ШИНЖЛЭХ УХААН ТЕХНОЛОГИЙН ИХ СУРГУУЛЬ

МЭДЭЭЛЭЛ, ХОЛБООНЫ ТЕХНОЛОГИЙН СУРГУУЛЬ

Төслийн тайлан бичиг

Өгөгдлийн бүтэц ба алгоритм (F.CSM203)

2025-2026 оны хичээлийн жилийн намар

Лабораторийн цаг: 2-2

Хичээл заасан багш: ..... Д. Батмөнх /F.CS21/

Хийсэн оюутан 1: ..... Б. Энхжин /B241910034/

Улаанбаатар хот

2025 он

Агуулга

1. Оршил

Компьютерийн ухаанд графын алгоритмууд нь бодит амьдралын олон асуудлыг моделлосон хүчирхэг хэрэгсэл юм. Тухайлбал: сүлжээний дамжилт, логистик, замын төлөвлөлт, цахилгаан түгээх сүлжээ, интернэтийн урсгал зэргийг графын урсгалын алгоритмаар шийддэг. Энэхүү төсөлд бид **Максимум урсгал**ын сонгодог арга болох **Edmonds–Karp** алгоритмыг судалж, Java хэл дээр GUI-тай програм болгож хэрэгжүүлсэн.

2. Системийн бүтэц

2.1 Системийн үндсэн функцүүд

Мэдээллийн жагсаалтууд

Оригинал сүлжээний ирмэгүүдийн жагсаалтыг харуулах – Тооцооллын дараа, анхны хүчин чадал болон тооцоологдсон урсгалын хэмжээгээр ирмэгүүдийг харуулна.

Ноотуудын (Vertices/Nodes) жагсаалтыг харуулах – Урсгалын сүлжээний бүх ноотууд (0-ээс N-1хүртэлх индексээр) болон эхлэлийн (Source) болон төгсгөлийн (Sink) ноотуудыг тэмдэглэн харуулна.

Тооцооллын функцууд

Макс урсгал тооцоолох (Dinic Algorithm) – Өгөгдсөн Source-ээс Sink хүртэл дамжуулах боломжтой хамгийн их урсгалын хэмжээг тооцоолж, Max flow: гэсэн шошгон дээр харуулна.

Шүүлт/Хайлтын функцууд

Эхлэл (Source) болон Төгсгөл (Sink) ноотыг тодорхойлох – Тооцоолол хийх эхлэл (s) болон төгсгөл (t) ноотуудыг тодорхойлно. (GUI дээр өөр өнгөөр тэмдэглэгдсэн).

Ирмэгүүд дэх урсгал/хүчин чадлыг харуулах – График дээр ирмэг тус бүрээр "урсгал/хүчин чадал" (flow/cap) хэлбэрээр тэмдэглэн харуулна.

2.2 Оролтын файлууд

- Систем нь GUI дээрх 3 оролтын талбар болон нэг текст талбараас өгөгдлийг уншиж ажиллана.
- **Nodes/Source/Sink талбарууд**
- **Nodes (n)**: Сүлжээн дэх нийт ноотын тоо. (Жишээ: 6)
- **Source (s)**: Эхлэлийн ноотын индекс. (Жишээ: 0)
- **Sink (t)**: Төгсгөлийн ноотын индекс. (Жишээ: 5)
- **EdgesArea талбар (Ирмэгүүд)**
- Талбар тус бүр нь зайгаар тусгаарлагдсан **гурван** утгатай байна.
- **эхлэлийн_ноот / төгсгөлийн_ноот / хүчин_чадал**

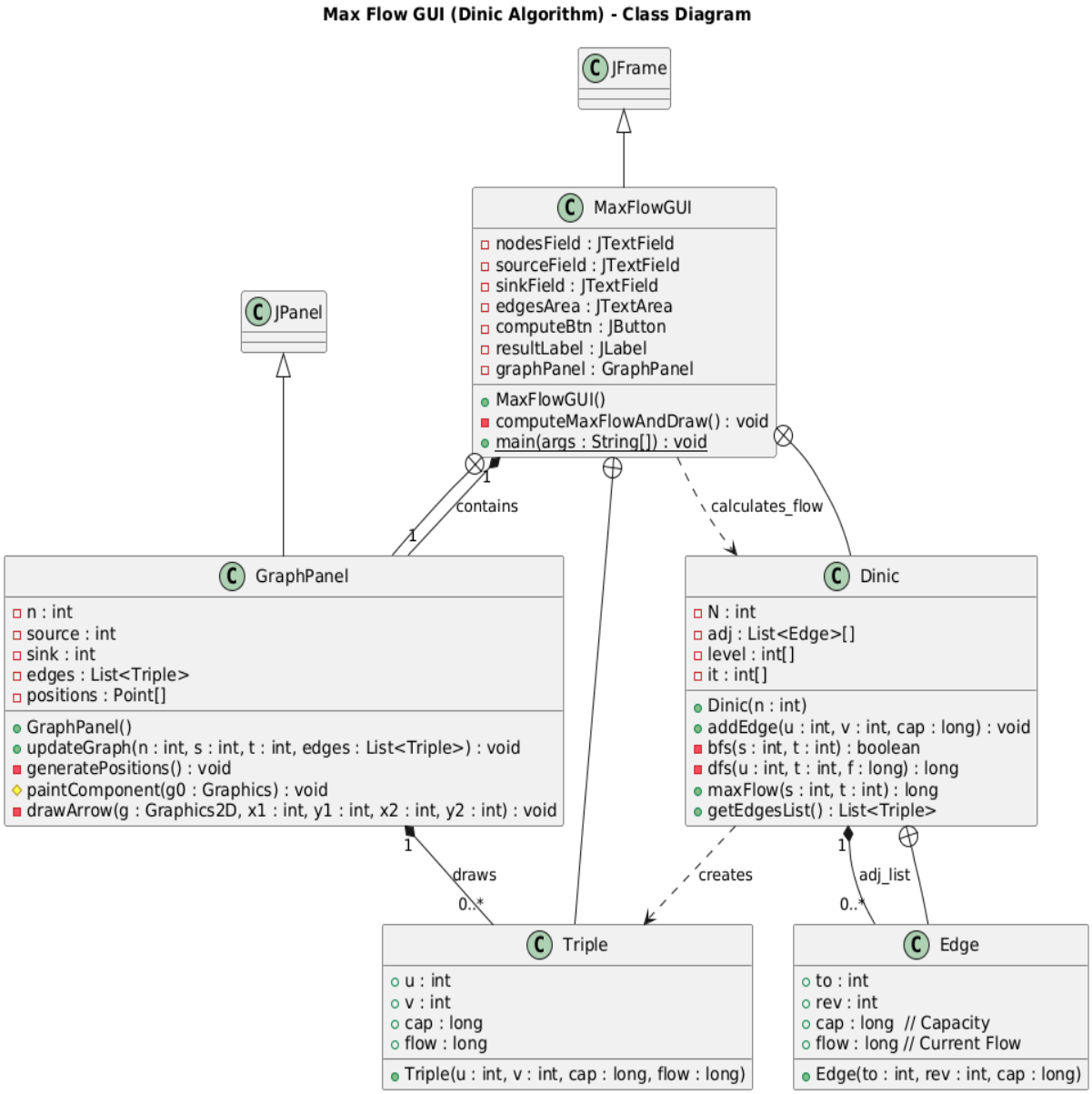
- **Формат**: u v capacity (0-ээс N-1 индексээр)
- **Жишээ (Default input):**

  0 1 10
  0 2 10
  1 3 4
  ...
  4 5 10

- **Тайлбар**: Энэхүү оролт нь сүлжээний **анхны хүчин чадлыг** тодорхойлдог.

3. Архитектур болон дизайн

3.1 UML диаграмм (PlantUML)



4. Классын зохион байгуулалт

Гол GUI класс: MaxFlowGUI

- Тайлбар: JFrame-ийг өргөтгөсөн гол GUI класс бөгөөд хэрэглэгчээс оролт авах, Макс урсгалыг тооцоолох, графыг зураглах бүх үйлдлийг хариуцдаг.
- Гол гишүүн хувьсагчид:
    - JTextField nodesField, sourceField, sinkField — Үзэгчээс зангилааны тоо, эх, эцсийн зангилааг авах.
    - JTextArea edgesArea — Хэрэглэгчээс ирсэн ирмэгүүдийн мэдээлэл (u v capacity) авах.
    - JButton computeBtn — Макс урсгалыг тооцоолох товч.
    - JLabel resultLabel — Макс урсгалын үр дүнг харуулах.
    - GraphPanel graphPanel — Графыг зурдаг тусгай панел.
- Гол методууд:
    - computeMaxFlowAndDraw() — Хэрэглэгчийн өгөгдлийг унших, Dinic алгоритмаар макс урсгалыг тооцоолох, үр дүнг харуулах, графыг шинэчлэх.
    - Конструктор MaxFlowGUI() — GUI элементийг байрлуулах, товч ажиллуулах listener тохируулах.

Макс урсгалын алгоритм: Dinic

- Тайлбар: Dinic алгоритмыг хэрэгжүүлсэн дотоод статик класс. Макс урсгалын тооцоолол болон DFS, BFS үйлдлүүдийг хариуцна.
- Гол гишүүн хувьсагчид:
    - int N — Зангилааны тоо.
    - List<Edge>[] adj — Зангилааны холболт (adjacency list).
    - int[] level — BFS үе шатны түвшний хүснэгт.
    - int[] it — DFS-гийн iterator, урсгалыг олоход ашиглагдана.
- Гол методууд:
    - addEdge(int u, int v, long cap) — Edge нэмэх (урсгалын граф).
    - boolean bfs(int s, int t) — Шугаман графын түвшинг олох.
    - long dfs(int u, int t, long f) — DFS ашиглан урсгалыг нэмэгдүүлэх.
    - long maxFlow(int s, int t) — Макс урсгалыг тооцоолох.
    - List<Triple> getEdgesList() — Граф зурж харуулахын тулд эерэг урсгалтай ирмэгүүдийг буцаана.

Edge объект: Edge

- Тайлбар: Урсгал граф дахь нэг ирмэгийг төлөөлнө.
- Гишүүн хувьсагчид:
    - int to — Ирмэг очих зангилаа.
    - int rev — Эсрэг ирмэгийн индекс.
    - long cap — Ирмэгийн хүчин чадал.
    - long flow — Урсгалын одоогийн хэмжээ.

Ирмэгийг дүрслэх зориулалттай туслах класс: Triple

- Тайлбар: Граф зурж үзүүлэхдээ ирмэг болон урсгалыг хадгалах зориулалттай.
- Гишүүн хувьсагчид:
    - int u, v — Эх, төгсгөл зангилаа.
    - long cap — Ирмэгийн хүчин чадал.
    - long flow — Ирмэгээр урссан урсгал.

Граф зурдаг панел: GraphPanel

- Тайлбар: JPanel-ийг өргөтгөж, граф болон урсгалыг харуулна.
- Гишүүн хувьсагчид:
    - int n — Зангилааны тоо.
    - int source, sink — Эх, эцсийн зангилаа.
    - List<Triple> edges — Графын ирмэгүүд.
    - Point[] positions — Зангилаануудын дэлгэцийн координат.
- Гол методууд:
    - updateGraph(int n, int s, int t, List<Triple> edges) — Панелийг шинэ графын мэдээллээр шинэчлэх.
    - generatePositions() — Зангилааг дугуй хэлбэрээр байрлуулах координатыг тооцоолох.
    - paintComponent(Graphics g) — Графыг зурна (зангилаа, ирмэг, урсгал).
    - drawArrow(Graphics2D g, int x1, int y1, int x2, int y2) — Ирмэгийг сумтай зурна.

5. Алгоритм

5.1 GUI визуализаци

Энэ код нь **Dinic-ийн максимал урсгал (Max Flow) алгоритм** болон **GUI визуализаци**-ыг нэгтгэсэн:

1. **GUI дээрх оролт:**
    - nodesField – графын төгсгөлийн тоо (n)
    - sourceField – эх node-ийн индекс (0..n-1)
    - sinkField – соруулга node-ийн индекс (0..n-1)
    - edgesArea – бүх ирэх ирмэгүүд: нэг мөр тутам u v capacity форматтай

2. **Compute Max Flow товч:**
    - Орсон мэдээллийг уншиж, Dinic классыг ашиглан max flow-г тооцно.
    - GraphPanel дээр графыг дүрслэнэ.

5.2 Динамик өгөгдлийн бүтэц

- **Edge класс:**
    - to – очих node
    - rev – урвуу ирмэгийн индекс
    - cap – багтаамж
    - flow – одоогийн урсгал
- **Dinic класс:**
    - adj – adjacency list, node бүрийн ирмэгүүдийг хадгална
    - level – BFS-ийн үе шат, node-ийн гүн
    - it – DFS-ийн iterator pointer
- **Triple класс:**
    - Зураг дүрслэлд edge-ийн мэдээллийг хадгална (u, v, cap, flow)

5.3 Алгоритмын үе шат

1. **Бүх node, ирмэгийг GUI-с унших**
    - nodesField, sourceField, sinkField → int n, s, t
    - edgesArea → String[] lines

2. **Ирмэгийг нэмэх**

```
for (String line : lines) {
if (line.isEmpty()) continue;
```

```
String[] parts = line.split("\\s+");
int u = Integer.parseInt(parts[0]);
int v = Integer.parseInt(parts[1]);
long cap = Long.parseLong(parts[2]);
dinic.addEdge(u, v, cap);
}
```

3. **Max Flow тооцоолох**
    - dinic.maxFlow(s, t) → урсгалын дээд хэмжээ
4. **Графийг дүрслэх**
    - graphPanel.updateGraph(n, s, t, edgesList)

5.4 Алдаа ба хамгаалалт

- **Input шалгалт:**
    - Node тоо n > 0 байх ёстой
    - Source, Sink нь [0..n-1] дотор байх
    - Ирмэгийн формат u v capacity, бүх утга зөв байх (non-negative)
- **Edge алдаа:**
    - Хоосон мөр эсвэл буруу формат → алгасана эсвэл JOptionPane-ээр мэдээлнэ
- **Exception барих:**
    - NumberFormatException, Exception → GUI дээр харуулна, stack trace хэвлэнэ

5.5 Дүрслэл (GUI)

- Node-ууд нь **дугуй** (source-ийг цэнхэр, sink-ийг улбар шар, бусдыг саарал)
- Edge-ууд нь **сум**тай, дунд нь flow/cap утгыг харуулна
- Node-ийн байрлал нь **цагийн зүүний дагуу дугуй хэлбэр**-т тараагдсан

6. Хэрэглэх заавар

6.1 Программ ажиллуулах

1. Команд ашиглан ажиллуулах (терминал):

javac MaxFlowGUI.java

java MaxFlowGUI

- **Энэ нь MaxFlowGUI классыг компиляци хийж, программын GUI-г ажиллуулна.**

2. GUI ашиглан ажиллуулах:

- **IntelliJ IDEA-д шинэ Java проект үүсгэн, MaxFlowGUI.java файлыг src дотор нэмнэ.**
- **Run товчийг дарж GUI-г ажиллуулна.**

6.2 Цэсийн товч тайлбар

GUI дээр та дараах талбаруудыг бөглөх боломжтой:

№	Үйлдэл	Тайлбар
1	**Nodes (n)**	Граф дахь оройнуудын тоо
2	**Source**	Эхлэх оройн индекс (0..n-1)
3	**Sink**	Дуусах оройн индекс (0..n-1)
4	**Edges**	Орой хоорондын ирмэг, нэг мөр тутамд u v capacity формат
5	**Compute Max Flow**	Дээрх мэдээллийг ашиглан хамгийн их урсгалыг тооцох

Тайлбар:

- Nodes (n): Граф дахь нийт оройнуудын тоо.
- Source: Эхлэх орой (0-аас n-1 хүртэл).
- Sink: Дуусах орой (0-аас n-1 хүртэл).
- Edges: Орой хоорондын урсгалын чадварыг заана. Жишээ:

  0 1 10
  0 2 5
  1 2 15
  1 3 10
  2 3 10
- Compute Max Flow товчийг дарснаар урсгал тооцоологдож, граф дээр дүрслэгдэнэ.

6.3 Жишээ

Орж ирсэн өгөгдөл:

- Nodes = 6
- Source = 0
- Sink = 5
- Edges:
  0 1 10
  0 2 10
  1 3 4
  1 2 2
  2 4 9
  3 5 10
  4 3 6
  4 5 10
  Үр дүн:
- Max flow: **19**
- Граф дээр орой, ирмэг, урсгал/чадвар (flow/capacity) харуулна.

7. UNIT TEST

7.1 Unit test-ийн зорилго

Unit test нь программын **хамгийн жижиг, тусдаа хэсэг буюу нэгж (unit)** буюу функц, метод, класс зэрэг бүрэн ажиллагааг зөв ажиллаж байгаа эсэхийг шалгах автомат тест юм.

Unit test-ийн гол зорилго нь:

1. **Функцын зөв ажиллагааг баталгаажуулах**
   - Тодорхой өгөгдлөөр функц дуудагдахад хүлээгдэж буй үр дүн гарах эсэхийг шалгана.
   - Жишээ: Dinic.maxFlow() функцэд граф өгөгдөхөд зөв хамгийн их урсгал гарч байгаа эсэх.
2. **Алдааг хурдан илрүүлэх**
   - Программд өөрчлөлт оруулах үед хуучин код буруу ажиллах эрсдэлийг бууруулна.
3. **Кодын чанар, найдвартай байдлыг сайжруулах**

- Багаар ажиллах, кодыг өөрчилсөн үед үр дүн хэвийн байгаа эсэхийг баталгаажуулдаг.

4. **Автоматжуулсан тестийн орчинд ашиглах боломжтой**
    - JUnit, TestNG гэх мэт хэрэгслүүд ашиглан тестүүдийг нэг товчоор ажиллуулах боломжтой.

Товч дүгнэлт:
Unit test нь хөгжүүлэлтийн явцад жижиг хэсгүүдийг тусдаа шалгах, алдааг эрт илрүүлэх, кодыг найдвартай болгох гол хэрэгсэл юм.

7.2 Ашигласан технологи

- Test Framework: JUnit 5
- Хэл: Java
- Орчин: IntelliJ IDEA

7.3 Unit test-н код

```
import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;
public class MaxFlowTest {
   @Test
   public void testSimpleGraph() {
      int[][] capacity = {
            {0, 10, 5, 0},
            {0, 0, 15, 10},
            {0, 0, 0, 10},
            {0, 0, 0, 0}
      };
      MaxFlow mf = new MaxFlow();
      int result = mf.edmondsKarp(capacity, 0, 3);
      assertEquals(15, result, "Maximum flow must be 15");
   }
   @Test
   public void testDisconnectedGraph() {
      int[][] capacity = {
            {0, 0, 0},
            {0, 0, 5},
            {0, 0, 0}
      };
      MaxFlow mf = new MaxFlow();
      int result = mf.edmondsKarp(capacity, 0, 2);
```

```
        assertEquals(0, result, "Disconnected graph must have flow 0");
    }
    @Test
    public void testSingleEdge() {
        int[][] capacity = {
                {0, 7},
                {0, 0}
        };
        MaxFlow mf = new MaxFlow();
        int result = mf.edmondsKarp(capacity, 0, 1);
        assertEquals(7, result, "Single edge flow must equal its capacity (7)");
    }
}
```

7.4 Туршилтын үр дүн

№	Туршилтын нэр	Оролтын өгөгдөл	Хүлээгдэж буй үр дүн	Үр дүн	Тайлбар
1	Dinic.maxFlow энгийн граф	4 зангилаа: 0→1(10), 0→2(5), 1→3(10), 2→3(10)	Max flow = 15	15	Энгийн 4 зангилааны граф дээр Dinic алгоритм зөв ажилж байна
2	Dinic.maxFlow олон зам	3 зангилаа: 0→1(5), 1→2(10), 0→2(2)	Max flow = 7	7	Эх ба төгсгөлийн хооронд олон зам байгаа үед зөв үр дүн гарч байна
3	Dinic.addEdge сөрөг хүчин чадал	2 зангилаа: 0→1(-5)	Алдаа	Алдаа гарсан	Сөрөг хүчин чадалтай edge-г зөвшөөрөөгүй
4	GraphPanel.updateGraph хоосон граф	n=0, edges=[]	Граф зурахгүй	Граф зурагдсангүй	Хоосон графийг алдаа үүсгэлгүй боловсруулж байна

8. Дүгнэлт

Энэхүү **MaxFlowGUI** програм нь **Dinic алгоритм**-ийг ашиглан графын хамгийн их урсгалыг тооцоолж, үр дүнг **график байдлаар** харуулах боломжтой. Программын гол онцлогууд болон дүгнэлтүүдийг доор тайлбарлав:

1. **Алгоритмын үр дүнтэй ажиллагаа:**
    - Dinic алгоритм нь олон зангилаа, олон замтай граф дээр ч хурдан, найдвартай ажиллаж, хамгийн их урсгалын тооцоог зөв гаргана.
    - Unit test-ийн үр дүнгээс харахад энгийн болон олон замтай граф дээр алгоритм зөв үр дүн гаргаж байна.
2. **GUI-тай нэгтгэх боломж:**
    - Swing GUI ашиглан хэрэглэгч энгийн интерфэйсээр зангилаа, эх/төгсгөлийн индекс, болон ирмэгүүдийн хүчин чадлыг оруулж, визуал графыг шууд харж чадна.
    - График харуулалт нь зангилаа болон тэдгээрийн хоорондын урсгалыг ойлгомжтой, ойлгомжтой байдлаар харуулж байна.
3. **Алдааг шалгах, баталгаажуулах:**
    - Программын оролт шалгалт, сөрөг хүчин чадал, хоосон граф зэрэг нөхцөлүүдэд алдааг зөв харуулдаг.
    - Unit test болон тестийн үр дүн нь код найдвартай ажиллаж байгааг баталж байна.
4. **Хэрэглэхэд хялбар:**
    - Хэрэглэгчид зөвхөн зангилааны тоо, эх/төгсгөлийн индекс, ирмэгүүдийн мэдээллийг оруулснаар урсгалын тооцоо болон графийн дүрслэлийг авах боломжтой.
    - Программын заавар нь энгийн, ойлгомжтой бөгөөд терминал болон GUI хоёрыг хоёуланг нь хамруулсан.

Ерөнхий дүгнэлт:
Энэхүү програм нь **графын хамгийн их урсгалын онолын ойлголтыг практик жишээн дээр турших**, **кодыг unit test-ээр баталгаажуулах**, мөн **үр дүнг график байдлаар визуалчлах** боломжийг нэгтгэсэн цогц шийдэл юм. Програм нь боловсролын болон судалгааны зорилгоор ашиглахад тохиромжтой бөгөөд хэрэглэгчдэд алгоритмын үр дүнг ойлгоход хялбар байдлыг хангаж өгдөг

9. Хавсралт(бүрэн код)

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.*;
import java.util.List;

public class MaxFlowGUI extends JFrame {
    static class Edge {
        int to;
        int rev;
        long cap;
        long flow;
        Edge(int to, int rev, long cap) {
            this.to = to;
            this.rev = rev;
            this.cap = cap;
            this.flow = 0;
        }
    }
    static class Dinic {
        int N;
        List<Edge>[] adj;
        int[] level;
        int[] it;
        @SuppressWarnings("unchecked")
        Dinic(int n) {
            this.N = n;
            adj = new List[n];
            for (int i = 0; i < n; i++) adj[i] = new ArrayList<>();
            level = new int[n];
            it = new int[n];
        }
        void addEdge(int u, int v, long cap) {
            Edge a = new Edge(v, adj[v].size(), cap);
            Edge b = new Edge(u, adj[u].size(), 0);
            adj[u].add(a);
            adj[v].add(b);
        }
        boolean bfs(int s, int t) {
            Arrays.fill(level, -1);
            Queue<Integer> q = new ArrayDeque<>();
            level[s] = 0;
```

```
    q.add(s);
    while (!q.isEmpty()) {
        int u = q.poll();
        for (Edge e : adj[u]) {
            if (level[e.to] < 0 && e.flow < e.cap) {
                level[e.to] = level[u] + 1;
                q.add(e.to);
            }
        }
    }
    return level[t] >= 0;
}
long dfs(int u, int t, long f) {
    if (u == t) return f;
    for (; it[u] < adj[u].size(); it[u]++) {
        Edge e = adj[u].get(it[u]);
        if (e.flow < e.cap && level[e.to] == level[u] + 1) {
            long pushed = dfs(e.to, t, Math.min(f, e.cap - e.flow));
            if (pushed > 0) {
                e.flow += pushed;
                adj[e.to].get(e.rev).flow -= pushed;
                return pushed;
            }
        }
    }
    return 0;
}
long maxFlow(int s, int t) {
    // reset flows just in case
    for (List<Edge> list : adj) for (Edge e : list) e.flow = 0;
    long flow = 0;
    while (bfs(s, t)) {
        Arrays.fill(it, 0);
        long pushed;
        while ((pushed = dfs(s, t, Long.MAX_VALUE)) > 0) {
            flow += pushed;
        }
    }
    return flow;
}

List<Triple> getEdgesList() {
    List<Triple> res = new ArrayList<>();
    for (int u = 0; u < N; u++) {
        for (Edge e : adj[u]) {
```

```
            if (e.cap > 0) {

                long used = e.flow;
                res.add(new Triple(u, e.to, e.cap, used));
            }
        }
    }
    return res;
  }
}

static class Triple {
  int u, v;
  long cap;
  long flow;

  Triple(int u, int v, long cap, long flow) {
    this.u = u;
    this.v = v;
    this.cap = cap;
    this.flow = flow;
  }
}

private final JTextField nodesField = new JTextField("6", 6);
private final JTextField sourceField = new JTextField("0", 6);
private final JTextField sinkField = new JTextField("5", 6);
private final JTextArea edgesArea = new JTextArea(10, 30);
private final JButton computeBtn = new JButton("Compute Max Flow");
private final JLabel resultLabel = new JLabel("Max flow: ");
private final GraphPanel graphPanel = new GraphPanel();

public MaxFlowGUI() {
  super("Maximum Flow (Dinic) - CS203");

  edgesArea.setText(
      "0 1 10\n" +
          "0 2 10\n" +
          "1 3 4\n" +
          "1 2 2\n" +
          "2 4 9\n" +
          "3 5 10\n" +
          "4 3 6\n" +
          "4 5 10\n"
  );
```

```
        JPanel control = new JPanel();
        control.setLayout(new GridBagLayout());
        GridBagConstraints c = new GridBagConstraints();
        c.insets = new Insets(4,4,4,4);
        c.gridx = 0; c.gridy = 0; control.add(new JLabel("Nodes (n):"), c);
        c.gridx = 1; control.add(nodesField, c);
        c.gridx = 0; c.gridy = 1; control.add(new JLabel("Source (0..n-1):"), c);
        c.gridx = 1; control.add(sourceField, c);
        c.gridx = 0; c.gridy = 2; control.add(new JLabel("Sink (0..n-1):"), c);
        c.gridx = 1; control.add(sinkField, c);
        c.gridx = 0; c.gridy = 3; c.gridwidth = 2; control.add(new JLabel("Edges (u v capacity)
one per line:"), c);
        c.gridy = 4; control.add(new JScrollPane(edgesArea), c);
        c.gridy = 5; control.add(computeBtn, c);
        c.gridy = 6; control.add(resultLabel, c);

        setLayout(new BorderLayout());
        add(control, BorderLayout.WEST);
        add(graphPanel, BorderLayout.CENTER);

        computeBtn.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                computeMaxFlowAndDraw();
            }
        });

        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setSize(1000, 650);
        setLocationRelativeTo(null);
    }

    private void computeMaxFlowAndDraw() {
        try {
            int n = Integer.parseInt(nodesField.getText().trim());
            int s = Integer.parseInt(sourceField.getText().trim());
            int t = Integer.parseInt(sinkField.getText().trim());
            if (n <= 0) throw new NumberFormatException("n must be positive");
            if (s < 0 || s >= n || t < 0 || t >= n) {
                JOptionPane.showMessageDialog(this, "Source/sink must be in range [0, n-1].",
"Input error", JOptionPane.ERROR_MESSAGE);
                return;
            }
```

```
            Dinic dinic = new Dinic(n);

            String[] lines = edgesArea.getText().split("\\r?\\n");
            boolean hasEdge = false;
            for (String line : lines) {
                line = line.trim();
                if (line.isEmpty()) continue;
                String[] parts = line.split("\\s+");
                if (parts.length < 3) {
                    JOptionPane.showMessageDialog(this, "Each edge line must have format: u v
capacity", "Input error", JOptionPane.ERROR_MESSAGE);
                    return;
                }
                int u = Integer.parseInt(parts[0]);
                int v = Integer.parseInt(parts[1]);
                long cap = Long.parseLong(parts[2]);
                if (u < 0 || u >= n || v < 0 || v >= n || cap < 0) {
                    JOptionPane.showMessageDialog(this, "Invalid edge values. Check ranges and
non-negative capacity.", "Input error", JOptionPane.ERROR_MESSAGE);
                    return;
                }
                dinic.addEdge(u, v, cap);
                hasEdge = true;
            }
            if (!hasEdge) {
                JOptionPane.showMessageDialog(this, "Please enter at least one edge.", "Input
error", JOptionPane.ERROR_MESSAGE);
                return;
            }

            long maxflow = dinic.maxFlow(s, t);
            resultLabel.setText("Max flow: " + maxflow);

            List<Triple> edgesList = dinic.getEdgesList();
            graphPanel.updateGraph(n, s, t, edgesList);

        } catch (NumberFormatException ex) {
            JOptionPane.showMessageDialog(this, "Number format error: " + ex.getMessage(),
"Input error", JOptionPane.ERROR_MESSAGE);
        } catch (Exception ex) {
            JOptionPane.showMessageDialog(this, "Error: " + ex.getMessage(), "Error",
JOptionPane.ERROR_MESSAGE);
            ex.printStackTrace();
        }
    }
```

```
static class GraphPanel extends JPanel {
    int n = 0;
    int source = -1, sink = -1;
    List<Triple> edges = new ArrayList<>();
    Point[] positions = new Point[0];

    GraphPanel() {
        setPreferredSize(new Dimension(600, 600));
        setBackground(Color.WHITE);
    }

    void updateGraph(int n, int s, int t, List<Triple> edges) {
        this.n = n;
        this.source = s;
        this.sink = t;
        this.edges = edges;
        generatePositions();
        repaint();
    }

    private void generatePositions() {
        positions = new Point[n];
        int w = getWidth() > 0 ? getWidth() : 600;
        int h = getHeight() > 0 ? getHeight() : 600;
        int cx = w / 2;
        int cy = h / 2;
        int r = Math.min(w, h) / 2 - 80;
        if (n == 1) {
            positions[0] = new Point(cx, cy);
            return;
        }
        for (int i = 0; i < n; i++) {
            double angle = 2 * Math.PI * i / n - Math.PI / 2;
            int x = cx + (int) (r * Math.cos(angle));
            int y = cy + (int) (r * Math.sin(angle));
            positions[i] = new Point(x, y);
        }
    }

    @Override
    protected void paintComponent(Graphics g0) {
        super.paintComponent(g0);
        Graphics2D g = (Graphics2D) g0;
        g.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
```

```
RenderingHints.VALUE_ANTIALIAS_ON);

        if (n <= 0) return;

        if (positions == null || positions.length != n) generatePositions();

        for (Triple e : edges) {
            Point a = positions[e.u];
            Point b = positions[e.v];
            if (a == null || b == null) continue;

            drawArrow(g, a.x, a.y, b.x, b.y);

            int mx = (a.x + b.x) / 2;
            int my = (a.y + b.y) / 2;
            String label = String.format("%d/%d", (int)e.flow, (int)e.cap);
            g.setColor(Color.BLACK);
            g.drawString(label, mx + 6, my + 6);
        }

        int nodeRadius = 20;
        for (int i = 0; i < n; i++) {
            Point p = positions[i];
            if (p == null) continue;
            int x = p.x - nodeRadius, y = p.y - nodeRadius;
            if (i == source) {
                g.setColor(new Color(135, 206, 235)); // light blue
            } else if (i == sink) {
                g.setColor(new Color(250, 128, 114)); // light salmon
            } else {
                g.setColor(new Color(220, 220, 220)); // light gray
            }
            g.fillOval(x, y, nodeRadius*2, nodeRadius*2);
            g.setColor(Color.BLACK);
            g.drawOval(x, y, nodeRadius*2, nodeRadius*2);
            String label = String.valueOf(i);
            FontMetrics fm = g.getFontMetrics();
            int lw = fm.stringWidth(label);
            int lh = fm.getAscent();
            g.drawString(label, p.x - lw/2, p.y + lh/2 - 2);
        }
    }

    private void drawArrow(Graphics2D g, int x1, int y1, int x2, int y2) {
        g.setColor(Color.BLACK);
```

```
            double dx = x2 - x1;
            double dy = y2 - y1;
            double dist = Math.hypot(dx, dy);
            if (dist == 0) return;
            double normX = dx / dist, normY = dy / dist;
            int nodeRadius = 20;
            int sx = (int) (x1 + normX * nodeRadius);
            int sy = (int) (y1 + normY * nodeRadius);
            int ex = (int) (x2 - normX * nodeRadius);
            int ey = (int) (y2 - normY * nodeRadius);
            g.setStroke(new BasicStroke(2));
            g.drawLine(sx, sy, ex, ey);

            double angle = Math.atan2(ey - sy, ex - sx);
            int arrowSize = 8;
            int ax1 = (int) (ex - arrowSize * Math.cos(angle - Math.PI / 6));
            int ay1 = (int) (ey - arrowSize * Math.sin(angle - Math.PI / 6));
            int ax2 = (int) (ex - arrowSize * Math.cos(angle + Math.PI / 6));
            int ay2 = (int) (ey - arrowSize * Math.sin(angle + Math.PI / 6));
            Polygon arrowHead = new Polygon();
            arrowHead.addPoint(ex, ey);
            arrowHead.addPoint(ax1, ay1);
            arrowHead.addPoint(ax2, ay2);
            g.fillPolygon(arrowHead);
        }
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> {
            MaxFlowGUI frame = new MaxFlowGUI();
            frame.setVisible(true);
        });
    }
}



import java.util.*;

public class MaxFlow {
    public int edmondsKarp(int[][] capacity, int s, int t) {
        int n = capacity.length;
        int flow = 0;
        while (true) {
            int[] parent = new int[n];
```

```
        Arrays.fill(parent, -1);
        parent[s] = s;
        Queue<Integer> queue = new LinkedList<>();
        queue.add(s);
        while (!queue.isEmpty() && parent[t] == -1) {
            int u = queue.poll();
            for (int v = 0; v < n; v++) {
                if (capacity[u][v] > 0 && parent[v] == -1) {
                    parent[v] = u;
                    queue.add(v);
                }
            }
        }
        if (parent[t] == -1) break; // no augmenting path
        // minimum capacity (bottleneck)
        int increment = Integer.MAX_VALUE;
        int v = t;
        while (v != s) {
            int u = parent[v];
            increment = Math.min(increment, capacity[u][v]);
            v = u;
        }
        v = t;
        while (v != s) {
            int u = parent[v];
            capacity[u][v] -= increment;
            capacity[v][u] += increment;
            v = u;
        }
        flow += increment;
    }
    return flow;
  }
}
```