

Home work for
Fundamental Algorithms
SS 2007
Sheet 6

Exercise 15: Given a set H of n men, a set D of n women ($D \cap H = \emptyset$) and a sympathy relation $R \subset H \times D$ with $(h, d) \in R$ iff d would not object to marry h , the question is whether D and H can be paired such that every man marries a woman that accepts him. This problem is called **stable marriage problem**.

The following theorem is due to Hall, 1975: Let $N^-(D') = \{h \in H \mid (h, d) \in R \text{ for some } d \in D'\}$. The stable marriage problem (H, D, R) has a solution iff

$$|N^-(D')| \geq |D'| \text{ for all } D' \subseteq D.$$

Prove Hall's theorem using the Max-flow Min-cut Theorem.

Exercise 16: A directed graph $G = (V, E)$ is **bipartite**, if the set of nodes V can be partitioned into two sets X, Y , $X \cup Y = V$, $X \cap Y = \emptyset$, such that $E \subset X \times Y$.

A **matching** M of a graph $G = (V, E)$ is a subset $M \subseteq E$ such that every node $v \in V$ is incident to at most one edge in M .

A **vertex cover** U of a graph G is a subset $U \subseteq V$ such that every edge $e \in E$ is incident to at least one node in U .

König's theorem states that in a bipartite graph G the maximum cardinality of a matching equals the minimum cardinality of a vertex cover.

Prove König's theorem using the Max-flow Min-cut Theorem.

Exercise 17: (Programming task: to be returned by email until June 8th 1:00 pm)

Write a program that, given a flow network (G, s, t, c) in a way as described below, computes a maximum flow using either

- a) the Edmonds-Karp algorithm or
- b) the FIFO preflow push algorithm.

An input/output specification is given on the next page.

To complete this task you must send by email to obelix@upb.de

- 1) The source code of your program
- 2) The command to compile your program (this must be possible on computer `mistel.cs.upb.de` (Linux system))
- 3) The command to start your program. The input file name must be given as the first argument. The second argument is a string from $\{EKA, PPA\}$, where EKA is used to start the Edmond-Karp algorithm, while PPA is used to start the FIFO-Preflow Push algorithm.

Input specification:

```
<comment line>*  
n = <int>  
m = <int>  
s = <int>  
t = <int>  
<int> : <list of edges>  
:  
<int> : <list of edges>
```

where <comment line> is a line starting with #, and <list of edges> is a list of edges separated by a blank (the adjacency lists), and an edge is given as <int>c<int>, where the first integer is the target of the edge, and the second integer is the capacity of the edge. Capacities can be assumed to be from the set $\{1, \dots, 1000000\}$.

An example is given below:

```
# This is a graph with 3 nodes, 5 edges  
n = 3  
m = 5  
s = 0  
t = 2  
0 : 1c11 2c3  
1 : 2c6  
2 : 1c5 0c9
```

Starting command specification:

The command to start the program should be of the form

```
<program> <file> [EKA|PPA]
```

where file is the input file and the second argument is either *EKA* for the execution of the Edmonds-Karp-Algorithm or *PPA* for the execution of the FIFO-Preflow Push Algorithm. E.g. the command

```
flowPusher db-20.txt EKA
```

will start the Edmonds-Karp-Algorithm coded in program flowPusher on input file db-20.txt.

Output specification:

The output should be of the following form:

```
flow <int> <int> <int>  
<list of edges>  
:  
<list of edges>
```

where the first integer after “flow” represents the source s , the second one the sink t , and the third one the value of the flow. Edges carrying flow > 0 are given in form of adjacency lists. Each edge must be of the form <int>f<int>, where the first number represents the to-node of the edge, and the second one is the flow. All adjacency lists, even empty ones, must be given.

For example, for the input given to the left the following output is valid:

```
flow 0 2 9  
0 : 1f6 2f3  
1 : 2f6  
2 :
```

The following output is **not valid**:

```
flow 0 2 9  
0 : 1f6 2f3  
1 : 2f6
```

since the adjacency list of node 2 is missing.