Instructors:
Michael Ritter
Hugo K. Kasuya Rosado

Exercise sheet #5
Combinatorial Optimization
SuSe 2024

Exercises with an asterisk mark "*" are optional.

1 Sending Children to School

Professor Max F. Lous has three children who, unfortunately, dislike each other. The problem is so severe that not only do they refuse to walk to school together, but in fact each one refuses to walk on any path that any of the other children has stepped on that day, although the children have no problem crossing their paths. Thus, the professor is not sure if it is going to be possible to send all of his children to the same school.

a. How can you help the professor? Can his problem be solved in polynomial time?
b. Consider the additional constraint that the children also refuse to cross each other's paths. How can one solve this new problem? Can it be solved in polynomial time?

2 Executing Maximum Flows

Consider the network (G, s, t, u) featured in Exercise sheet #4 on Figure 1.

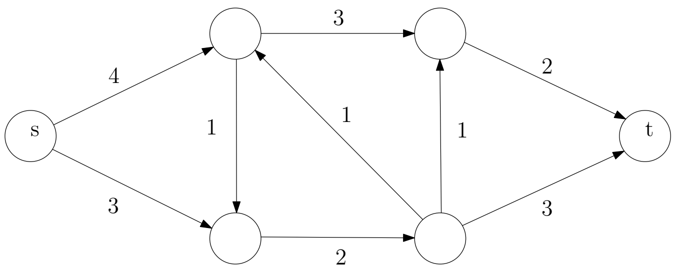

Figure 1: A flow graph with source s and sink t.

a. Execute Push-Relabel algorithm. What is the flow you found? What is its value? How many pushed and relabel operations did you execute?

3 Coding the Push-Relabel algorithm

In this exercise, we will write a program to compute the maximum flow of a network using the Push-Relabel algorithms.

a. Download from Moodle the following files and place them directory/folder: main.py, and graph_module.py.
   (a) file main.py is the file we will be executing in the terminal and it calls functions and methods defined in the other file;
   (b) graph_module.py is the file we will be working on in this exercise. It contains an incomplete implementation of the Push-Relabel algorithms.

   With Python 3 installed, to run the code, simply open the terminal, navigate to the folder containing the files, then run the following command to run the code

```
python3 main.py
```

   Some status messages should appear on the terminal and the execution should enter an infinite loop - this is because an essential part of the code is incomplete. To kill the execution, you may press the shortcut `Ctrl + c` on the terminal window.

b. Open graph_module.py. This file contains the definition of the method `push()` and `relabel()`, which are the method called by `push_relabel()`. The method `push()` has two arguments:

   (a) `v1` and `v2` denoting the edge (`v1`,`v2`) for which flow should be pushed;

   and method `relabel()` has one argument:

   (a) `v` which denotes the vertex for which the label `self.psi[v]` should be modified.

   As you may notice, nothing has been implemented in these functions and this is the reason the execution of the code enters an infinite loop.
   Your task in this exercise is to implement both `push_relabel()` and `relabel()` methods - there is no need to optimize the time complexity of your code. Here is a list of useful variables you could use for the implementation:

   (a) `self.adjacency_matrix` is an adjacency matrix of the residual graph, and `self.adjacency_matrix[v][u]` returns the capacity of the edge (`v`,`u`).
   (b) `self.vertex_list` is a list of vertices and `self.numOfVertices` is the number of vertices in the graph.
   (c) `self.ex_f_list` is a list of the excess flow of each vertex, and `self.ex_f_list[v]` contains the value of the excess flow on vertex `v`.
   (d) `self.psi` is a list of labels, and `self.psi[v]` returns the label of vertex `v`.

c. Uncomment line 18 to generate random graphs, and modify the parameters `num_vertices`, `lower_cap`, `upper_cap` - details on this are written in the comments of the code. Compare the running times of the different maximum flow algorithms.