Algorithms With Java

Week 1 Day 4: Алгоритм Дейкстры

Академия Ковалевского

Содержание

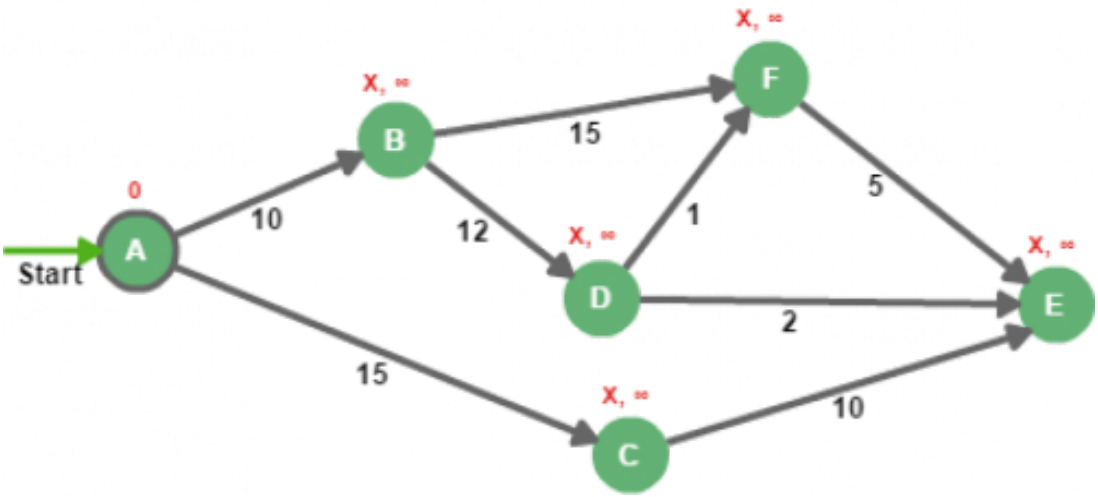
1. Теория

Алгоритм Дейкстры

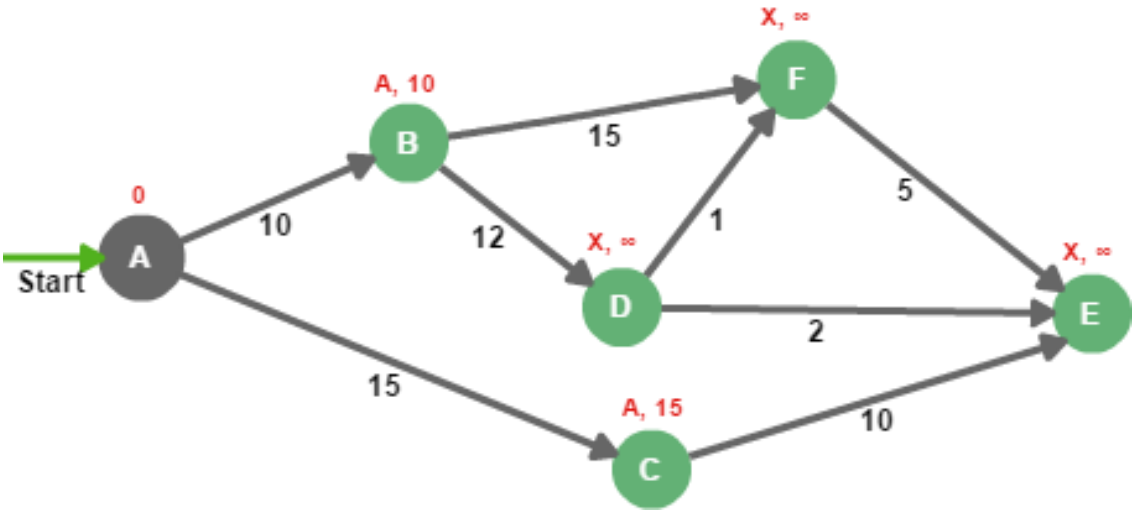
Сегодня жизнь изменится на до и после. Алгоритм Дейкстры — это тот самый молоток после чего все кажется гвоздем. Число задач, которые могут быть решены этим инструментом не влезло бы даже на табло долга США. Знать и уметь его использовать должен каждый, задачи которые решаются этим алгоритмом спрашивают даже если устраиваетесь на должность уборщика. Короче говоря, учить от первого абзаца и до заката солнца, а потом проснуться и снова повторить.

Как вы уже поняли, это один из самых важных алгоритмов, который используется для определения кратчайшего пути от начального узла до всех остальных. Алгоритм называют жадным, потому что он всегда отмечает ближайшую вершину графа.

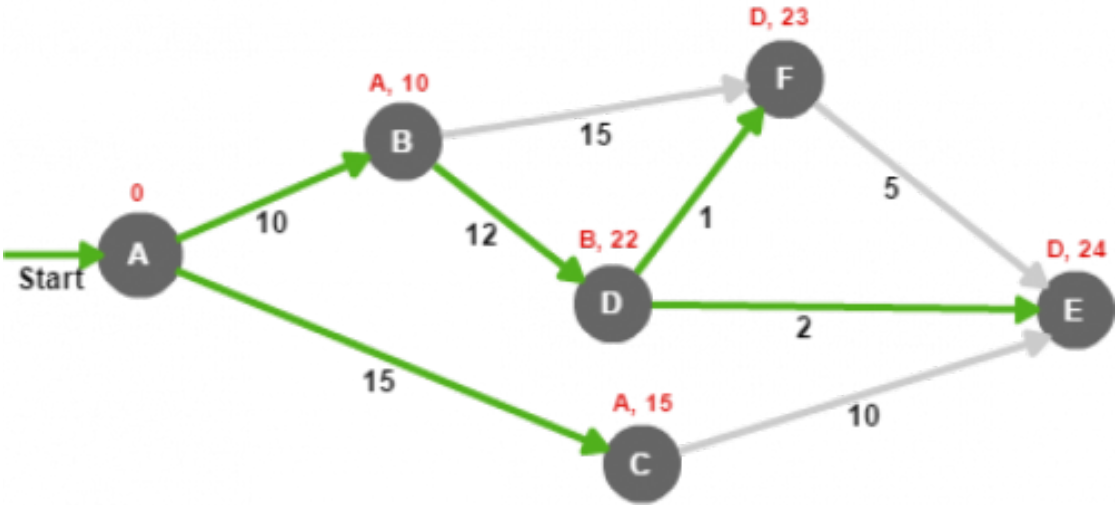
На старте начальному узлу присваивается значение 0, поскольку расстояние от вершины A до A равно 0:



Затем алгоритм оценивает расстояние до всех соседних узлов и ищет вершину с наименьшим расстоянием от изначальной точки:



Этапы повторяются, пока все вершины не будут отмечены:



2. Практическая работа

В данном практическом задании мы применим алгоритм Дейкстры, а также решим интересную задачку с реального собеседования.

Node

Граф в сегодняшнем дне будет описан следующем классом:

```
package academy.kovalevskyi.algorithms.week1.day4;

import java.util.HashMap;
import java.util.Map;

public class Node {
  public Map<Node, Integer> connections = new HashMap<>();
  public Integer distance = Integer.MAX_VALUE;
}
```

Переменная *distance* служит в качестве временной при написании алгоритма Дейкстры.

NetworkPathFinder

В данном классе вам необходимо реализовать поиск кратчайшего пути от вершины графа *start* до вершины *end* используя алгоритм Дейкстры (рис.1).

Давайте разберем пример на рисунке. Допустим, мы имеем такие вершины: *start*, *1*, *2*, *3*, *end*, которые соединены ребрами с указанными весами. Для данного примера кратчайший путь от *start* до *end* будет: 1 + 7 + 100 + 10 = 118

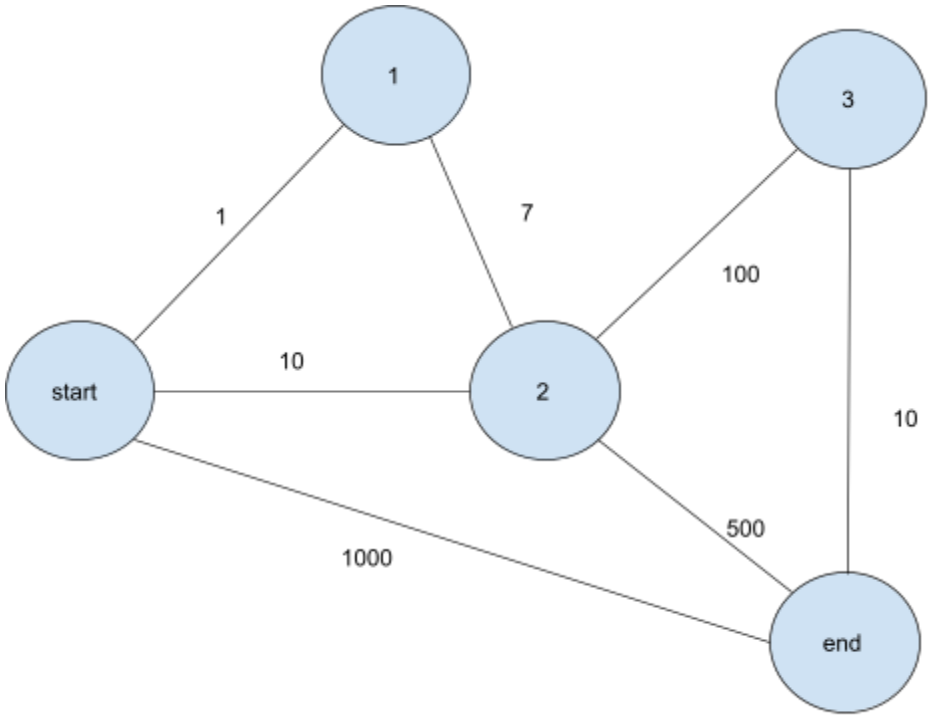


Рис.1

Сигнатура класса NetworkPathFinder() :

```
package academy.kovalevskyi.algorithms.week1.day4;

import java.util.ArrayList;
import java.util.List;

public class NetworkPathFinder {

  public static int shortestPath(Node start, Node end) {
    // TODO
  }
}
```

VirusApocalypse

В этом классе вы должны реализовать при помощи любого из алгоритмов следующую задачку. Дано три значения:

- 0 — здоровый человек
- 1 — человек с вирусом
- 2 — стена, предотвращающая передачу вируса

Каждый день человек с вирусом заражает всех кто находится рядом (вверху, внизу, справа и слева, но не по диагоднали). Пример:

День 1.

0	2	0	2	1
0	2	1	2	0
1	1	0	2	0
0	1	0	2	2

День 2.

0	2	**1**	2	1
1	2	1	2	**1**
1	1	**1**	2	0
1	1	**1**	2	2

День 3.

1	2	1	2	1
1	2	1	2	1
1	1	1	2	**1**
1	1	1	2	2

Результатом работы программы должно быть количество дней, через которое произойдет полное заражение (начиная с 0, если уже все заражены) или *-1* в случае, если полное заражение невозможно. Пример, когда полное заражение невозможно:

1	2	**0**	2	1
1	2	2	2	1
1	1	1	2	1
1	1	1	2	2

Сигнатура класса VirusApocalypse():

```
package academy.kovalevskyi.algorithms.week1.day4;

import java.util.Arrays;

public class VirusApocalypse {

  public static int countDays(Integer[][] startingUniverse) {
    // TODO
  }
}
```