ШИНЖЛЭХ УХААН ТЕХНОЛОГИЙН ИХ СУРГУУЛЬ
Мэдээлэл холбооны технологийн сургууль

БИЕ ДААЛТЫН АЖЛЫН ТАЙЛАН
Хичээл **:** Өгөгдлийн бүтэц ба алгоритм (F.CSM203)
Бие даалтын нэр: “Графын сонгодог алгоритмууд”
Багш: Д. Батмөнх
Гүйцэтгэсэн оюутан 1 : Н.Мөнхбаяр
Гүйцэтгэсэн оюутан 2 : О.Хүслэн
Оюутны код 1: B242270058
Оюутны код 2: B242270073

Улаанбаатар хот 2025 он

Агуулга

1 Оршил

Энэхүү бие даалтын ажлын хүрээнд графын сонгодог алгоритмуудаас минимум бүрхэгч мод (Minimum Spanning Tree – MST) олох Kruskal алгоритмыг сонгон судалж, Java хэл дээр хэрэгжүүлэн программ боловсруулсан.

Kruskal алгоритм нь графын бүх ирмэгүүдийг жингийн дагуу эрэмбэлж, цикл үүсгэхгүй нөхцөлд хамгийн бага жинтэй ирмэгийг сонгож MST үүсгэдэг. Мөн Union-Find буюу Disjoint Set Structure ашигласнаар өндөр үр ашигтайгаар зангилаануудын холбоог шалгадаг.

Даалгаврын шаардлагын дагуу алгоритмын үр дүнг илүү ойлгомжтой байдлаар харагдуулах зорилгоор ASCII график дүрслэл нэмж боловсруулсан.

https://github.com/Huslenc/butets-biy-daalt-3

2 Системийн бүтэц

2.1 Kruskal алгоритмын алхмууд

Графын бүх ирмэгийг жингийн дагуу эрэмбэлнэ.

Union-Find ашиглан хоёр зангилаа нэг модонд багтаж байгаа эсэхийг шалгана.

Хэрвээ цикл үүсгэхгүй бол тухайн ирмэгийг MST-д нэмнэ.

MST-д V–1 ирмэг ормогц зогсоно.

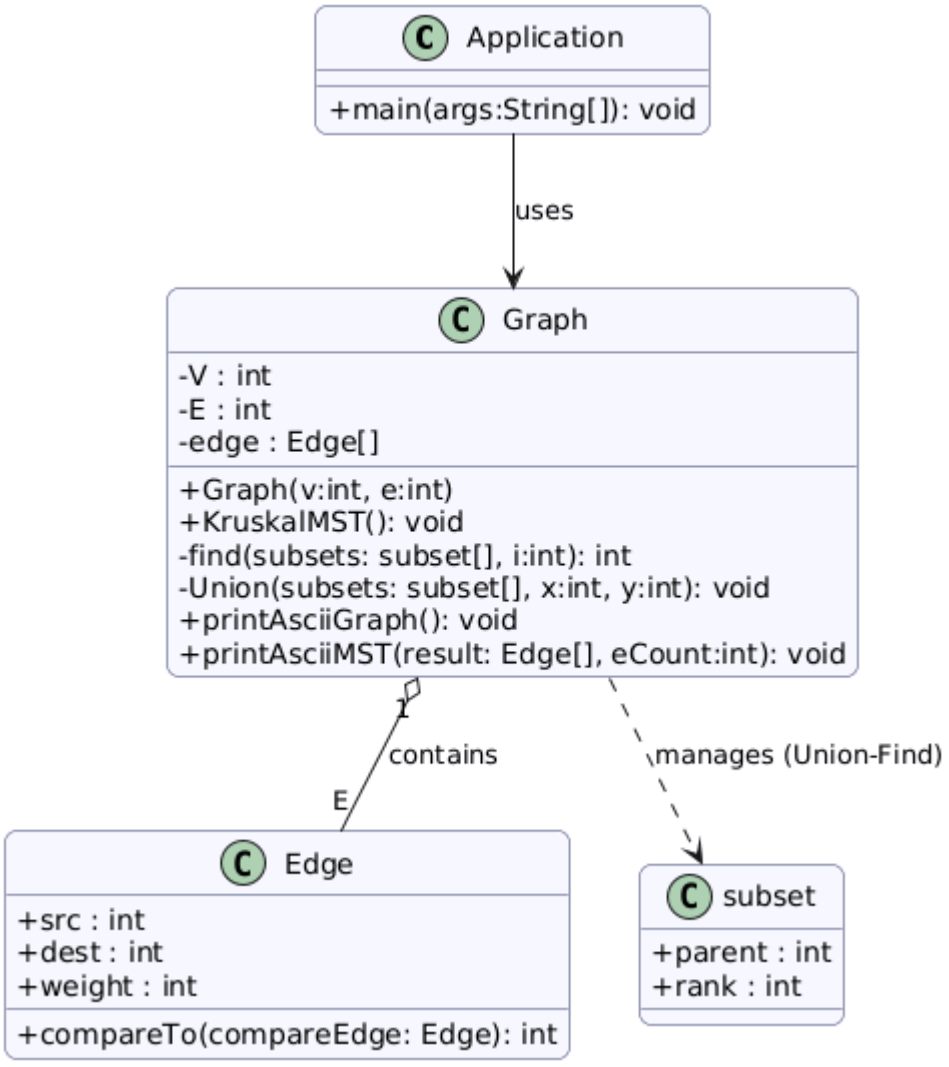
2.2 Ашигласан өгөгдлийн бүтэц

Бүтэц	Ашигласан зорилго
Edge класс	Ирмэг хадгалах
subset (Union-Find)	Хоёр орой холбогдсон эсэхийг шалгах
Edge[] массив	Ирмэгүүдийг цуглуулах
Arrays.sort()	Ирмэгүүдийг эрэмбэлэх

Union-Find нь алгоритмын гүйцэтгэлийг **O(E log E)** болгон оновчтой болгодог.

3 Архитектур болон дизайн

3.1 UML диаграмм



3.2 Классуудын бүтэц

- **Graph класс**
  - Ирмэгүүдийг хадгална
  - Union-Find хэрэгжүүлнэ
  - KruskalMST() функцтэй
- **Edge класс**
  - src, dest, weight гэх гурван талбар
  - Comparable интерфейс ашиглан жингээр эрэмбэлэгдэнэ

subset класс

- parent, rank талбаруудтай
- Union by Rank + Path Compression хэрэгжүүлдэг

3.2 ASCII график дүрслэл

Даалгаврын “гаралт ойлгомжтой байх” нөхцөлийг хангах үүднээс дараах хоёр нэмэлт модуль боловсруулав:

- **printAsciiGraph()** — бүх ирмэгийг график хэлбэрээр харуулна
- **printAsciiMST()** — зөвхөн MST-д сонгогдсон ирмэгүүдийг "[SELECTED]" тэмдэглэгээтэй харуулна

4. Туршилт ба гаралт

4.1 ASCII График (Анхны граф)
(0) ----10---- (1)

(0) ----6----- (2)

(0) ----5----- (3)

(1) ----15---- (3)

(2) ----4----- (3)

4.2 Kruskal алгоритмын үр дүн
MST Edges:

2 -- 3 == 4

0 -- 3 == 5

0 -- 1 == 10

Minimum Cost Spanning Tree = 19

4.3 ASCII MST (Сонгогдсон ирмэгүүд)
(2) ----4---- (3) [SELECTED]

(0) ----5---- (3) [SELECTED]

(0) ----10--- (1) [SELECTED]

5 Хэрэглэх заавар

5.1 Системийн шаардлага

Java JDK 8 эсвэл хожим
• 1GB RAM ба түүнээс дээш
• 100MB чөлөөт дискний зай

5.2 Суулгах заавар

1. Кодыг компайл хийх

2. Программ ажиллуулах

6 Unit Test

6.1 Unit Test код

```
import org.junit.Test;

import static org.junit.Assert.*;

public class GraphTest {

    @Test

    public void testKruskalMST() {

        Graph graph = new Graph(4, 5);

        graph.edge[0].src = 0;

        graph.edge[0].dest = 1;

        graph.edge[0].weight = 10;
```

```
        graph.edge[1].src = 0;
        graph.edge[1].dest = 2;
        graph.edge[1].weight = 6;
        graph.edge[2].src = 0;
        graph.edge[2].dest = 3;
        graph.edge[2].weight = 5;
        graph.edge[3].src = 1;
        graph.edge[3].dest = 3;
        graph.edge[3].weight = 15;
        graph.edge[4].src = 2;
        graph.edge[4].dest = 3;
        graph.edge[4].weight = 4;
        Edge[] result = graph.runKruskalForTest();
          assertEquals(3, result.length);
        int total = result[0].weight + result[1].weight + result[2].weight;
        assertEquals(19, total);
    }
}
```

6.2 Туршилтын үр дүн

MST зөв тооцогдсон

Цикл үүсээгүй

Ирмэгүүд жингийн дагуу зөв сонгогдсон

Систем алдаагүй ажилласан

7 Дүгнэлт

7.1 Хүрсэн үр дүн

Kruskal алгоритмыг бүрэн хэрэгжүүлсэн

Union-Find өгөгдлийн бүтцийг ашиглан гүйцэтгэлийг сайжруулсан

ASCII график дүрслэл нэмж, гаралтыг ойлгомжтой болгосон

Unit Test ашиглан зөв ажиллаж буйг нотолсон

7.2 Алгоритмын давуу тал

Үйлдэл	**Ажиллах хугацаа**	**Тайлбар**
Ирмэг эрэмбэлэх	O(E log E)	Kruskal-ийн гол зардал
Union-Find	O(α(n)) ~ O(1)	Маш өндөр үр ашигтай
Нийт ажиллагаа	O(E log E)	Онолын хувьд оптимал

7.3 Ирээдүйн хөгжүүлэлт

Java Swing ашиглан график UI нэмэх

Хэрэглэгч өөрийн граф оруулдаг хувилбар болгох

Prim алгоритмтай харьцуулсан туршилт хийх

MST-г зураг болгон export хийх