Test cases for lab 3
DD2350 ADK

15 October 2021

These test cases are meant to assist you with testing and debugging your programs. In practice (in reality) such test cases are normally not available, so an important skill is to be able to create good test cases.

You don't have to run these test cases in order to pass the lab. These test cases are a complement to the test cases built into Kattis. But Kattis won't give any feedback on what is wrong with your program. Therefore we recommend that you initially test your program with these test cases.

Input and output files can be found in the directory

`/afs/kth.se/misc/info/kurser/DD2350/adk21/labb3/testfall`

Note that there typically exist several solutions to these problems, so a direct comparison between the example output and your program output may be misleading. Kattis is more forgiving and accepts all correct solutions.

1 Reduction of bipartite matching to maximum flow

Problem 1.1

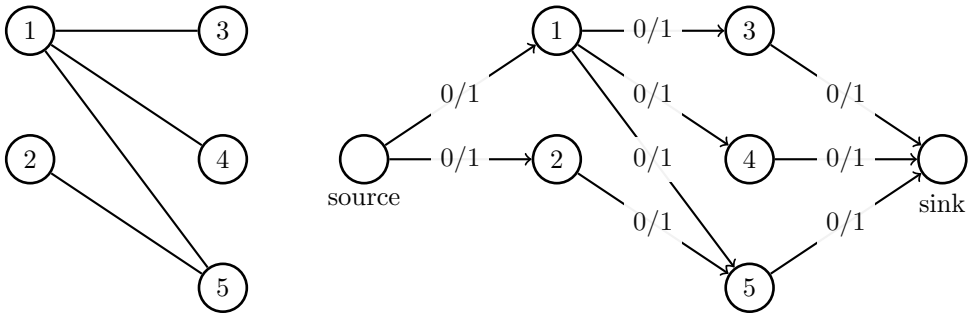
This problem lets you test that you have constructed a correct reduction from the bipartite matching problem to maximum flow. Don't run the maxflow program on this problem, just look at the flow problem instance and compare to the one below. The graphs might differ, especially if you have modified the reduction given in the course somewhat.

Indata

See the figure for visualisation of input and output.
File:
`tillflodetest.indata`.

```
2 3
4
1 3
1 4
1 5
2 5
```

Figure 1: *Input and example output for problem 1.1.*



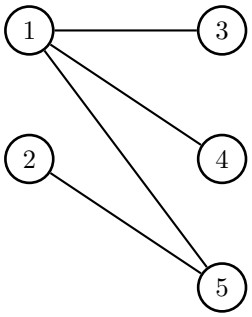
(a) Input

(b) Example output.

Problem 1.2

This continues Problem 1.1. Now the problem should be solved completely giving a maximum matching as output.

Figure 2: *Input for problem 1.2.*



Input

File:
`kapacitetstest.indata`

```
2 3
4
1 3
1 4
1 5
2 5
```

Example output

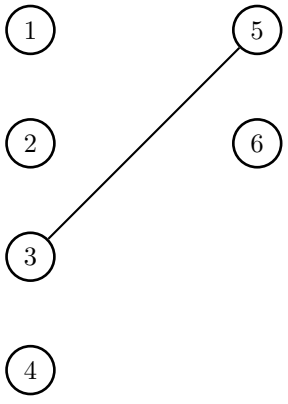
File:
`kapacitetstest.utdata`

```
2 3
2
1 3
2 5
```

Problem 1.3

The following problem tests that your program handles empty vertices and formats the output correctly. Your program should produce output *identically* to one below.

Figure 3: *Input for problem 1.3.*



Input

File:
`formattest.indata`

```
4 2
1
3 5
```

Utdata

File:
`formattest.utdata`

```
4 2
1
3 5
```

2 Solve the maximum flow problem

Problem 2.1

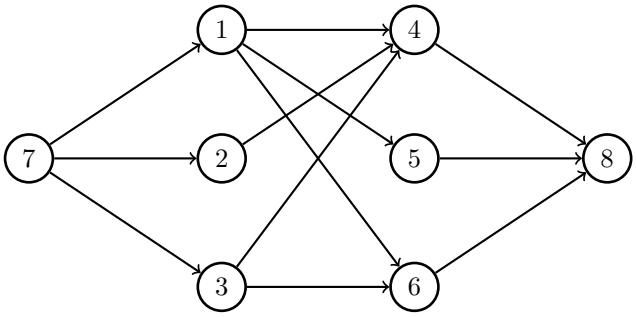
The problem tests the type of flow problem that you will create in Step 3.

Input

File:
`bipartittest.indata`

```
8
7 8
12
1 4 1
1 5 1
1 6 1
2 4 1
3 4 1
3 6 1
7 1 1
7 2 1
7 3 1
4 8 1
5 8 1
6 8 1
```

Figure 4: *Input for problem 2.1, capacities*= 1.



Example output

File:
`bipartittest.utdata`

```
8
7 8 3
9
1 5 1
2 4 1
3 6 1
4 8 1
5 8 1
6 8 1
7 1 1
7 2 1
7 3 1
```

Problem 2.2

For this problem you should print the number of iterations, i.e. the number of *augmenting paths* that the algorithm needs to terminate. A correct solution with breadth-first-search will find a solution using only one augmenting path.

Input

File:
`bfstest.indata`

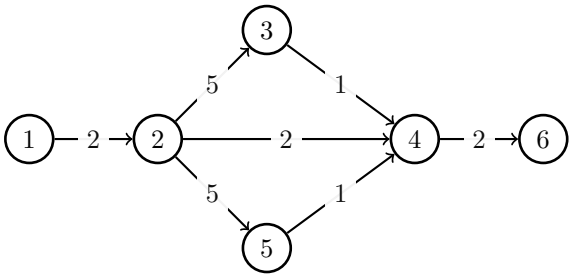
```
6
1 6
7
1 2 2
2 3 5
2 4 2
2 5 5
3 4 1
4 6 2
5 4 1
```

Example output

File:
`bfstest.utdata`

```
6
1 6 2
1 2
2 4
4 6
```

Figure 5: *Input for problem 2.2.*



Problem 2.3

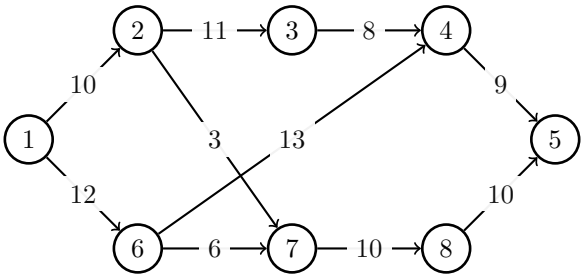
The following example will test that you add and use edges in the residual graph.

Input

File:
`residualtest.indata`

```
8
1 5
10
1 2 10
2 3 11
3 4 8
4 5 9
2 7 3
6 4 13
1 6 12
6 7 6
7 8 10
8 5 10
```

Figure 6: *Input for problem 2.3.*



Example output

File:
`residualtest.utdata`

```
8
1 5 18
10
1 2 6
1 6 12
2 3 3
2 7 3
3 4 3
4 5 9
6 4 6
6 7 6
7 8 9
8 5 9
```

3 Combine Step 1 & 2

Problem 3.1

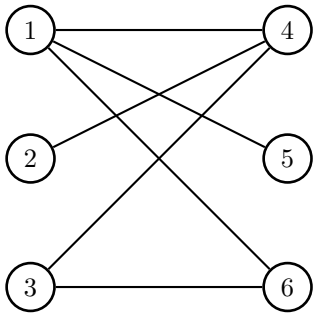
This is the matching variant of a previous problem:

Input

File:
`matchingstest.indata`

```
3 3
6
1 4
1 5
1 6
2 4
3 4
3 6
```

Figure 7: *Input for problem 3.1.*



Example output

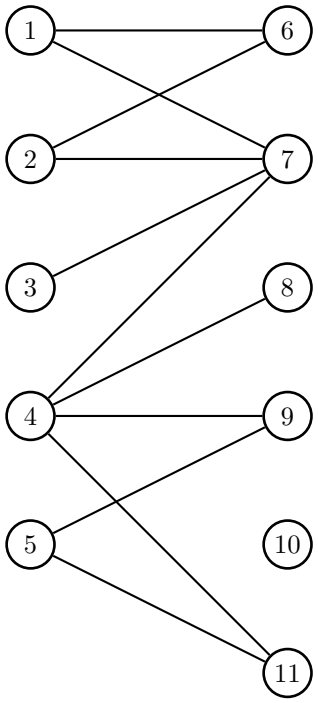
File:
`matchingstest.utdata`

```
3
3
3
1 5
2 4
3 6
```

Problem 3.2

A final, a bit larger, example to test when you have made everything else to work.

Figure 8: *Input for problem 3.2.*



Input

File:
`maffigttest.indata`

```
5 6
11
1 6
1 7
2 6
2 7
3 7
4 7
4 8
4 9
4 11
5 9
5 11
```

Example output

File:
`maffigttest.utdata`

```
5 6
4
2 6
3 7
4 9
5 11
```