Łukasz Kowalczyk
Dawid Wojciechowski

SPRAWOZDANIE PROJEKTOWE
GRAFY I SIECI, 17Z

Temat projektu:
Wyznaczanie najkrótszej sieci połączeń wierzchołków grafu.

Opis:
W grafie spójnym G(V,E) każda krawędź ma określoną długość. Dla zadanego podzbioru wierzchołków U należy wyznaczyć spójny podgraf grafu G, który zawiera wierzchołki ze zbioru U i suma długości wszystkich jego krawędzi jest **jak najmniejsza**. Projekt polega na opracowaniu algorytmu i programu wyznaczającego taki podgraf.

1. Założenia projektowe

Zadany problem sprowadza się do znalezienie **minimalnego drzewa Steinera** w grafie.
Definicja problemu:

- dany jest graf spójny niezorientowany G = (V, E, w);
  - *w* - funkcja kosztu krawędzi
  - *V* - zbiór wierzchołków
  - *E* - zbiór krawędzi
- dany jest zbiór terminali (wierzchołków terminalnych) *U* - podzbiór zbioru *V*
- *V\U* - zbiór wierzchołków Steinera (wierzchołków nieterminalnych)
- należy znaleźć minimalne drzewo Steinera - *D*, tzn. drzewo rozpinające wszystkie wierzchołki terminalne o minimalnym koszcie (do drzewa może należeć podzbiór wierzchołków Steinera)

Założenia i oznaczenia:

- v = |V| , e = |E| , t = |U|
- t = 2, …, v
- wagi krawędzi są dodatnie, tzn. w: E -> R_+
- wejściem programu jest graf G oraz zbiór U
- wyjściem programu jest graf D

Dla t = 2 powyższy problem sprowadza się do odnalezienia najkrótszej ścieżki w grafie, natomiast dla t = v – do znalezienia minimalnego drzewa rozpinającego.

2. Format danych wejściowych

Przykładowy plik wejściowy dla pełnego grafu z trzema wierzchołkami(oznaczonymi 1, 2, 3), gdzie dwa wierzchołki są terminalami:

```
GRAPH
E 1 2 1
E 2 3 1
E 3 1 1

TERMINALS
T 1
T 2
```

Gdzie GRAPH oznacza początek sekcji opisującej krawędzie. W tej sekcji nową krawędź oznaczamy literą E, a następujące dwie liczby opisują końcowe wierzchołki dla tej krawędzi, trzecia liczba oznacza natomiast wagę tej krawędzi. Krawędzie te są nie skierowane więc nie ma znaczenia kolejność wierzchołków.
Od frazy TERMINALS zaczyna się opis wierzchołków które są terminalami. W tej sekcji nowy terminal oznaczamy jako T i następnie podajemy identyfikator wierzchołka który będzie terminalem.

3. Format danych wyjściowych

Plik wyjściowy ma taką samą strukturę jak plik wejściowy z dodatkowymi sekcjami.

Sekcja RESULT:

```
RESULT
length 1
time 20
```

Gdzie length jest to suma wag wszystkich krawędzi tworzących drzewo, a time jest to czas w milisekundach jaki zajął algorytmowi.

Sekcja SOLUTION:

```
SOLUTION
S 1 2
```

Literami S w tej sekcji opisujemy krawędzie należące do rozwiązania (tworzące drzewo Steinera).

4. Algorytm

WARIANT I – algorytm dokładny

1) Zbadanie zbioru U:
    a) **jeśli U = V:** szukamy minimalnego drzewo rozpinające (MST) grafu G: algorytm Prima lub Kruskala (wybór algorytmu zależny od typu grafów wejściowych)
    b) **jeśli t = 2:** szukamy najkrótszej ścieżki w grafie G: algorytm Dijkstry
    c) **w p.p.** przejdź do p. 2
2) Przetwarzanie wstępne grafu wejściowego:
    a) usunięcie wszystkich wierzchołków nieterminalnych stopnia 1
    b) scalenie wszystkich wierzchołków terminalnych stopnia 1 z jego sąsiadami, zapamiętując ich krawędzie (będą należały do rozwiązania)
1) **Szukanie minimalnego drzewa Steinera (algorytm Hakimi [1][2]):**

```
min = nieskończoność
D_min = null
dla każdego zbioru A będącego podzbiorem zbioru V\U:
        D = MST dla grafu indukowanego na zbiorze wierzchołków A+U
        jeśli D istnieje:
                jeśli koszt(D) < min:
                        D_min = D
                        min = koszt(D)
zwróć D_min
```

2) zwróć D = D_min z odtworzonymi krawędziami z punktu 2b

Złożoność obliczeniowa: rzędu $O(2^{(v-t)} e \log v)$

WARIANT II – algorytm przybliżony

1) Jak w wariancie I
2) **Szukanie drzewa Steinera (algorytm 1.7-aproksymacyjny KMB: Kou-Markovsky-Berman [2][3]):**
    a) usuń wierzchołki nieterminalne i utwórz graf pełny G'=(U,E'), gdzie E' jest zbiorem krawędzi reprezentujących najkrótsze ścieżki (które trzeba zapamiętać do późniejszego odtworzenia) pomiędzy terminalami - algorytm Dijkstry dla każdej pary terminali
    b) wyznacz D' jako MST w grafie G': algorytm Prima lub Kruskala
    c) wyznacz D poprzez rozwinięcie najkrótszych ścieżek między wierzchołkami D'
    d) zwróć D

Istnieją algorytmy o lepszym przybliżeniu, np. algorytm 1,39-aproksymacyjny opisany w [3].

Złożoność obliczeniowa: rzędu $O(t^2 e \log v)$

5. Testowanie

W ogólności testowanie rozwiązania polegać będzie na mierzeniu czasu wykonania oraz weryfikacji poprawności generowanych rezultatów względem spodziewanych. Głównym wyznacznikiem poprawności jest tutaj sumaryczny koszt wyznaczonego drzewa. Wizualizacja takiego drzewa również może być pomocna w weryfikacji.

Testowanie **wariantu I** polegałoby na uruchomieniu algorytmu na zbiorze kilku/kilkunastu przygotowanych grafów dla różnych zbiorów wierzchołków terminalnych (z uwzględnieniem przypadków, w których |U| = 2 i |U| = |V|).

Testowanie **wariantu II** jest podobne do testowania wariantu I, przy czym ze względu na szybsze działanie i potencjalną suboptymalność generowanych rozwiązań, należałoby sprawdzić jego algorytm na większych i odpowiednio dobranych grafach. Do tego celu można wykorzystać wybrane grafy ze zbiorów ***SteinLib***: http://steinlib.zib.de/steinlib.php.

6. Technologie i narzędzia realizacji

Do realizacji projektu wykorzystana zostanie technologia ***Java 8***. Do kompilacji projektu posłuży nam biblioteka *maven*.

Odpowiednie algorytmy i struktury danych do wizualizacji oraz reprezentacji grafów w programie posłuży biblioteka ***GraphStream***.

7. Literatura

[1] S. L. Hakimi, *Steiner problems in graphs and its implications*, 1971
[2] J. Koszelew, *Problem minimalnego drzewa Steinera. Definicja problemu. Zastosowania. Algorytm dokładny - Hakimi. Algorytmy aproksymacyjne*: http://www.asdpb.republika.pl/gis10.pdf
[3] L. Kou, G. Markowsky, and L. Berman. *A Fast Algorithm for Steiner Trees*, 1981
[4] J. Byrka, F. Grandoni, T. Rothvoss, L. Sanita, *Steiner Tree Approximation via Iterative Randomized Rounding*, 2013