wielomianem stopnia drugiego.

Bibliografia:

[1] Algorytmy optymalizacji dyskretnej – Maciej M. Sysło, Narsingh Deo, Janusz S. Kowalik

[2] Algorytmy, struktury danych i techniki programowania – Piotr Wróbelski

[3] <http://iair.mchtr.pw.edu.pl/~bputz/aisd_cpp/lekcja7/segment5/main.htm>

[4] http://www.comp.dit.ie/rlawlor/Alg\_DS/MST/Kruskal.pdf