Реализация алгоритма Флойда поиска кратчайших путей на графе с использованием технологий ASP.NET Web API и Windows Forms

Подготовил: Акимов А.А., КНиИТ, 341 гр.
Научный руководитель: Доцент кафедры иип, к.ф. – м. н. К. П. Вахлаева
Консультант: Глава лаборатории .NET учебного центра саратовского филиала компании «EPAM Systems» Д. М. Верескун

Цели: пример графа

- изучение технологий работы с клиент-серверным приложением, алгоритма поиска кратчайших путей на большом разреженном графе и наглядная визуализация хода его работы.

Задачи:

- изучение алгоритма Флойда поиска кратчайших путей на графе и его реализация на языке программирования C#;
- изучение технологии ASP.NET Web API;
- реализация Windows Forms приложения с использованием ASP.NET Web API;
- тестирование реализованного приложения на наборе тестов для проверки алгоритмов поиска кратчайших путей.

Основные понятия и определения:

Граф – это пара $G = (V, E)$, где V- множество вершин, E- множество ребер.

Ориентированным называется граф, в котором $E \subseteq V \times V$ – множество упорядоченных пар вершин вида (v_i, v_j), где v_i называется *началом*, а v_j – *концом* дуги.

Для любого графа $G = (V, E)$ полагаем, что множество вершин V имеет мощность: $|V| = n$, а множество ребер E имеет мощность: $|E| = m$.

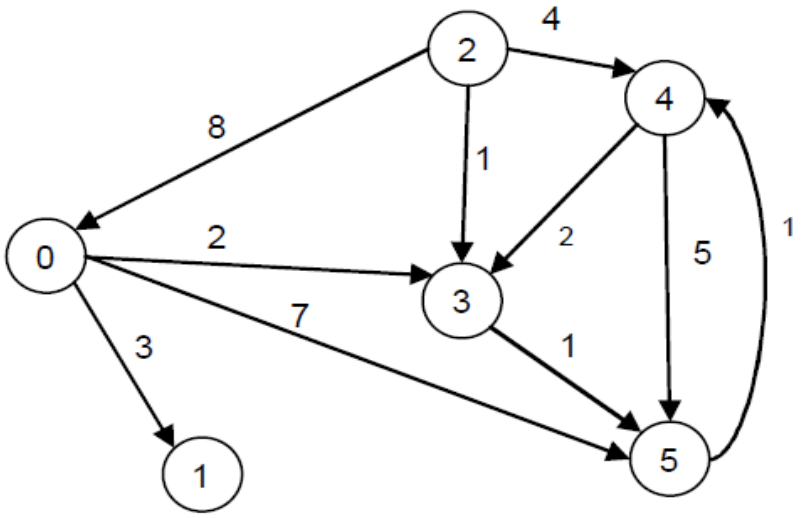
Неориентированным называется граф, в котором $E \subseteq \{(v_i, v_j) : v_i, v_j \in V \& v_i \neq v_j\}$ – множество неупорядоченных пар вершин. Элементы этого множества называются ***ребрами***.

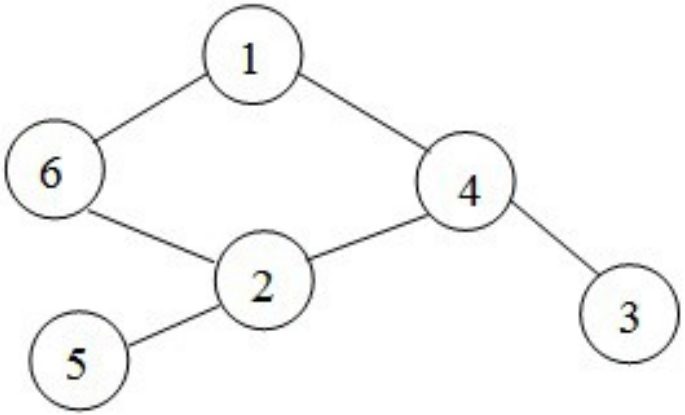
Вершины, соединенные ребром, называются ***смежными***. Ребра, имеющие общую вершину, также называются *смежными*. Ребро и любая из двух его вершин называются ***инцидентными***.

На практике неориентированный граф используется для задания симметричных отношений для объектов.

Граф называется ***взвешенным***, если каждой его дуге (ребру) поставлена в соответствие некоторая числовая характеристика $w_{ij} = w(v_i, v_j)$, $v_i, v_j \in V$, называемая ***весом*** данной дуги.

Примеры графа





Формат хранения графа

- Граф карты дорого Рима задается в файле и имеет текстовый формат. Файл содержит строчки следующих типов:
- Строка с описанием графа, например, "sp 2000 6000", означает, что граф разреженный и содержит 2000 вершин, 6000 ребер.
- Список ребер графа, например, "596 959 78", означает ребро из вершины 596 в вершину 959 с весом 78.
- Граф карты дорог Рима имеет разреженный формат, поэтому для его хранения в оперативной памяти компьютера будет использоваться строчный формат CRS (Compressed Sparse Rows) хранения разреженных матриц

Хранение разреженной матрицы

$$\begin{pmatrix} 2 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 2 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 2 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 2 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 2 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 2 \end{pmatrix}$$

Разреженная матрица

Первый массив хранит значения элементов построчно (строки рассматриваются по порядку сверху вниз), второй – номера столбцов для каждого элемента, а третий заменяет номера строк, используемые в координатном формате, на индекс начала каждой строки.

1				2	
		3	4		
			8		5
	7	1			6

Структура хранения:

1	2	3	4	8	5	7	1	6

Value

0	4	2	3	3	5	1	2	5

Col

0	2	4	4	6	6	9

RowIndex

Разреженный строчный формат хранения

Алгоритм Флойда

```
for(k = 0; k < n; k++)
    for(i = 0; i < n; i++)
        for(j = 0; j < n; j++)
            D[i,j] = min(D[i,j], D[i,k]+D[k,j]);
```

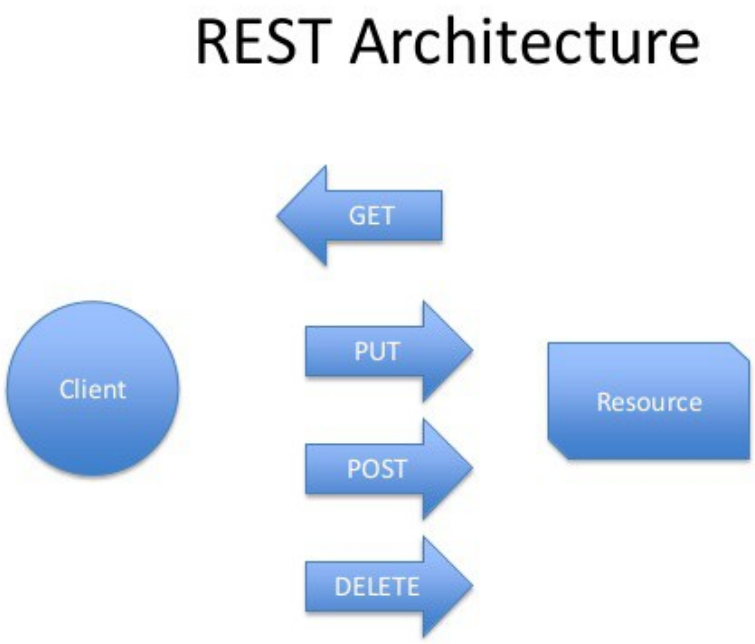
Клиент-серверная архитектура

REST-сервис

- **REST** (**Representational State Transfer** – передача состояния представления)

1. **Клиент-серверная архитектура.** Единый интерфейс между клиентом и сервером.
2. **Отсутствие состояния.** Серверы не связаны с интерфейсами клиентов и их состояниями.
3. **Кэширование.**
4. **Единообразие интерфейса.**
5. **Слои.** Клиент может взаимодействовать не напрямую с сервером, а через промежуточные узлы (слои).

Реализация REST-сервиса с помощью ASP.NET WEB API



JavaScript Object Notation

- При вызове метода GET по URL http://localhost:52566/api/Notes на видим граф, представленный в формате JSON.


```
  "pointerB":
  [
      0,
      2,
      5,
      6,
      9,
      11,
      13,
      14,
      17,
      20,
      21,
      24,
      27,
      28,
      32,
    ...
  "value":
  [
      193,
      2172,
      188,
      193,
      403,
      188,
      2007,
      403,
    ...
  "column":
  [
      1,
      21,
      2,
      0,
      3,
      1,
      20,
    ...
  "sizeV": 3353,
  "sizeE": 8862
}
```

Демонстрация