Wintersemester 2024/25
Professor Dr. V. Traub
Dr. U. Brenner

Forschungsinstitut für Diskrete Mathematik
Universität Bonn

Algorithmische Mathematik I

Übungszettel 1

1. Es seien X, Y, Z drei Mengen und $f : X \to Y$ und $g : Y \to Z$ zwei Abbildungen. Es sei $g \circ f$ die Hintereinanderausführung von f und g, also $g \circ f(x) = g(f(x))$ für alle $x \in X$. Welche dieser Aussagen folgen aus diesen Voraussetzungen?

   (a) Wenn f und g injektiv sind, dann ist auch $g \circ f$ injektiv.

   (b) Wenn $g \circ f$ injektiv ist, dann ist f injektiv.

   (c) Wenn $g \circ f$ injektiv ist, dann ist g injektiv.

   (d) Wenn f und g surjektiv sind, dann ist auch $g \circ f$ surjektiv.

   (e) Wenn $g \circ f$ surjektiv ist, dann ist f surjektiv.

   (f) Wenn $g \circ f$ surjektiv ist, dann ist g surjektiv.

   Begründen Sie Ihre Antworten. (1+1+1+1+1+1 Punkte)

2. Es seien m und n zwei natürliche Zahlen. Außerdem seien $A := \{1, \ldots, m\}$ und $B := \{1, \ldots, n\}$. Bestimmen Sie in Abhängigkeit von m und n die Zahl der ...

   (a) Funktionen von A nach B.

   (b) injektiven Funktionen von A nach B.

   (c) bijektiven Funktionen von A nach B.

   (d) Relationen auf (A, B). (1+1+1+1 Punkte)

3. Zeigen Sie, dass $\mathbb{N} \times \mathbb{N}$ abzählbar ist. (5 Punkte)

4. Schreiben Sie ein C++-Programm, das drei positive ganze Zahlen einliest, welche die Breite, Höhe und Tiefe eines Quaders angeben sollen. Geben Sie anschließend das Volumen des Quaders aus. (5 Punkte)

Abgabe: Bis Mittwoch, 16.10., 23:59 Uhr, auf der eCampus-Seite der eigenen Gruppe.

Informationen zur Vorlesung und zu den Übungen:

- Die Vorlesung ist montags 10 – 12 und donnerstags 8 – 10 Uhr in CP1-HSZ / Hörsaal 1, Friedrich-Hirzebruch Allee 5. Sie beginnt jeweils c.t., also eine Viertelstunde nach der vollen Stunde.
- Informationen zur Vorlesung und zu den Übungen gibt es auf der eCampus-Seite der Vorlesung: `https://ecampus.uni-bonn.de/goto.php?target=crs_3494733&client_id=ecampus`
- Die Übungen sind vierstündig, davon zwei Stunden Theorie und zwei Stunden Programmierung.
- Der Übungsbetrieb beginnt am 14.10.2024.
- Jede Woche gibt es donnerstags morgens einen neuen Übungszettel (auf der eCampus-Seite der Vorlesung).
- Die Bearbeitungen zu den Aufgaben sind (wenn nicht anders angegeben) bis zum folgenden Mittwoch um 23:59 Uhr auf der eCampus-Seite der Übungen hochzuladen. Eine verspätete Abgabe ist nicht möglich.
- Bei der Bearbeitung der Aufgaben ist eine Abgabe in Gruppen von bis zu zwei Personen erlaubt. Größere Gruppen sind nicht zugelassen. Alle Mitglieder einer Abgabegruppe müssen zu derselben Übungsgruppe gehören und in der Lage sein, alle abgegebenen Lösungen zu erklären.
- Die Lösungen zu den Programmieraufgaben müssen in C++ geschrieben werden. Sie müssen mit dem g++-Compiler (der z.B. auf den Rechnern im PC-Pool installiert ist) mit dem Befehl

  ```
  g++ -std=c++11 -Wall -Wpedantic -o EXECUTABLE QUELLDATEI
  ```

  kompiliert werden können (wobei `EXECUTABLE` der Name des ausführbaren Programms und `QUELLDATEI` der Name der zu kompilierenden Datei ist).
- Als zusätzliches Angebot gibt es ein sogenanntes Help Desk als offenes Lehrangebot für Fragen zur Vorlesung und zu den Übungen. Informationen zum Help Desk finden sich hier: `https://www.mathematics.uni-bonn.de/studium/medienordner-studium-1/dateien/listen/helpdesk.pdf`
- Zulassungskriterien zur Modulprüfung:
  - Regelmäßige aktive Teilnahme an den Übungen, einschließlich mehrmaliger Präsentation von Lösungen.
  - Mindestens 50 % der erreichbaren Punkte in den Theorieübungen.
  - Mindestens 50 % der erreichbaren Punkte in den Programmierübungen.

  Alle drei Bedingungen müssen erfüllt sein.

Wintersemester 2024/25
Professor Dr. V. Traub
Dr. U. Brenner

Forschungsinstitut für Diskrete Mathematik
Universität Bonn

Algorithmische Mathematik I

Übungszettel 2

1. Beweisen Sie die folgenden Aussagen:

   (a) Für alle Funktionen $f, g : \mathbb{N} \to \mathbb{R}_{\geq 0}$ gilt: $f = O(g) \Leftrightarrow g = \Omega(f)$.

   (b) $(\log n)^k = O(n^{\frac{1}{l}})$ für alle $k, l \in \mathbb{N}$.

   (c) $\log_2(n!) = \Theta(n \log_2 n)$. (2+2+2 Punkte)

   Hinweis: Sie dürfen Schulwissen über Ableitungen verwenden.

2. Es sei $z \in \mathbb{R}_{\geq 0}$ eine Konstante. Zeigen Sie, dass dann gilt:

$$\sum_{i=1}^{n} i^z = \Theta(n^{1+z}).$$

(4 Punkte)

3. Die Potenzmenge einer Menge X ist die Menge aller Teilmengen von X. Zeigen Sie, dass die Potenzmenge von $\mathbb{N}$ überabzählbar ist. (5 Punkte)
   Hinweis: Verfahren Sie ähnlich wie in dem Beweis aus der Vorlesung, dass es Funktionen gibt gibt, die von keinem C++-Programm berechnet werden.

4. Schreiben Sie ein C++-Programm, das alle Primzahlzwillinge (also Paare von Primzahlen, deren Differenz 2 ist) auflistet, die kleiner als 10^6 sind. Wie viele gibt es? (5 Punkte)

Abgabe: Bis Mittwoch, 23.10., 23:59 Uhr, auf der eCampus-Seite der eigenen Gruppe.

Wintersemester 2024/25
Professor Dr. V. Traub
Dr. U. Brenner

Forschungsinstitut für Diskrete Mathematik
Universität Bonn

Algorithmische Mathematik I

Übungszettel 3

1. Ersetzen Sie bei der Berechnung der Collatz-Folge die Anweisung `n = 3 * n + 1;` durch

   (a) `n = n + 1;`. Zeigen Sie, dass das Programm dann stets terminiert, und geben Sie (mit Hilfe der O-Notation) eine möglichst gute Schranke für die Zahl der Rechenschritte an.

   (b) `n = n + 3;`. Für welche Startwerte terminiert das Verfahren dann? Beweisen Sie die Korrektheit Ihrer Antwort. (3+2 Punkte)

2. Seien n_1 und n_2 zwei natürliche Zahlen mit identischer Ziffernfolge $z_{l-1}z_{l-2}\dots z_0$ bezüglich unterschiedlicher Basen b_1 und b_2. Welche der folgenden Aussagen sind wahr? Begründen Sie Ihre Antwort!

   (a) Falls $b_1 > b_2$, so ist $n_1 > n_2$.

   (b) Falls $n_1 > n_2$, so ist $b_1 > b_2$.

   (c) Falls b_1 Teiler von b_2 ist, so ist n_1 Teiler von n_2.

   (d) Falls n_1 Teiler von n_2 ist, so ist b_1 Teiler von b_2. (1+1+1+1 Punkte)

3. Es sei $(a_{l-1}a_{l-2}\dots a_0)_{-10} := \sum_{i=0}^{l-1} a_i(-10)^i$, wobei $a_i \in \{0,\dots,9\}$ sei für $i \in \{0,\dots,l-1\}$. Man nennt $a_{l-1}\dots a_0$ Darstellung von $\sum_{i=0}^{l-1} a_i(-10)^i$ zur Basis -10, falls $a_{l-1} \neq 0$ gilt oder $l = 1$ und $a_0 = 0$ gelten.

   (a) Schreiben Sie $(19375573910)_{-10}$ als Dezimalzahl.

   (b) Geben Sie eine Darstellung von $(9230753)_{10}$ zur Basis -10 an.

   (c) Zeigen Sie, dass es für jede ganze Zahl x eine Darstellung zur Basis -10 gibt.

   (d) Ist die Darstellung aus Aufgabenteil (c) immer eindeutig? (1+1+2+2 Punkte)

4. Schreiben Sie ein C++-Programm, das für jede Zahl $n \in \{1,\dots,1000000\}$ die Länge der in n beginnenden Collatz-Folge berechnet und anschließend ein $n^* \in \{1,\dots,1000000\}$ ausgibt, für das die in n^* beginnende Collatz-Folge am längsten ist. Geben Sie auch die Länge der in n^* beginnenden Collatz-Folge an. Achten Sie darauf, mögliche Überläufe durch geeignete Abfragen abzufangen. Bei welcher Zahl n genügt der Datentyp `short` erstmals nicht mehr? (5 Punkte)

Abgabe: Bis Mittwoch, 30.10., 23:59 Uhr, auf der eCampus-Seite der eigenen Gruppe.

Wintersemester 2024/25
Professor Dr. V. Traub
Dr. U. Brenner

Forschungsinstitut für Diskrete Mathematik
Universität Bonn

Algorithmische Mathematik I

Übungszettel 4

1. In dieser Aufgabe betrachten wir Komplementdarstellungen zur Basis 2.

   (a) Schreiben Sie die Zahl -25 in Komplementdarstellung mit 8 Bits und mit 16 Bits.

   (b) Sei z die Komplementdarstellung einer negativen Zahl mit l Bits. Welche Zahl entsteht, wenn man in z jede 0 durch eine 1 und jede 1 durch eine 0 ersetzt?

   (c) Für welche negativen Zahlen x ist die Komplementdarstellung mit l Bits bis auf die erste Stelle identisch mit der Komplementdarstellung von $-x$?

   (d) Sei z die Komplementdarstellung einer negativen Zahl mit l Bits. Wie sieht die Darstellung derselben Zahl aus, wenn $2l$ Stellen für die Komplementdarstellung zur Verfügung stehen? *Bemerkung:* Eine solche Umwandlung kann zum Beispiel notwendig werden, wenn eine Variable vom Typ `short` in eine Variable vom Typ `long` umgewandelt wird. (1+1+1+1 Punkte)

2. Zeigen Sie, dass man mit Hilfe von Karatsubas Algorithmus zwei natürliche Zahlen mit k- bzw. l-stelliger Binärdarstellung (mit $k \leq l$) in $O(k^{\log_2(3)-1}l)$ Zeit multiplizieren kann. (5 Punkte)

3. Geben Sie einen Algorithmus in Pseudocode an, der zu gegebenen natürlichen Zahlen a und b die Zahl a^b berechnet, und der nur $O(\log_2 b)$ elementare Rechenoperationen (`+, -, *, /, %`) durchführt. Beweisen Sie, dass Ihr Algorithmus tatsächlich diese Eigenschaften hat. (6 Punkte)

4. Implementieren Sie Subtraktion, Multiplikation und Division in der Klasse `Fraction`. (5 Punkte)

Abgabe: Bis Mittwoch, 6.11., 23:59 Uhr, auf der eCampus-Seite der eigenen Gruppe.

Wintersemester 2024/25
Professor Dr. V. Traub
Dr. U. Brenner

Forschungsinstitut für Diskrete Mathematik
Universität Bonn

Algorithmische Mathematik I

Übungszettel 5

1. (a) Zeigen Sie, dass es für alle natürlichen Zahlen m und n ganze Zahlen s und t gibt, für die $\mathrm{ggT}(n,m) = s \cdot m + t \cdot n$ gilt.

   (b) Geben Sie für $m = 35$ und $n = 84$ eine Darstellung wie in (a) an. (4+1 Punkte)

2. Beweisen Sie die Korrektheit der folgenden Funktion zur Berechnung des größten gemeinsamen Teilers zweier natürlicher Zahlen. Diese Funktion benutzt nur die Division durch 2 (die in Binärdarstellung natürlich sehr einfach und schnell realisierbar ist). Welche Laufzeit hat diese Funktion? (4 Punkte)

```
template <typename IntType>
IntType  gcd2 ( IntType a , IntType b ) // compute greatest commondivisor
{                                       // ("binary Euclidean algorithm ")
  if ( b == 0) return a ;
  else if ( a % 2 == 0) {
    if ( b % 2 == 0) {return 2 * gcd2( b / 2 , a / 2);}
    else {return gcd2 (b , a / 2);}
  }
  else {
    if ( b % 2 == 0) {return  gcd2(a, b / 2);}
    else if ( a < b ) {return  gcd2(a, ( b - a ) / 2);}
    else {return gcd2(b, ( a - b ) / 2);}
  }
}
```

3. Schreiben Sie die Zahlen 125,125 und 99,9 und $\frac{1}{7}$ jeweils in normalisierter 2-adischer Darstellung. (3 Punkte)

4. Was ist die kleinste natürliche Zahl, die nicht in $F_{\texttt{double}}$ ist? (3 Punkte)

5. Wir betrachten einen Ein-Byte-Rechner mit Gleitkomma-Arithmetik. Bei der Zahlendarstellung werden ein Bit für das Vorzeichen, vier Bits für die Mantisse und drei Bits für den Exponenten (mit denen die Exponenten -2, -1, 0, 1, 2 und 3 kodiert werden können) verwendet. Die führende Eins in der Mantisse wird nicht abgespeichert. Schreiben Sie ein Programm, das alle Zahlen, die in dieser Gleitkomma-Arithmetik darstellbar sind, aufsteigend sortiert als Dezimalzahlen ausgibt. (5 Punkte)

Abgabe: Bis Mittwoch, 13.11., 23:59 Uhr, auf der eCampus-Seite der eigenen Gruppe.

Wintersemester 2024/25
Professor Dr. V. Traub
Dr. U. Brenner

Forschungsinstitut für Diskrete Mathematik
Universität Bonn

Algorithmische Mathematik I

Übungszettel 6

1. Sei $F = F(b, m, E_{\min}, E_{\max})$ ein Maschinenzahlbereich mit $m > 1$.

   (a) Man zeige, dass $\text{eps}(F) \notin F$.

   (b) Sei rd eine Rundung zu F, und sei $x \in \text{range}(F)$. Zeigen Sie, dass dann ein $\varepsilon \in \mathbb{R}$ existiert mit $|\varepsilon| \leq \text{eps}(F)$ und $\text{rd}(x) = x \cdot (1 + \varepsilon)$. (3+3 Punkte)

2. Betrachten Sie folgendes Problem: Es sei eine stetige Funktion $f : [0, 1] \to \mathbb{R}$ gegeben, so dass für alle $x, y, \alpha \in [0, 1]$ gilt: $f(\alpha x + (1 - \alpha)y) \leq \alpha f(x) + (1 - \alpha)f(y)$. Die Funktion sei über ein Orakel gegeben, das zu einem beliebigen Wert $x \in [0, 1]$ den Wert $f(x)$ ausgibt. Außerdem sei ein $\epsilon > 0$ gegeben. Gesucht ist ein $x^* \in [0, 1]$, für das es ein $\tilde{x}$ mit $|x^* - \tilde{x}| < \epsilon$ gibt, so dass für alle $x \in [0, 1]$ gilt: $f(\tilde{x}) \leq f(x)$. Zeigen Sie, dass $O(\lceil \log\left(\frac{1}{\epsilon} + 1\right) \rceil)$ Abfragen von Funktionswerten reichen, um ein solches x^* zu berechnen. Sie können in dieser Aufgabe annehmen, dass Sie in $\mathbb{R}$ ohne Rundungsfehler rechnen können. (5 Punkte)

   Hinweis: Modifizieren Sie die binäre Suche geeignet.

3. Schreiben Sie die folgenden Ausdrücke so um, dass für die angegebenen Argumente Auslöschung vermieden wird:

   (a) $\frac{1}{x} - \frac{1}{x+1}$ für $x \gg 1$ (d.h. x ist wesentlich größer als 1).

   (b) $\sqrt[3]{1 + x} - 1$ für $x \approx 0$.

   (c) $\frac{1-\cos x}{\sin x}$ für $x \approx 0$.

   (d) $\frac{1}{x-\sqrt{x^2-1}}$ für $x \gg 1$. (1+1+1+1 Punkte)

4. Implementieren Sie ein C++-Programm, das eine natürliche Zahl $n \in \mathbb{N}$ einliest und im Datentyp `double` die Kubikwurzel $n^{1/3}$ mittels binärer Suche mit Maschinengenauigkeit ausrechnet und dann ausgibt. Hierzu sollten Sie die Ausgabegenauigkeit mittels `std::cout « std::setprecision(16);` aus der Standardbibliothek `iomanip` (`#include <iomanip>`) erhöhen. (5 Punkte)

Abgabe: Bis Mittwoch, 20.11., 23:59 Uhr, auf der eCampus-Seite der eigenen Gruppe.

Wintersemester 2024/25
Professor Dr. V. Traub
Dr. U. Brenner

Forschungsinstitut für Diskrete Mathematik
Universität Bonn

Algorithmische Mathematik I

Übungszettel 7

1. Berechnen Sie die Kondition der folgenden Funktionen und geben Sie an, wo die Funktionsauswertung qualitativ gut bzw. schlecht konditioniert ist.

   (a) $f : \mathbb{R}_{\geq 0} \to \mathbb{R}$ mit $f(x) = \sqrt[3]{x}$,

   (b) $f : \mathbb{R} \to \mathbb{R}$ mit $f(x) = a^x = e^{x \ln a}$ für festes $a > 0$,

   (c) $f : \mathbb{R} \setminus \{0\} \to \mathbb{R}$ mit $f(x) = \frac{1}{x}$

   (d) $f : \mathbb{R} \to \mathbb{R}$ mit $f(x) = \ln(1 + |x - 1|)$ (1+1+1+1 Punkte)

2. Man kann die Quadratwurzel einer Zahl $a \geq 0$ auch mit dem Newtonverfahren angewandt auf $f(x) = 1 - \frac{a}{x^2}$ berechnen. Dabei können wir uns auf Eingaben a mit $1 \leq a < 4$ und den Startwert $x_0 = 1$ beschränken.

   (a) Wie sieht eine Iteration dieses Verfahrens aus? Berechnen Sie x_1, x_2, x_3 für $a = 3$ und $x_0 = 1$.

   (b) Beweisen Sie, dass auch diese Variante quadratisch konvergiert.

   (c) Ist das Verfahren besser als das babylonische Wurzelziehen? (2+3+2 Punkte)

3. Es sei G ein einfacher ungerichteter Graph, in dem jeder Knoten mindestens einen Nachbarn hat. Zeigen Sie, dass G einen Knoten v enthält mit

$$\frac{1}{|\delta(v)|} \sum_{w \in N(v)} |\delta(w)| \;\geq\; \frac{2|E(G)|}{|V(G)|}.$$

   Dabei ist $N(v)$ die Menge der Nachbarn von v. (4 Punkte)

4. Erweitern Sie das C++-Programm aus der Programmieraufgabe von Zettel 6, so dass es die Kubikwurzel $\sqrt[3]{n}$ zusätzlich mit zwei Newtonverfahren angewandt auf $f(x) = x^3 - n$ und auf $f(x) = 1 - \frac{n}{x^3}$ (analog zu Aufgabe 2) mit einer Genauigkeit von 15 Dezimalstellen ausrechnet und dann ausgibt. Geben Sie für jedes der 3 Verfahren das Ergebnis sowie die Anzahl der Iterationen aus. (5 Punkte)

Abgabe: Bis Mittwoch, 27.11., 23:59 Uhr, auf der eCampus-Seite der eigenen Gruppe.

Wintersemester 2024/25
Professor Dr. V. Traub
Dr. U. Brenner

Forschungsinstitut für Diskrete Mathematik
Universität Bonn

Algorithmische Mathematik I

Übungszettel 8

1. Beweisen Sie, dass eine Folge natürlicher Zahlen $d_1, \ldots, d_n$ mit $d_1 \geq d_2 \geq \cdots \geq d_n$ genau dann die Gradfolge eines einfachen ungerichteten Graphen ist (d.h. es gibt einen Graph G mit $V(G) = \{v_1, \ldots, v_n\}$ und $|\delta(v_i)| = d_i$ für alle $i \in \{1, \ldots, n\}$), wenn $d_2 - 1, d_3 - 1, \ldots, d_{d_1+1} - 1, d_{d_1+2}, \ldots, d_n$ dies ist. (5 Punkte)

2. Es sei G ein Baum und $A, B \subseteq V(G)$, $A \cap B \neq \emptyset$, so dass $G[A]$ und $G[B]$ Bäume sind. Zeigen Sie, dass $G[A \cap B]$ ein Baum ist. (5 Punkte)

3. Sei G ein Baum mit n Knoten und $n \geq 2$. Zeigen Sie:

   (a) G hat einen Knoten v, so dass keine Zusammenhangskomponente von $G - v$ mehr als $\frac{n}{2}$ Knoten enthält.

   (b) G hat genau $2 + \sum_{v \in V(G)} \max\{0, |\delta(v)| - 2\}$ Blätter. (3+3 Punkte)

4. Seien (V, F_1) und (V, F_2) zwei Wälder mit $|F_1| < |F_2|$. Man beweise, dass es eine Kante $e \in F_2 \setminus F_1$ gibt, so dass $(V, F_1 \cup \{e\})$ ein Wald ist. (4 Punkte)

Abgabe: Bis Mittwoch, 4.12., 23:59 Uhr, auf der eCampus-Seite der eigenen Gruppe.

Wintersemester 2024/25
Professor Dr. V. Traub
Dr. U. Brenner

Forschungsinstitut für Diskrete Mathematik
Universität Bonn

Algorithmische Mathematik I

Übungszettel 9

1. Beweisen Sie die folgende Aussage: Wenn G ein stark zusammenhängender gerichteter Graph ist, dessen zugrundeliegender ungerichteter Graph mindestens einen Kreis ungerader Länge enthält, dann enthält G auch einen gerichteten Kreis ungerader Länge. (5 Punkte)

2. Beweisen Sie die folgenden Aussagen:

   (a) Seien (V, F_1) und (V, F_2) zwei Branchings mit $2|F_1| < |F_2|$. Dann gibt es eine Kante $e \in F_2 \setminus F_1$, so dass $(V, F_1 \cup \{e\})$ ein Branching ist.

   (b) Die Aussage aus (a) wird falsch, wenn man die Bedingung „$2|F_1| < |F_2|$“ durch „$2|F_1| \leq |F_2|$“ ersetzt. (3+2 Punkte)

3. In einem Baum T bezeichne $\text{dist}_T(x, y)$ für je zwei Knoten $x, y \in V(T)$ die Länge des x-y-Weges in T. Geben Sie ein Verfahren für das folgende Problem an: Zu einem gegebenen Baum T hat man Zeit $O(|V(T)|)$ für ein Präprozessing. Danach soll, wenn zwei Knoten x und y von T gegeben sind, in Zeit $O(\text{dist}_T(x, y))$ der x-y-Weg in T ausgegeben werden. (5 Punkte)

4. Schreiben Sie ein C++-Programm, das zu einem gegebenen ungerichteten Graphen G alle Knoten $v \in V(G)$ bestimmt, die die Ungleichung

$$\frac{1}{|\delta(v)|} \sum_{w \in N(v)} |\delta(w)| \;\geq\; \frac{2|E(G)|}{|V(G)|}$$

   erfüllen.

   Die Graph sei über eine Datei gegeben, die folgendes Format hat:

```
Knotenanzahl
Knoten0a Knoten0b
Knoten1a Knoten1b
...
```

   In der ersten Zeile steht eine einzelne natürliche Zahl, welche die Anzahl der Knoten angibt. Jede weitere Zeile spezifiziert genau eine Kante. Die beiden Einträge einer Zeile sind zwei verschiedene nichtnegative ganze Zahlen, welche die Nummern der Endknoten der Kante sind. Die Reihenfolge der beiden Einträge spielt keine Rolle. Dabei nehmen wir an, dass, wenn wir n Knoten haben, die Knoten von 0 bis $n-1$ durchnummeriert sind. Die Kanten können in beliebiger Reihenfolge in der Datei stehen.

   Dieses Format ist kompatibel zu der Einlese-Funktion, die in der Vorlesung gezeigt wurde. Sie sollten diese Einlese-Funktion und die in der Vorlesung gezeigte Graphenklasse für Ihre Implementierung verwenden. Sie können sie hier herunterladen: `http://www.or.uni-bonn.de/~hougardy/alma/alma2nd.html` (5 Punkte)

Abgabe: Bis Mittwoch, 11.12., 23:59 Uhr, auf der eCampus-Seite der eigenen Gruppe.

Wintersemester 2024/25
Professor Dr. V. Traub
Dr. U. Brenner

Forschungsinstitut für Diskrete Mathematik
Universität Bonn

Algorithmische Mathematik I

Übungszettel 10

1. Sei G ein zusammenhängender ungerichteter Graph, $r \in V(G)$, und T ein durch Tiefensuche ausgehend von r gefundener aufspannender Baum. Für $u, v \in V(G)$ bezeichne P_{uv} den u-v-Weg in T. Zeigen Sie: Für alle Kanten $\{x, y\} \in E(G)$ gilt $x \in V(P_{ry})$ oder $y \in V(P_{rx})$. (5 Punkte)

2. Das Komplement $\bar{G}$ eines ungerichteten Graphen G ist der Graph mit $V(\bar{G}) = V(G)$, in dem zwei Knoten genau dann durch eine Kante verbunden sind, wenn sie es in G nicht sind. Geben Sie einen Algorithmus mit linearer Laufzeit an, der zu einem gegebenen Graphen überprüft, ob sein Komplement bipartit ist. (5 Punkte)

3. Wir wollen Binärstrings (d.h. Wörter über $\{0, 1\}$) lexikographisch sortieren. Wenn s und t zwei Binärstrings der Länge m sind, dann heißt s *lexikographisch kleiner als* t, wenn es einen Index $j \in \{1, \ldots, m\}$ gibt, so dass s und t an den ersten $j-1$ Stellen identisch sind, s an der Stelle j den Wert 0 hat und t an der Stelle j den Wert 1 hat. Zeigen Sie, dass man n Binärstrings der Länge m in Zeit $O(mn)$ (also in linearer Laufzeit) lexikographisch sortieren kann. (5 Punkte)

4. Implementieren Sie eine Funktion, die zu einem gegebenen gerichteten Graphen G entweder eine topologische Ordnung von G oder einen Kreis in G berechnet.

   Ihr Programm sollte einen gerichteten Graphen einlesen, die obige Funktion aufrufen und dann die Knoten in topologischer Ordnung bzw. auf einem Kreis ausgeben.

   Das Eingabeformat entspricht dem für die Programmieraufgabe von Zettel 9. Der einzige Unterschied ist, dass eine Zeile der Form

   ```
   Knotenia Knotenib
   ```

   als Kante, die von `Knotenia` nach `Knotenib` gerichtet ist, interpretiert wird.

   Insbesondere ist das Format wieder kompatibel zu der Einlese-Funktion, die in der Vorlesung gezeigt wurde. Sie sollten wieder diese Einlese-Funktion und die in der Vorlesung gezeigte Graphenklasse für Ihre Implementierung verwenden. Sie können sie wie immer hier herunterladen:
   `http://www.or.uni-bonn.de/~hougardy/alma/alma2nd.html` (5 Punkte)

Abgabe: Bis Mittwoch, 18.12., 23:59 Uhr, auf der eCampus-Seite der eigenen Gruppe.

Wintersemester 2024/25
Professor Dr. V. Traub
Dr. U. Brenner

Forschungsinstitut für Diskrete Mathematik
Universität Bonn

Algorithmische Mathematik I

Übungszettel 11

1. Sei S eine endliche Menge mit einer partiellen Ordnung „$\preceq$“. Beweisen Sie, dass dann folgende Aussagen äquivalent sind:

   (a) $\preceq$ ist durch Schlüssel induziert;

   (b) $(a \not\preceq b \wedge b \not\preceq c \wedge a \neq c) \Rightarrow a \not\preceq c$ für alle $a, b, c \in S$;

   (c) $\{(x, y) \in S \times S \mid x = y \vee (x \not\preceq y \wedge y \not\preceq x)\}$ ist eine Äquivalenzrelation. (6 Punkte)

2. Modifizieren Sie den Algorithmus Mergesort so, dass die gegebene Menge S nicht mehr in zwei, sondern in b (mit $3 \leq b \leq n$) Teilmengen $S_1, \ldots, S_b$ aufteilen. Diese Teilmengen sollen natürlich möglichst ähnliche Größen haben, d.h. je zwei von ihnen sollen sich in ihren Größen höchstens um 1 unterscheiden. Die Mengen $S_1, \ldots, S_b$ werden dann rekursiv sortiert und die sortierten Teilmengen anschließend zu einer sortierten Gesamtmenge verschmolzen.

   (a) Zeigen Sie, dass das Verschmelzen der Teillösungen in Zeit $O(n \log b)$ möglich ist.

   (b) Führen Sie eine asymptotische Laufzeitanalyse des Verfahrens durch. (2+3 Punkte)

3. Zeigen Sie, dass man, wenn n Elemente mit Schlüsseln gegeben sind, in Zeit $O(n)$ einen Binärheap für diese Elemente aufbauen kann. (4 Punkte)

4. Implementieren Sie Mergesort, um einen Vektor von `int`-Variablen zu sortieren. Testen Sie Ihr Programm auf Vektoren von Zufallszahlen. (5 Punkte)

Zusatzaufgabe:

5. Der Weihnachtsmann soll alle n Knoten eines zusammenhängenden ungerichteten Graphen G besuchen, um den n dort wartenden Kindern Geschenke zu bringen. Eine *Tour* ist ein geschlossener Kantenzug, der jeden Knoten mindestens einmal besucht. Zeigen Sie:

   (a) Es gibt immer eine Tour mit höchstens $2n - 2$ Kanten.

   (b) Diese Schranke ist scharf.

   (c) Für gegebenes G kann man in linearer Zeit eine Tour finden, die höchstens doppelt so lang ist wie jede andere. (3+2+3 Bonuspunkte)

Abgabe: Bis Mittwoch, 8.1.2025, 23:59 Uhr, auf der eCampus-Seite der eigenen Gruppe.

Wintersemester 2024/25
Professor Dr. V. Traub
Dr. U. Brenner

Forschungsinstitut für Diskrete Mathematik
Universität Bonn

Algorithmische Mathematik I

Übungszettel 12

1. (a) Es sei (G, c) eine Instanz des Minimum-Spanning-Tree-Problems, bei der $c(e) \neq c(e')$ für je zwei verschiedene Kanten e und e' gilt. Zeigen Sie, dass es dann nur eine optimale Lösung geben kann.

   (b) Beweisen Sie die folgende Aussage: Entfernt man aus einem zusammenhängenden ungerichteten Graphen mit Kantengewichten sukzessive eine schwerste Kante, deren Herausnahme nicht den Zusammenhang des Graphen zerstört, so bleibt am Ende ein MST übrig. (4+4 Punkte)

2. Angenommen, alle Kantengewichte sind ganzzahlig und liegen zwischen 0 und einer Konstante C. Zeigen Sie, dass es dann einen Algorithmus mit linearer Laufzeit für das Kürzeste-Wege-Problem gibt. (3 Punkte)

3. Gegeben seien ein gerichteter Graph G mit Kantengewichten $c : E(G) \to \mathbb{R}_{\geq 0}$ und $s, t \in V(G)$. Gesucht ist ein s-t-Weg, dessen längste Kante möglichst kurz ist. Zeigen Sie, dass dieses Problem in $O(m \log n)$ Zeit lösbar ist mit $m := |E(G)|$ und $n := |V(G)|$. (4 Punkte)
   Hinweis: Modifizieren Sie Dijkstras Algorithmus geeignet.

4. Implementieren Sie Kruskals Algorithmus. Der Eingabegraph sei wieder durch eine Auflistung aller Kanten in dem Format gegeben, das durch den entsprechenden Konstruktor der Klasse `Graph` aus der Vorlesung eingelesen wird. Das Programm soll in der ersten Zeile das Gewicht des berechneten Baums ausgeben. Jede weitere Zeile soll je eine Baumkante durch die Angabe der Indizes ihrer Endknoten spezifizieren. Sie können annehmen, dass der gegebene Graph einfach und zusammenhängend ist.

   Ihre Implementierung soll eine Laufzeit $O(mn)$ erreichen (wobei wieder m die Kantenzahl und n die Knotenzahl sei). Sie erhalten 5 Zusatzpunkte, wenn Sie sogar eine Laufzeit $O(m \log n)$ erreichen. (5 Punkte)

Abgabe: Bis Mittwoch, 15.1.2025, 23:59 Uhr, auf der eCampus-Seite der eigenen Gruppe.

Wintersemester 2024/25
Professor Dr. V. Traub
Dr. U. Brenner

Forschungsinstitut für Diskrete Mathematik
Universität Bonn

Algorithmische Mathematik I

Übungszettel 13

1. Sei $k \in \mathbb{N}$ eine Konstante. Zeigen Sie, dass man zu einem gegebenen ungerichteten Graphen G in polynomieller Laufzeit ein Matching M bestimmen kann, für das es keinen M-augmentierenden Weg in G gibt, der weniger als k Kanten hat. Um welchen Faktor kann ein solches Matching höchstens kleiner sein als ein kardinalitätsmaximales Matching? Zeigen Sie, dass die von Ihnen bewiesene Schranke bestmöglich ist. (5 Punkte)
   Bemerkung: Mit polynomieller Laufzeit ist eine Laufzeit von $O((n+m)^c)$ für eine Konstante c gemeint, wobei n und m wie üblich die Knoten- bzw. Kantenzahl bezeichnen.

2. Sei G ein bipartiter Graph mit Bipartition $V(G) = A \dot\cup B$. Es seien $S \subseteq A$ und $T \subseteq B$ zwei Mengen, so dass es ein Matching M_1 in G gibt, das S überdeckt, und ein Matching M_2, das T überdeckt. Zeigen Sie, dass es dann auch ein Matching in G gibt, das S und T überdeckt. (5 Punkte)

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

4. Implementieren Sie den Bipartiten Matching-Algorithmus, um ein größtes Matching in einem bipartiten Graph G zu bestimmen.

   Der Eingabegraph sei wie üblich durch eine Auflistung aller Kanten in dem Format gegeben, das durch den entsprechenden Konstruktor der Klasse `Graph` aus der Vorlesung eingelesen wird. Sie können annehmen, dass der gegebene Graph einfach und bipartit ist und die Knoten mit den Indizes $0, \ldots, \frac{n}{2}-1$ eine Seite einer Bipartition und die Knoten mit den Indizes $\frac{n}{2}, \ldots, n-1$ die andere Seite der Bipartition bilden. Außerdem dürfen Sie annehmen, dass es keine Knoten vom Grad 0 gibt.

   Das Programm soll in jeder Zeile je eine Matching-Kante durch die Angabe der Indizes ihrer Endknoten spezifizieren. Ihre Implementierung soll eine Laufzeit $O(mn)$ erreichen, wobei wie immer m die Zahl der Kanten und n die Zahl der Knoten sei. (5 Punkte)

Abgabe: Bis Mittwoch, 22.1.2025, 23:59 Uhr, auf der eCampus-Seite der eigenen Gruppe.