k12338084
YISONG TANG.

Institut für Pervasive Computing

Algorithms and Data Structures 2
Winter term 2024

Assignment 05

Deadline: **Wed. 08.01.2025, 23:59**

Maximal flows in graphs

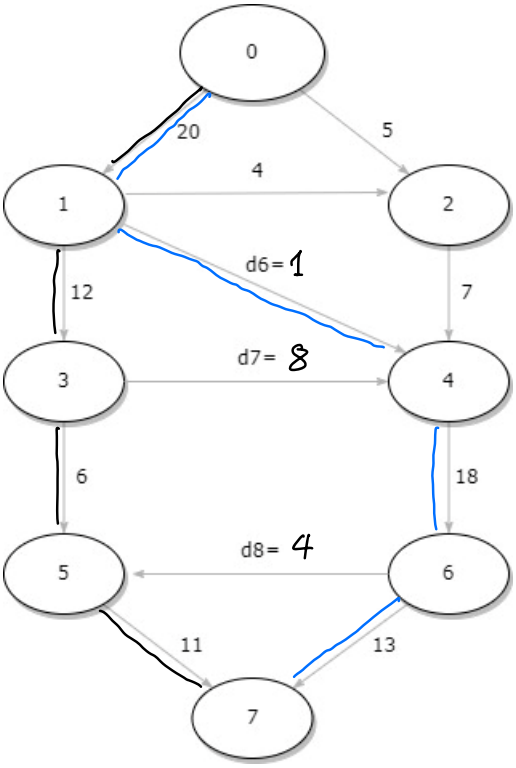
Please don't change any skeleton we provide. Of course, you are allowed to add code in your classes, as long as you don't break the skeleton and their interfaces (method names and variables' types) we need for the unit tests to work correctly. **Please remember to fill out the time log survey in Moodle.**

1. Max Flow — 6 points

Apply the **Ford-Fulkerson MaxFlow** algorithm (from vertex 0 to vertex 7), presented in the exercise using **pen & paper**.

Insert of student ID
Fill in the missing edge capacities using the digits of your student id as follows (if one of the digits is 0, then use 1 instead):

Student id: **k** X X X X X $\underset{d6}{1}$ $\underset{d7}{8}$ $\underset{d8}{4}$

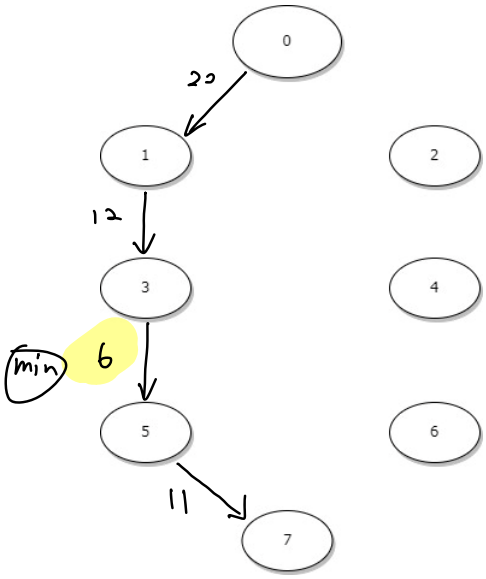


Perform the following tasks on the given network, assuming no existing flows in the beginning.
a) **Complete the residual graph** (i.e., add missing edges) and fill in its capacities.
b) **Find an augmenting path using BFS** (in order, so path [0, 1, 3, 5, 7] comes before [0, 1, 4, 6, 7]) and highlight it with any color and/or bold lines in the residual graph.
c) **Update the current flow** by the maximum possible flow according to augmenting path.
Repeat these tasks in 4 consecutive iterations below:

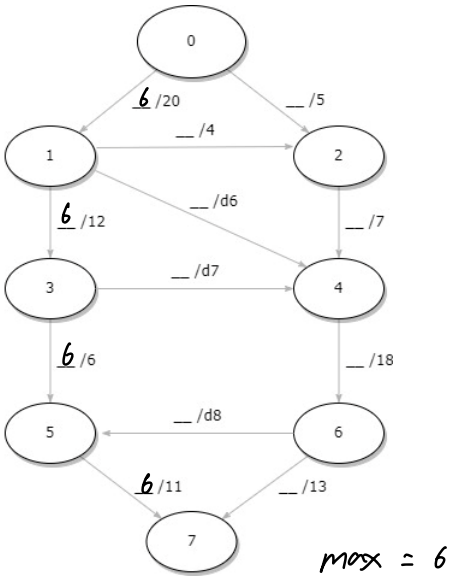
Assignment 05

Deadline: **Wed. 08.01.2025, 23:59**

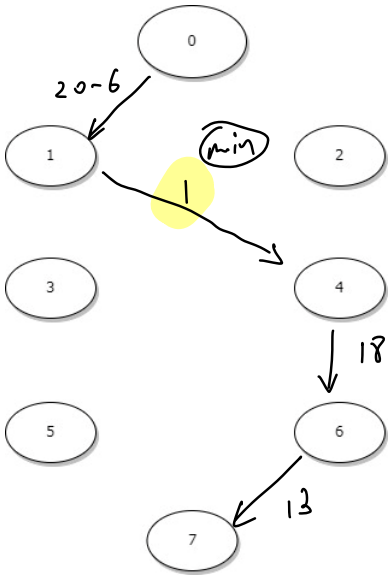
1. **Residual Graph (with augmenting path)**



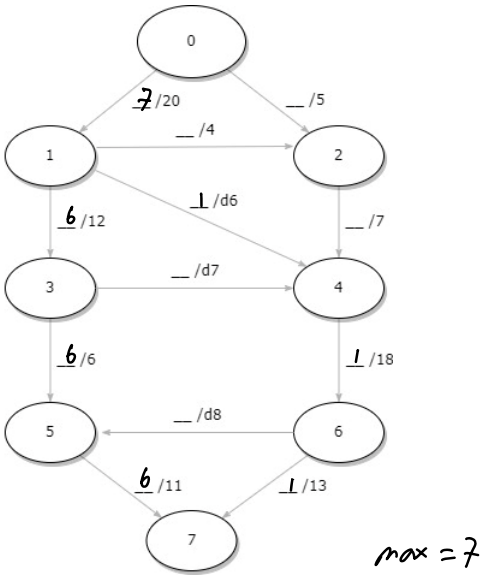
Current Flow



2. **Residual Graph (with augmenting path)**



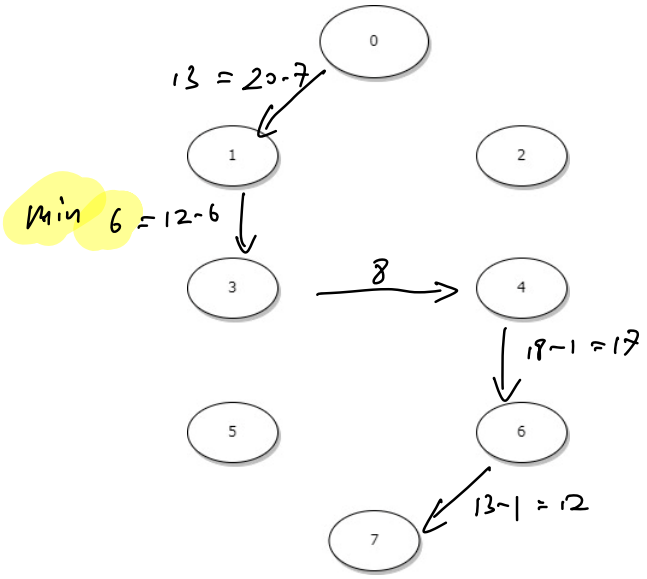
Current Flow



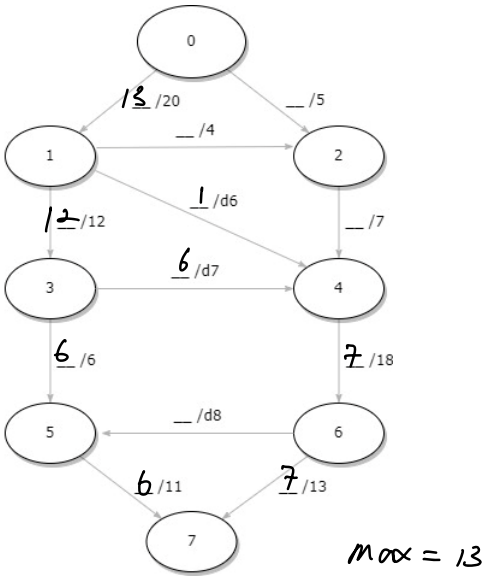
Assignment 05

Deadline: **Wed. 08.01.2025, 23:59**

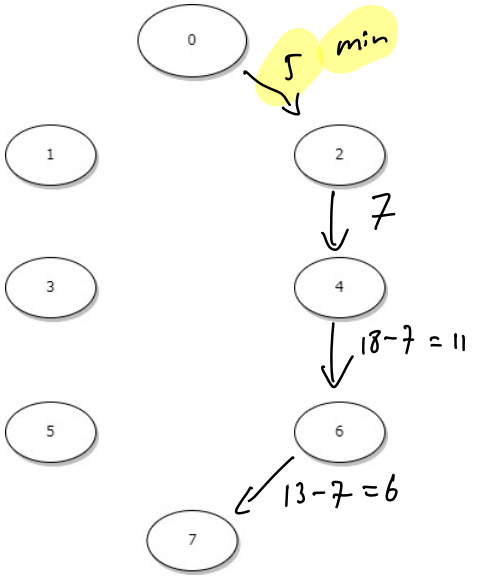
3. **Residual Graph (with augmenting path)**



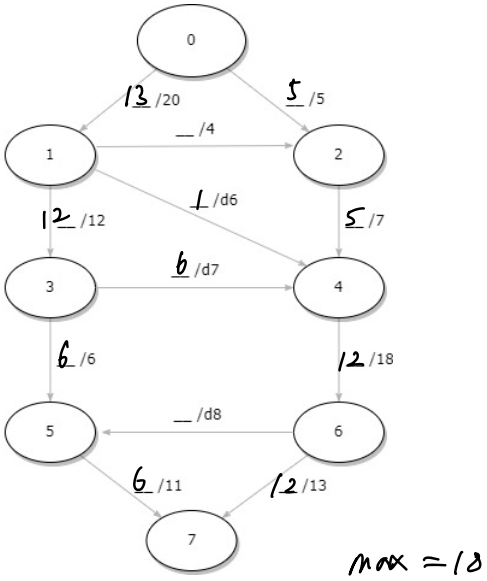
Current Flow



4. **Residual Graph (with augmenting path)**



Current Flow



Assignment 05

Deadline: **Wed. 08.01.2025, 23:59**

2. Ford-Fulkerson Algorithm — 14 points

Implement the Ford-Fulkerson algorithm for finding the maximal flow in a directed graph, using the provided skeleton and adhering to the following guidelines:

- the **graph (`graph`)** is provided in an **adjacency**
- **matrix** form
- the latest **augmenting path (`latest_augmenting_path`)** has to be calculated using **BFS (Breadth First Search – see exercise 4 slides**) on the residual graph and stored during each iteration in an **adjacency matrix** form (**`ff_step`**)

  For example, given a residual graph that consists of six vertices, the path 0 -> 1 -> 3 -> 5 (vertex numbers) with the edge weights 3, 4 and 5 respectively (i.e., the maximal flow increase is 3) would be represented as

```
[
    [0, 3, 0, 0, 0, 0], # vertex 0
    [0, 0, 0, 3, 0, 0], # vertex 1
    [0, 0, 0, 0, 0, 0], # vertex 2
    [0, 0, 0, 0, 0, 3], # vertex 3
    [0, 0, 0, 0, 0, 0], # vertex 4
    [0, 0, 0, 0, 0, 0], # vertex 5
]
```

- the **residual graph (`residual_graph`)**, also in adjacency matrix form, has to be updated and stored at the end of every iteration (**`ff_step`**)
- the total **current flow (`max_flow`)**, also in adjacency matrix form, in the graph has to be updated and stored at the end of every iteration (**`ff_step`**)

```
class Graph:

    def __init__(self, graph):
        self.graph = graph # original graph
        self.residual_graph = [[cell for cell in row] for row in graph] # cloned graph
        self.latest_augmenting_path = [[0 for _ in row] for row in graph] # empty graph with same
dimension as graph
        self.current_flow = [[0 for _ in row] for row in graph] # empty graph with same dimension
as graph


    def ff_step(self, source, sink):
        Perform a single flow augmenting iteration from source to sink. Update the latest
augmenting path, the residual graph and the current flow by the maximum possible amount, according
to the path found by BFS.
        @param source the source's vertex id
        @param sink the sink's vertex id
        @return the amount by which the flow has increased.


    def ford_fulkerson(self, source, sink):
        Execute the ford-fulkerson algorithm (i.e., repeated calls of ff_step())
        @param source the source's vertex id
        @param sink the sink's vertex id
        @return the max flow from source to sink
```

Submission

Submit your **`source code`** as well as your digitized **paper solution** in a ZIP archive and name the file **k12345678-assignment05.zip** where k12345678 should reflect your student ID.