Probe zur Klausur zum Modul *Algorithmische Mathematik I* 16.1.2013

**Name:** **Matrikelnr.:**

**Aufgabe 1 [2+2+2 Punkte]** Geben Sie für folgende Aussagen jeweils ein Beispiel an, das die Aussage beweist, und begründen Sie kurz, warum.

(a) Es gibt eine Instanz des Kürzeste-Wege-Problems (mit teilweise negativen Kantengewichten), bei der Dijkstras Algorithmus nicht die korrekte Lösung findet.

(b) Es gibt eine reelle Zahl, die eine endliche Binärdarstellung hat, aber keine endliche Darstellung zur Basis $b$ für irgendein ungerades $b \geq 3$.

(c) Es gibt eine Matrix, deren Einträge alle ungleich null sind und die keine LU-Zerlegung hat.

**Aufgabe 2 [2+2+2 Punkte]** Welche der folgenden Aussagen gelten? Begründen Sie Ihre Aussage jeweils kurz.

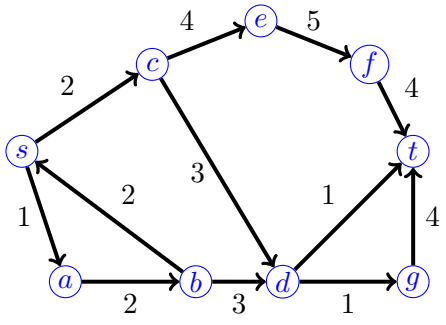
(a) Quicksort ist immer schneller als Merge-Sort.

(b) Addiert man zwei Maschinenzahlen $x$ und $y$ und rundet das Ergebnis zur nächsten Maschinenzahl, so ist der absolute Fehler, den die Rundung verursacht, höchstens $\min\{|x|, |y|\}$.

(c) Das Sieb des Erathosthenes ist ein polynomieller Algorithmus.

**Aufgabe 3 [6 Punkte]** Wie kann man in linearer Zeit entscheiden, ob es in einem gegebenen ungerichteten Graphen zwei Knoten gibt, die durch genau einen Weg verbunden sind?

**Aufgabe 4 [6 Punkte]** Berechnen Sie mit dem Edmonds-Karp-Algorithmus einen $s$-$t$-Fluss mit maximalem Wert in folgendem Netzwerk. Geben Sie das Endergebnis und die Knotenfolgen aller dabei benutzten augmentierenden Wege an.



**Aufgabe 5 [6 Punkte]** Sei $G$ ein nicht-leerer bipartiter Graph mit Bipartition $V(G) = A \dot\cup B$. Es gelte $\min\{|\delta_G(v)| : v \in A\} \geq \max\{|\delta_G(v)| : v \in B\}$. Zeigen Sie, dass $G$ ein Matching enthält, das $A$ überdeckt.

**Aufgabe 6 [10 Punkte]**

Betrachten Sie das folgende Programmstück in C++. Füllen Sie die Lücke (Zeile 18 bis 30) so, dass diese Funktion korrekt eine topologische Ordnung ausgibt, wenn der Digraph azyklisch ist, und andernfalls meldet, dass er einen Kreis enthält.

```
1  int topological_order(Graph const & graph)
2  {
3        std::vector<Graph::NodeId> sorted_nodes;
4        std::vector<int> indegrees(graph.num_nodes());
5
6        for (Graph::NodeId v = 0; v != graph.num_nodes(); v++)
7        {
8              if (graph.get_node(v).in_edges().empty())
9              {
10                   sorted_nodes.push_back(v);
11             }
12             indegrees[v] = graph.get_node(v).in_edges().size();
13       }
14
15       int i = 0;
16       while (i < sorted_nodes.size())
17       {
18
19
20
21
22
23
24
25
26
27
28
29
30
31             i++;
32       }
33
34       if (i < graph.num_nodes())
35       {
36             std::cout << "The digraph contains a circuit.\n";
37             return 1;
38       }
39       else
40       {
41             std::cout << "The graph is acyclic. Topological order:\n";
42             for (Graph::NodeId v = 0; v != graph.num_nodes(); v++)
43             {
44                   std::cout << sorted_nodes[v] << " ";
45             }
46             std::cout << "\n";
47             return 0;
48       }
49 }
```