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) Das Assoziativgesetz gilt für die Addition von Maschinenzahlen in $F_{\texttt{double}}$ nicht.

(b) Es gibt stark zusammenhängende gerichtete Graphen, in denen jeder alle Knoten enthaltende geschlossene Kantenzug mindestens eine Kante mehrfach durchläuft.

(c) Für $\alpha > 0$ sei das eindeutige Berechnungsproblem $f_\alpha : \mathbb{R} \to \mathbb{R}$ durch $f_\alpha(x) := \ln(x^2 + \alpha)$ für alle $x \in \mathbb{R}$ definiert. Dann ist f_α nicht für alle $\alpha > 0$ gut konditioniert.

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

(a) In einem Binärheap steht an der Wurzel immer ein Element mit kleinstem Schlüssel.

(b) Je zwei aufeinanderfolgende positive Fibonacci-Zahlen sind teilerfremd.

(c) Sind p und q natürliche Zahlen, und haben $\frac{1}{p}$ und $\frac{1}{q}$ jeweils eine endliche Binärdarstellung, dann hat auch $\frac{1}{p+q}$ eine endliche Binärdarstellung.

Aufgabe 3 [3+3 Punkte] Ein *Gerüst* eines ungerichteten Graphen G sei eine Menge $F \subseteq E(G)$, so dass $(V(G), F)$ ein Wald ist, aber für alle $e \in E(G) \setminus F$ der Graph $(V(G), F \cup \{e\})$ kein Wald ist. Beweisen Sie:

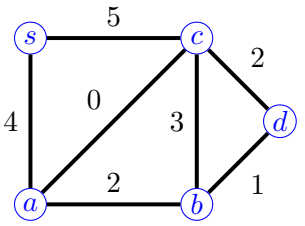
(a) Zu einem gegebenen ungerichteten Graphen G kann man in linearer Zeit ein Gerüst von G finden.

(b) Zu einem gegebenen ungerichteten Graphen G mit n Knoten, m Kanten, und mit Kantengewichten $c : E(G) \to \mathbb{R}$ kann man in $O(n + m \log n)$ Zeit ein Gerüst mit kleinstem Gesamtgewicht finden.

Aufgabe 4 [3+3 Punkte]

(a) Geben Sie eine voll pivotisierte LU-Zerlegung folgender Matrix an: $\begin{pmatrix} 0 & 1 & 2 \\ 0 & 3 & 4 \end{pmatrix}$

(b) Geben Sie einen Kürzeste-Wege-Baum mit Wurzel s in dem ungerichteten Graphen rechts mit den gezeigten Kantengewichten an:



Aufgabe 5 [6 Punkte] Sei G ein einfacher bipartiter Graph mit Bipartition $V(G) = A \dot\cup B$ und $|A| = |B| = k \in \mathbb{N}$. Jeder Knoten in G habe Grad mindestens $\frac{k}{2}$. Zeigen Sie, dass G dann ein perfektes Matching besitzt.

Aufgabe 6 [10 Punkte] Ergänzen Sie das folgende Programmstück in C++ (Zeile 6–28) so, dass es korrekt testet, ob der gegebene gerichtete Graph ein Kreis ist. Ein Ausdruck der aus der Vorlesung bekannten Klasse `Graph` liegt Ihnen vor.

```
1  #include "graph.h"
2
3
4  bool digraph_ist_kreis(Graph const & graph)
5  {
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29 }
30
31
32 int main(int argc, char* argv[])
33 {
34       if (argc > 1)
35       {
36             Graph g = read_graph(argv[1]);
37             print_graph(g);
38             if (digraph_ist_kreis(g))
39             {
40                   std::cout << "Der Digraph ist ein Kreis.\n";
41             }
42             else
43             {
44                   std::cout << "Der Digraph ist kein Kreis.\n";
45             }
46       }
47       return 0;
48 }
```

Aufgabe 1

(a): Für jede Rundung rd zu $F_{\texttt{double}}$ ist $\mathrm{rd}(1+\mathrm{rd}(3\cdot 2^{-55}+3\cdot 2^{-55})) = \mathrm{rd}(1+3\cdot 2^{-54}) = 1+2^{-52} \neq 1 = \mathrm{rd}(1+3\cdot 2^{-55}) = \mathrm{rd}(\mathrm{rd}(1+3\cdot 2^{-55})+3\cdot 2^{-55})$.

(b): Der Digraph $(\{a,b,c,d\}, \{(a,b),(b,c),(b,d),(c,a),(d,a)\})$ ist stark zusammenhängend, aber jeder die Knoten c und d enthaltende geschlossene Kantenzug durchläuft die Kante (a,b) mindestens zweimal.

(c): Die Instanz x des Berechnungsproblems f_α hat Kondition $\frac{2x^2}{(x^2+\alpha)\ln(x^2+\alpha)}$, was für $\alpha < 1$ und $x \approx \sqrt{1-\alpha}$ beliebig groß wird.

Aufgabe 2

(a): wahr: Ist (B,f) ein Binärheap, so ist B Arboreszenz. Für alle Knoten x gibt es einen r-x-Weg P in B, und wegen der Heapordnung ist der Schlüssel von $f(v)$ höchstens der Schlüssel von $f(w)$ für alle Kanten (v,w) von B und somit von P; also ist der Schlüssel von $f(r)$ höchstens der Schlüssel von $f(x)$.

(b): wahr: Für alle $k > 1$ ist $\mathrm{ggT}(F_{k+1}, F_k) = \mathrm{ggT}(F_{k+1}-F_k, F_k) = \mathrm{ggT}(F_{k-1}, F_k)$. Per Induktion über k folgt, dass $\mathrm{ggT}(F_{k+1}, F_k)) = \mathrm{ggT}(F_2, F_1) = 1$ ist.

(c): falsch, denn für $p = 2$ und $q = 4$ ist $\frac{1}{p} = 2^{-1}$ und $\frac{1}{q} = 2^{-2}$, aber $\frac{1}{p+q} = \frac{1}{6} = 2^{-3}(1+2^{-2}+2^{-4}+\cdots)$.

Aufgabe 3

(a): Man bestimme die Zusammenhangskomponenten von G laut Vorlesung in linearer Zeit und finde in jeder einen aufspannenden Baum per Graphendurchmusterung. Die Vereinigung dieser Bäume ist ein Gerüst.

(b): Man füge $n-1$ neue Kanten hinzu, die einen aufspannenden Baum bilden, und deren Gewicht jeweils höher ist als das aller Kanten von G. Auf das Ergebnis wende man Kruskals (oder Prims) Algorithmus an und entferne aus dem berechneten aufspannenden Baum schließlich die Hilfskanten.

Aufgabe 4

(a): Vertauschung der ersten und dritte Spalte und Subtraktion des doppelten der ersten Zeile von der zweiten ergibt:

$$\begin{pmatrix} 0 & 1 & 2 \\ 0 & 3 & 4 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 2 & 1 \end{pmatrix} \begin{pmatrix} 2 & 1 & 0 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

(b): Der Kürzeste-Wege-Baum enthält die Kanten $\{s,a\}$, $\{a,b\}$, $\{a,c\}$ und $\{c,d\}$.

Aufgabe 5

Nach dem Heiratssatz ist zu zeigen, dass $|\Gamma(X)| \geq |X|$ für alle $X \subseteq A$ gilt. Im Falle $X = \emptyset$ oder $\Gamma(X) = B$ ist dies trivial. Sonst ist $\Gamma(B \setminus \Gamma(X)) \subseteq A \setminus X$ und $|\Gamma(B \setminus \Gamma(X))| \geq \frac{k}{2}$, also $|A \setminus X| \geq \frac{k}{2}$ und somit $|X| \leq \frac{k}{2} \leq |\Gamma(X)|$.

Aufgabe 6

```cpp
#include "graph.h"


bool digraph_ist_kreis(Graph const & graph)
{
	if (graph.num_nodes() <= 1)
	{
		return false;
	}

	size_t n = 0;
	Graph::NodeId v = 0;

	while ((n == 0) or (v != 0))
	{
		if ((graph.get_node(v).in_edges().size() != 1) or
		    (graph.get_node(v).out_edges().size() != 1))
		{
			return false;
		}
		else
		{
			v = graph.get_edge(graph.get_node(v).out_edges()[0]).get_head();
			++n;
		}
	}

	return (n == graph.num_nodes());
}


int main(int argc, char* argv[])
{
	if (argc > 1)
	{
		Graph g = read_graph(argv[1]);
		print_graph(g);
		if (digraph_ist_kreis(g))
		{
			std::cout << "Der Digraph ist ein Kreis.\n";
		}
		else
		{
			std::cout << "Der Digraph ist kein Kreis.\n";
		}
	}
	return 0;
}
```