RWTH Aachen **SS 2015**
Lehrgebiet Theoretische Informatik Blatt 1
Rossmanith–Reidl–Sanchez 15. März 2015

Übung zur Vorlesung Algorithmen und Datenstrukturen

Aufgabe T1

Ordnen Sie die folgenden Funktionen in der Reihenfolge ihres asymptotischen Wachstums (ohne lange Nachzudenken). Mit $\log n$ bezeichnen wir den Logarithmus zur Basis 2.

- $\log(n)$
- $\log(n^5)/\log\log(n)$
- $n^{\log\log n}$
- $\log(n)^{\log n}$
- n^2
- $n \log n$
- 2^n

Aufgabe T2

Für welche Funktionen $f\colon \mathbf{N} \to \mathbf{N}$ gilt $f(O(1)) = O(1)$? Beweisen Sie Ihre Behauptung mithilfe der Definition der O-Notation.

Aufgabe T3

Entwerfen Sie einen einfachen Algorithmus, der zwei doppelte verkettete Listen aneinanderhängt und auf diese Weise eine neue doppelt verkettete Liste erzeugt. Können Sie dabei auf bedingte Verzweigungen verzichten?

Schreiben Sie den Algorithmus in Pseudocode oder in einer Programmiersprache nieder.

Aufgabe H1

Gegeben sei ein Array $a[0], \ldots, a[n-1]$, welches Zahlen in aufsteigend sortierter Reihenfolge enthält. Entwerfen Sie einen effizienten Algorithmus, der die Anzahl der Zahlen in diesem Array bestimmt, welche sich in einem Interval $[l, r]$ befinden, wobei l und r ebenfalls zwei Zahlen sind.

Die Laufzeit ihres Algorithmus sollte $O(\log n)$ betragen.

Erläutern Sie die Arbeitsweise des Algorithmus und begründen Sie, warum er schnell ist.

Beachten Sie insbesondere den Fall, wenn das Array auch die Zahlen l und r mehrfach enthält. Auch dann muß das Ergebnis korrekt sein.

Implementieren Sie Ihren Algorithmus in einer geeigneten Sprache und lassen Sie ihn auf einigen aussagekräftigen Beispielen laufen. Verwenden Sie dabei auch Fälle mit $n = 10000000$. Überlegen Sie sich ein Verfahren, wie sie die Laufzeit ihrer Suche in diesem Falle möglichst gut messen können und geben Sie die dabei gefundene Zeit an.

Aufgabe H2

Effizienz spielt in dieser Vorlesung eine große Rolle. Will man ein spezielles Problem aber nur ein einziges Mal lösen, dann achtet man nicht so gerne auf Effizienz, sondern versucht es mit so wenig Aufwand wie möglich zu lösen. Insbesondere versucht man vorhandene Programme wiederzuverwenden, um möglichst wenig selbst tun zu müssen.

In dieser Aufgabe versuchen wir also ein solches Problem mit möglichst wenig Aufwand zu lösen. Sie dürfen alle Hilfsmittel, die Ihnen einfallen, verwenden, mit Ausnahme von Hilfe durch andere Personen.

In der Datei `words` auf unserer Webseite finden Sie 462983 Zeilen mit je einem Wort. Einige der Worte in dieser Datei finden sich dort auch rückwärts geschrieben, wobei wir Groß- und Kleinschreibung ignorieren wollen. Ein Beispiel eines solchen Wortes ist `redrawer` und ein weiteres ist `Abba`.

Wie können Sie diese Aufgabe *mit so wenig Aufwand* wie möglich lösen? Was ist die genaue Anzahl dieser besonderen Wörter in der Datei `words`?

RWTH Aachen **SS 2015**
Lehrgebiet Theoretische Informatik Blatt 2
Rossmanith–Reidl–Sanchez 22. April 2015

Übung zur Vorlesung Algorithmen und Datenstrukturen

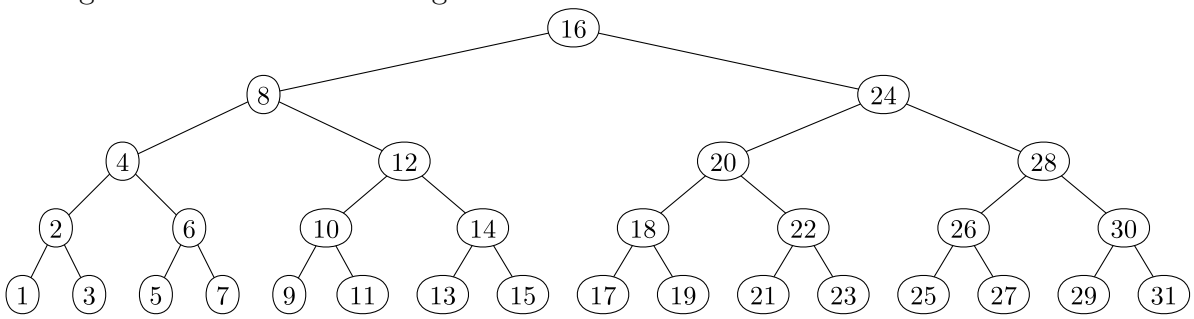
Aufgabe T4

Wie sieht ein anfangs leerer binärer Suchbaum aus, nachdem die Schlüssel 10, 5, 7, 20, 9 und 4 in dieser Reihenfolge eingefügt wurden? Wie sieht der Baum aus, wenn wir dann erst die 5 löschen und sie anschliessend wieder einfügen?

Aufgabe T5

In welcher Reihenfolge könnten die Schlüssel in folgenden binären Suchbaum eingefügt worden sein? Wieviele Vergleich müssen wir im Durchschnitt ausführen, wenn wir einen zufällig gewählten Schlüssel in diesem Baum suchen?

Welcher Suchbaum entstünde, fügten wir dieselben Schlüssel in aufsteigender Reihenfolge in einen anfangs leeren Suchbaum ein? Wieviele Vergleiche würde eine Suche nach einem zufälligen Schlüssel dann benötigen?



Aufgabe T6

Wie sieht ein AVL-Baum aus, in welchen – nachdem er leer war – die Schlüssel 1, 5, 3, 10 und 8 in dieser Reihenfolge eingefügt wurden? Gehen Sie schrittweise vor und überlegen Sie dabei, welche Rotationen ausgeführt werden.

Wie sieht der Baum aus, wenn dann die 1 gelöscht wird?

Aufgabe H3

Wir führen die Aufgabe T6 weiter und beginnen mit dem AVL-Baum, der dort am Ende betrachtet wurde (nachdem die 1 gelöscht wurde).

In diesen Baum werden jetzt die Schlüssel 1, 2, 6 und 7 in dieser Reihenfolge eingefügt. Wie sieht der AVL-Baum nach jedem dieser Schritte aus?

Anschliessend löschen wir die 2. Wie sieht der Baum danach aus?

Aufgabe H4

Welche AVL-Bäume gibt es mit den Schlüsseln 1, 2, 3, 4? Welche binären Suchbäume haben wir mit diesen Schlüsseln?

RWTH Aachen **SS 2015**
Lehrgebiet Theoretische Informatik Blatt 3
Rossmanith–Reidl–Sánchez 29. April 2015

Übung zur Vorlesung Algorithmen und Datenstrukturen

Aufgabe T7

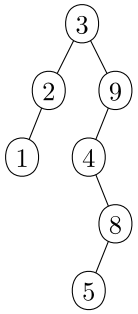
In einem Splay-Baum können wir so nach einem Schlüssel suchen:

```
public boolean containsKey(K k) {
  if(root == null) return false;
  Searchtreenode⟨K, D⟩ n = root, last = root;
  int c;
  while(n ≠ null) {
    last = n;
    c = k.compareTo(n.key);
    if(c < 0) n = n.left;
    else if(c > 0) n = n.right;
    else { splay(n); return true; }
  }
  splay(last); return false;
}
```

Warum gibt es am Ende die Anweisung *splay*(*last*)? Überlegen Sie sich ein Beispiel, in welchem bei einem anfangs leeren Splay-Baum n Operationen zu einem Zeitaufwand von $\Theta(n^2)$ führen, falls diese Anweisung fehlt.

Aufgabe T8

Wir betrachten folgenden Splay-Baum:



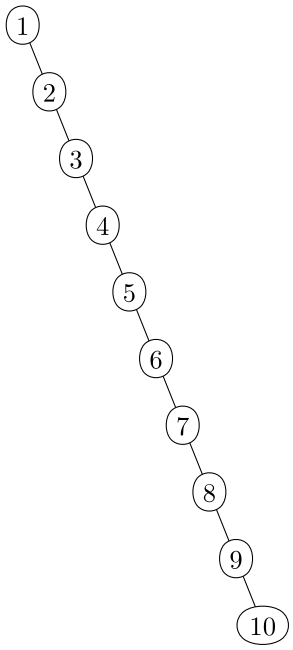
Was passiert, wenn wir in diesen Baum nach dem Schlüssel 10 suchen, dann nach 1 suchen, dann 6 einfügen und schließlich 8 löschen?

Aufgabe T9

Entwerfen Sie einen effizienten Algorithmus, der einen Splay-Baum B und einen Schlüssel k bekommt und den Baum B in zwei Splay-Bäume B_1 und B_2 teilt, wobei B_1 alle Schlüssel enthält, die kleiner oder gleich k sind und B_2 die übrigen Schlüssel.

Aufgabe H5

Wir betrachten folgenden Splay-Baum, der recht unbalanciert ist:



Was passiert, wenn wir erst nach dem Schlüssel 10 suchen und dann nach dem Schlüssel 9? Ist der Baum jetzt besser balanciert?

Aufgabe H6

Entwerfen Sie einen effizienten Algorithmus, der im wesentlichen das Gegenteil vom Algorithmus der Aufgabe T9 vollführt: Als Eingabe erhält er zwei Splay-Bäume B_1 und B_2, wobei garantiert wird, daß kein Schlüssel in B_2 kleiner ist, als ein Schlüssel in B_1. Als Ausgabe soll er einen einzigen Splay-Baum B erzeugen, der nun alle Schlüssel von B_1 und B_2 enthält.

RWTH Aachen **SS 2015**
Lehrgebiet Theoretische Informatik Blatt 4
Rossmanith–Reidl–Sánchez 6. Mai 2015

Übung zur Vorlesung Algorithmen und Datenstrukturen

Aufgabe T10

Wir betrachten die Hashfunktion

$$h\colon \Sigma^* \to \{0,\ldots,m-1\},\ c_1c_2\ldots c_k \mapsto \Big(\sum_{i=1}^{k} c_i\Big) \bmod m,$$

welche die Menge Σ^* aller Wörter (im ASCII-Alphabet) auf eine Zahl zwischen 0 und $m-1$ abbildet.

a) Ist h ein gute Hashfunktion? Überlegen Sie sich realistische Anwendungsbeispiele, für welche h sehr viele Kollisionen erzeugt.

b) Wie könnte eine vernünftige Hashfunktion für Zeichenketten aussehen, für welche Sie unter normalen Umständen ein gutes Verhalten erwarten würden? Wie gehen Sie mit dem Fall um, daß m sehr groß ist?

Aufgabe T11

Wir verwenden jetzt folgende universelle Familie von Hashfunktionen:

$$\{\, h_{a,b} \mid 1 \le a < 5,\ 0 \le b < 5 \,\}$$

mit $h_{a,b}(x) = ((ax+b) \bmod 5) \bmod 4$.

Berechnen Sie die Wahrscheinlichkeit, daß $h(2) = h(3)$ ist, falls wir h zufällig aus obiger Familie von Funktionen wählen.

Was müßte herauskommen, wenn man bedenkt, daß es sich um eine universelle Familie von Hashfunktionen handelt?

Da es sich um viele Funktionen handelt, sollte die Arbeit unter viele Personen verteilt werden, damit es schneller geht. Wie lange brauchen Sie gemeinsam, um diese Wahrscheinlichkeit ohne Taschenrechner oder ähnlichem zu berechnen?

Aufgabe H7

Erstellen Sie mithilfe eines Programms eine Tabelle, welche die Wahrscheinlichkeiten von $h(x) = h(y)$ für alle $0 \le x, y < 5$ enthält, falls h wieder zufällig aus der Familie von T11 gezogen wird.

Ist das Ergebnis das, was Sie erwarten?

Wiederholen Sie das Experiment, aber ersetzen Sie jetzt überall 5 durch 6. Kommentieren Sie das Ergebnis. Nehmen Sie insbesondere dazu Stellung, ob es sich auch jetzt um eine universelle Familie von Hashfunktionen handelt.

Aufgabe H8

Gegeben sei die (wohl nicht besonders gute) Hashfunktion

$$\{0, \ldots, 10^{10}\} \to \{0, \ldots, 9999\},\ n \mapsto (5n^2 + 3n + 7)^{16} \bmod 10000.$$

Finden Sie zehn verschiedene Werte aus dem Definitionsbereich dieser Funktion, welche alle auf denselben Funktionswert abgebildet werden.

RWTH Aachen **SS 2015**
Lehrgebiet Theoretische Informatik Blatt 5
Rossmanith–Reidl–Sánchez 13. Mai 2015

Übung zur Vorlesung Algorithmen und Datenstrukturen

Aufgabe T12

Fügen Sie die Zahlen 23, 12, 5, 17, 28, 10 und 5 in einen anfangs leeren Min-Heap ein (die kleineren Zahlen sind oben). Wie sieht dieser aus?
Entfernen Sie jetzt nacheinander dreimal die kleinste Zahl aus dem Heap. Wie sieht er nach jeder der drei Operationen aus?

Aufgabe T13

Eine *Prioritätswarteschlange* ist eine Datenstruktur, welche folgende Operationen erlaubt:

1. *extract-min*: Gebe das kleinste gespeicherte Element zurück und lösche es aus der Warteschlange.
2. *insert*(x): Füge das Element x ein.

Wie können diese Operationen mithilfe von Heaps umgesetzt werden? Was ist die Laufzeit? Sie können davon ausgehen, daß anfangs bekannt ist, wieviele Elemente sich höchstens gleichzeitig in der Warteschlange befinden können.
Welche Vor- und Nachteile hat ein Heap gegenüber einer Implementierung basierend auf balancierten Suchbäumen?

Aufgabe H9

Hier ist noch einmal eine einfache Variante des Quicksort-Algorithmus:

procedure *quicksort*(L, R) :
  if $R \leq L$ **then return fi**;
  $p := a[L]$; $l := L$; $r := R + 1$;
  do
    do $l := l + 1$ **while** $a[l] < p$;
    do $r := r - 1$ **while** $p < a[r]$;
    vertausche $a[l]$ *und* $a[r]$;
  while $l < r$;
  $temp := a[r]$; $a[L] := a[l]$; $a[l] := temp$; $a[r] := p$;
  quicksort$(L, r - 1)$; *quicksort*$(r + 1, R)$

Das Array a enthalte die Zahlen $4, 1, 3, 2, 5, 9$. Welche rekursiven Aufrufe gibt es? Geben Sie den Inhalt des Arrays jeweils am Anfang jedes Aufrufs und bevor die letzte Zeile ausgeführt wurde an.

Aufgabe H10

Analysieren Sie die Laufzeit von Quicksort für den Spezialfall, daß *alle* Elemente identisch sind. Wie verhält sich Insertionsort und Heapsort in diesem Fall?

RWTH Aachen **SS 2015**
Lehrgebiet Theoretische Informatik Blatt 6
Rossmanith–Reidl–Sánchez 20. Mai 2015

Übung zur Vorlesung Algorithmen und Datenstrukturen

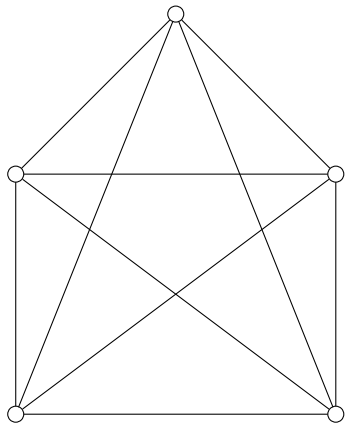
Aufgabe T14

	Quicksort	Heapsort	Mergesort	Insertion-Sort	Straight-Radix	Radix-Exchange
in-place?						
stabil?						
Laufzeit (worst-case)						
Laufzeit (Durchschnitt)						
vergleichsbasiert?						

Beantworten Sie die Fragen für alle Sortierverfahren. Gehen Sie davon aus, daß ein Vergleich in konstanter Zeit durchgeführt wird und die Anzahl der zu sortierenden Elemente n beträgt. Für Laufzeiten tragen Sie eine Funktion $f(n)$ in die Tabelle ein, um eine Laufzeit von $O(f(n))$ auszudrücken.

Aufgabe T15

Ein *Dreieck* in einem Graphen ist ein Untergraph, der aus drei Knoten besteht, welche paarweise miteinander verbunden sind.
Entwerfen Sie einen Algorithmus, der für einen Graphen mit n Knoten in $O(n^3)$ Schritten herausfinden kann, ob er ein Dreieck enthält.



Wieviele Dreiecke gibt es im oben gezeigten Graphen?

Aufgabe H11

Gegeben sei ein Array der Länge n, welches garantiert nur k verschiedene Zahlen enthält – jede aber beliebig oft. Wir gehen davon aus, daß n viel größer als k ist und k sogar viel kleiner als $\log n$ sein kann.
Erfinden und beschreiben Sie ein Sortierverfahren, welches in dieser speziellen Situation sehr schnell ist.
Genauer gesagt: Die Laufzeit soll nur $O(n \log k)$ betragen.

Aufgabe H12

Es gibt ein einfaches Verfahren, zwei quadratische $n \times n$-Matrizen miteinander zu multiplizieren. Die Laufzeit des einfachen Verfahrens ist $O(n^3)$.

Allerdings gibt es kompliziertere Verfahren, die asymptotisch deutlich schneller eine Matrixmultiplikation ausführen können.

In dieser Aufgabe wollen wir dies ausnutzen, um schneller nach Dreiecken in einem Graphen suchen zu können.

a) Stellen Sie sich vor, Sie multiplizieren eine Adjazenzmatrix eines Graphen G mit sich selbst und verwenden die resultierende Matrix wieder als Adjazenzmatrix eines neuen Graphens, den wir G' nennen. Welche interessante Beziehung besteht zwischen G und G'?

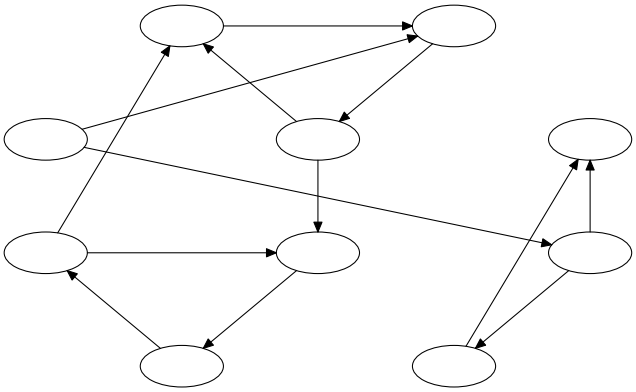
b) Entwerfen Sie jetzt einen auf schneller Matrixmultiplikation basierenden Algorithmus, der feststellen kann, ob ein Graph ein Dreieck enthält. Die Laufzeit soll dabei von der Laufzeit zur Matrixmultiplikation dominiert werden und alles andere soll nur $O(n^2)$ Zeit verbrauchen.

RWTH Aachen **SS 2015**
Lehrgebiet Theoretische Informatik Blatt 7
Rossmanith–Reidl–Sánchez 3. Juni 2015

Übung zur Vorlesung Algorithmen und Datenstrukturen

Aufgabe T16

Führen Sie auf folgenden Graphen eine Tiefensuche aus. Notieren Sie die *discovery*- und *finish*-Zeiten.
Benennen Sie die Baum-, Quer-, Vorwärts- und Rückwärtskanten.

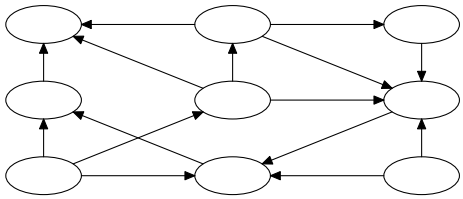


Aufgabe T17

Beweisen oder widerlegen Sie: Ergibt eine Tiefensuche in einem gerichteten Graphen genau eine Rückwärtskante, so liefert jede Tiefensuche in diesem Graphen genau eine Rückwärtskante.

Aufgabe H13

Behandeln Sie folgenden Graphen analog zu Aufgabe T16.



Aufgabe H14

Beweisen oder widerlegen Sie: Ergibt eine Tiefensuche in einem ungerichteten Graphen genau eine Rückwärtskante, so liefert jede Tiefensuche in diesem Graphen genau eine Rückwärtskante.

RWTH Aachen **SS 2015**
Lehrgebiet Theoretische Informatik Blatt 8
Rossmanith–Reidl–Sánchez 10. Juni 2015

Übung zur Vorlesung Algorithmen und Datenstrukturen

Aufgabe T18

Wo steckt der Gedankenfehler in folgendem „Beweis“?

> Wenn ich n Dinge sortieren will, dann sortiere ich sie einfach topologisch, anstatt normal. Danach sind sie sortiert und ich habe es in linearer Zeit geschafft. Quicksort braucht aber ja schon mehr als lineare Zeit und daher habe ich Quicksort geschlagen.

Aufgabe T19

Folgende Relation sei auf den natürlichen Zahlen definiert:
$n \prec' m$ gilt genau dann, wenn n ein Teiler von m ist oder wenn $n > m$ und sowohl n als auch m Primzahlen sind.
Die Halbordnung $\prec$ ist die transitive Hülle von $\prec'$.

a) Skizzieren Sie den gerichteten Graphen, der die Relation $\prec'$ (und damit indirekt auch $\prec$) auf der Teilmenge $\{1, 2, 3, \ldots, 10\}$ darstellt.

b) Bestimmmen Sie eine topologische Sortierung dieser Menge, indem Sie den Algorithmus aus der Vorlesung verwenden.

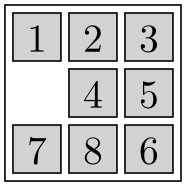
c) Kann man die ganze Menge der natürlichen Zahlen bezüglich $\prec$ topologisch sortieren? Wenn nicht, warum nicht?

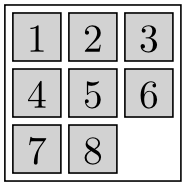
Aufgabe H15

Wir betrachten folgenden ungerichteten Graphen: Die Knoten sind die Zahlen $\{1, 2, 3, 4\}$ und es gibt Kanten zwischen allen Knotenpaaren. Darüber hinaus hat eine Kante zwischen den Knoten i und j das Gewicht $(i + j)^2$.
Führen Sie den Algorithmus von Dijkstra auf diesem gewichteten Graphen aus, wobei Sie 1 als Startknoten verwenden. Geben Sie alle Zwischenschritte an.

Aufgabe H16

Das „Neunerschiebepuzzle“ besteht aus acht beweglichen Feldern in einer 3×3-Matrix. Jeweils eine der neun Positionen ist frei und ein an die freie Position angrenzendes Feld kann in diese hineingeschoben werden. Dies nennen wir einen „Zug“.





Ziel des Spiels ist es, die rechts gezeigte Position zu erreichen. Wir können uns nun einen ungerichteten Graphen vorstellen, dessen Knoten die Positionen dieses Spiels sind. Zwei Positionen sind durch eine Kante verbunden, wenn ein Zug sie ineinander überführt.

Wir können eine Breitensuche auf diesen Graphen beginnen, ohne ihn vorher komplett zu konstruieren. Führen Sie eine solche Breitensuche auf dem links gezeigten Startknoten aus und brechen Sie sie ab, sobald die Zielposition erscheint. Zeichnen Sie den bis dahin entstandenen Breitensuchbaum auf.

Können Sie jetzt eine kürzestmögliche Lösung des Rätsels aus diesem Baum ablesen?

Ist es in dieser und ähnlichen Situationen Ihrer Meinung nach bessers eine Tiefen- oder eine Breitensuche durchzuführen?

RWTH Aachen **SS 2015**
Lehrgebiet Theoretische Informatik Blatt 9
Rossmanith–Reidl–Sánchez 17. Juni 2015

Übung zur Vorlesung Algorithmen und Datenstrukturen

Aufgabe T20

Gegeben sei ein gerichteter Graph G und zwei seiner Knoten s und t.
Es soll festgestellt werden, ob es zwischen s und t mindestens zwei kantendisjunkte Pfade gibt. Überlegen Sie, was das genau bedeutet.
Nun entwirft ein genialer Erfinder folgenden Algorithmus, um dieses Problem zu lösen:

> *Nun, zuerst suchen wir nach irgendeinem Pfad zwischen s und t. Wenn es keinen gibt, dann gibt es natürlich auch keine zwei kantendisjunkte Pfade. Das kann ich mit Tiefensuche in linearer Zeit bewerkstelligen.*
>
> *Falls ich einen solchen Pfad finde, dann lösche ich einfach alle Kanten aus G, die auf dem Pfad liegen. Dadurch kann ich keinesfalls eine Kante aus Versehen zweimal verwenden.*
>
> *Jetzt suche ich einfach wieder nach einem Pfad von s nach t. Finde ich einen, dann gibt es zwei kantendisjunkte Pfade zwischen diesen Knoten. Wenn nicht, dann eben nicht. Die Gesamtlaufzeit ist jedenfalls linear.*

Was sagen Sie dazu?

Aufgabe T21

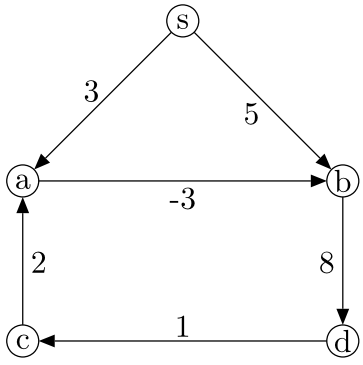
Der Algorithmus von Bellman und Ford kann die Abstände eines Knotens s zu allen anderen Knoten finden, falls es keinen Kreis mit negativem Gewicht gibt.
Entwerfen Sie einen Algorithmus, der nicht nur feststellen kann, ob es einen Kreis mit negativem Gewicht in einem gerichteten Graphen mit Kantengewichten gibt, sondern einen solchen auch finden und ausgeben kann.

Aufgabe H17

Entwerfen Sie einen Algorithmus, der in einem gerichteten Graphen einen Kreis minimaler Länge finden kann (also mit einer minimalen Anzahl von Kanten). Was ist die Laufzeit Ihres Verfahrens?

Aufgabe H18

Wenden Sie den Algorithmus von Bellman und Ford auf folgendes Netzwerk an. Verwenden Sie s als Startknoten.



RWTH Aachen **SS 2015**
Lehrgebiet Theoretische Informatik Blatt 10
Rossmanith–Reidl–Sánchez 24. Juni 2015

Übung zur Vorlesung Algorithmen und Datenstrukturen

Aufgabe T22

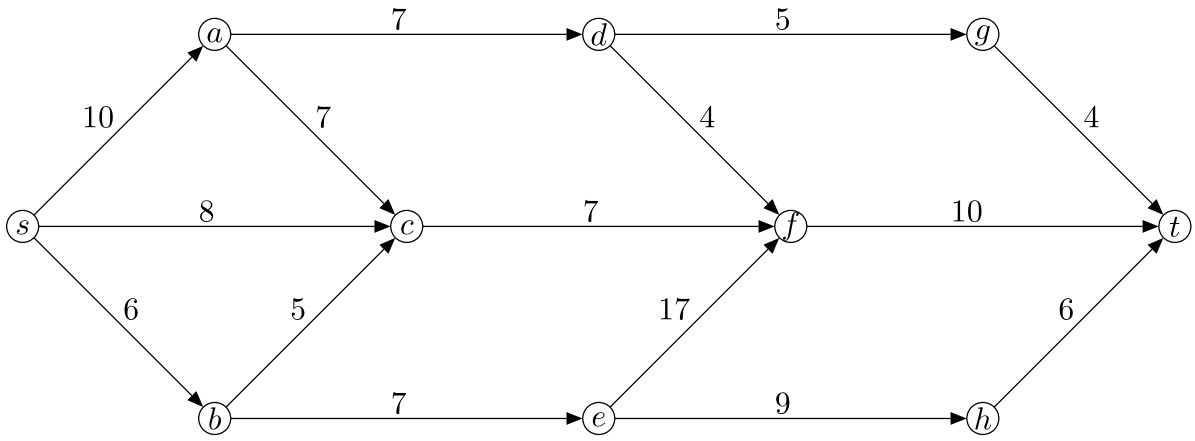
Wieder einmal wollen wir in einem ungerichteten Graphen feststellen, ob es mindestens zwei kantendisjunkte Pfade von einem Knoten s zu einem Knoten t gibt.

Wie läßt sich dieses Problem als Flußproblem modellieren und lösen?

Wenden Sie Ihre Erkenntnisse auf das Gegenbeispiel, das Sie letzte Wochen zum naiven Vorgehen fanden.

Aufgabe T23

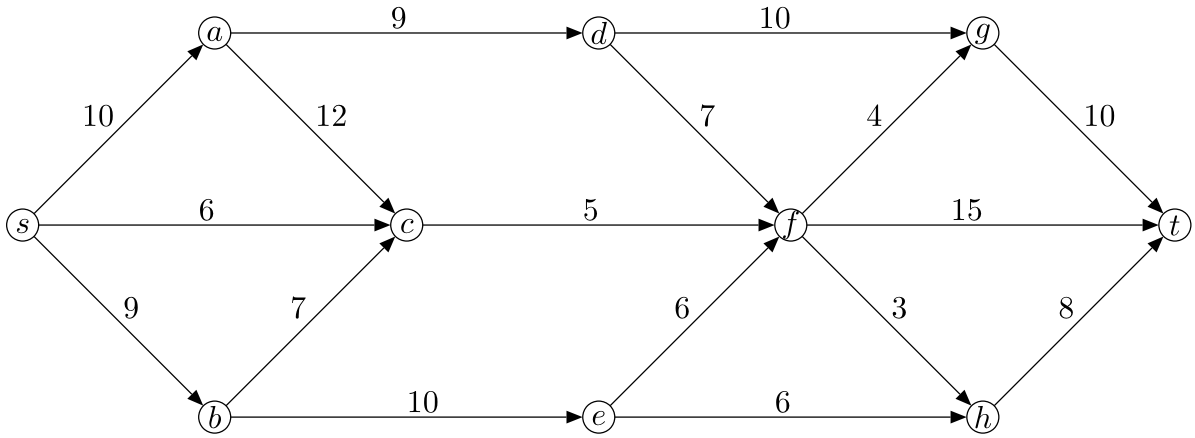
Wenden Sie die Ford–Fulkerson-Methode auf das folgende Flußnetzwerk an. Zeichnen Sie nach jeder Augmentierung das resultierende Residualnetzwerk.



Aufgabe H19

Gehen Sie analog zu Aufgabe T23 mit folgendem Flußnetzwerk um.

Was ist der maximale Fluß, den Sie erhalten?



RWTH Aachen **SS 2015**
Lehrgebiet Theoretische Informatik Blatt 11
Rossmanith–Reidl–Sánchez 30. Juni 2015

Übung zur Vorlesung Algorithmen und Datenstrukturen

Aufgabe T24

Die internationale Raumstation ISS steht auch Weltraumtouristen offen. Sie sollen nun entscheiden, welche Touristen Sie mitnehmen wollen, um möglichst viel Geld zu verdienen.

Gegeben sind Kandidaten $K_1, \ldots, K_n$, welche jeweils bereit sind, $k_1, \ldots, k_n$ US-Dollar zu zahlen. Allerdings sind sie anspruchsvoll und erwarten auf der ISS auch ein Unterhaltungsprogramm (der Erstbesucher Cameron wollte zum Beispiel einen Weltraumspaziergang machen). Zu diesem Zweck stehen eine Menge „Spielzeuge“ $Z_1, \ldots, Z_m$ zur Verfügung. Bei der Mitnahme eines Spielzeugs zur ISS entstehen allerdings jeweils Kosten $z_1, \ldots, z_m$. Der Kandidat K_i ist nur bereit zu zahlen, wenn die Spielzeuge $R_i \subseteq \{Z_1, \ldots, Z_m\}$ mitgenommen werden.

Entwerfen Sie einen effizienten Algorithmus, der eine Menge von Kandidaten auswählt, um die Einnahmen (also die gezahlten Gebühren der Touristen minus die Kosten für die Spielzeuge) zu maximieren. Jedes Spielzeug muß nur einmal mitgenommen werden, selbst wenn mehrere es benutzen wollen.

Aufgabe T25

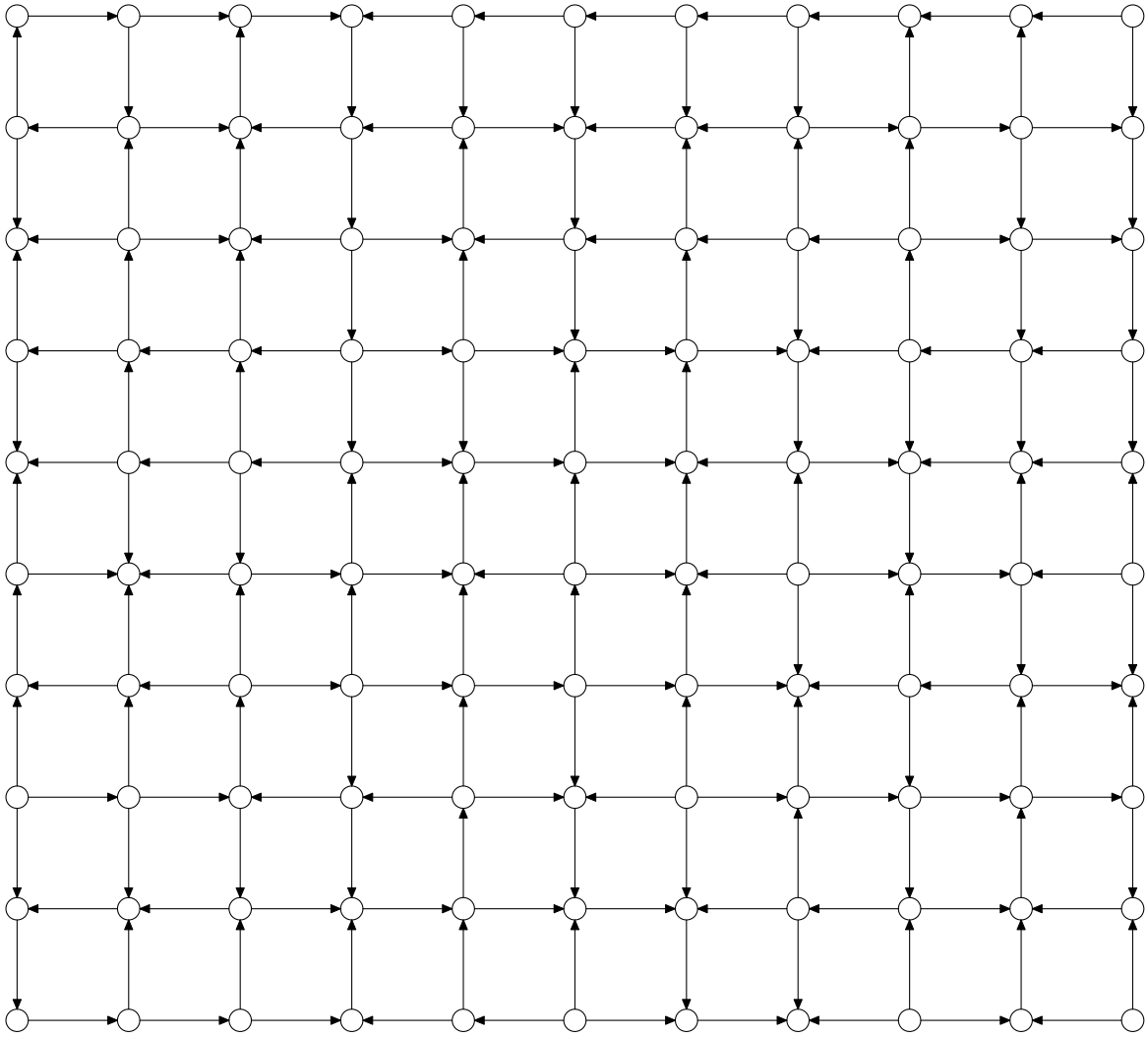
In einem bipartiten Graphen gibt es stets ein Matching maximaler Kardinalität. Es gibt dann kein anderes Matching mit mehr Kanten.

Ein *maximales Matching* nennen wir ein Matching, daß nicht vergrößert werden kann, indem eine weitere Kante hinzugefügt wird.

Entwerfen Sie einen effizienten Algorithmus, um maximale Matchings zu finden.

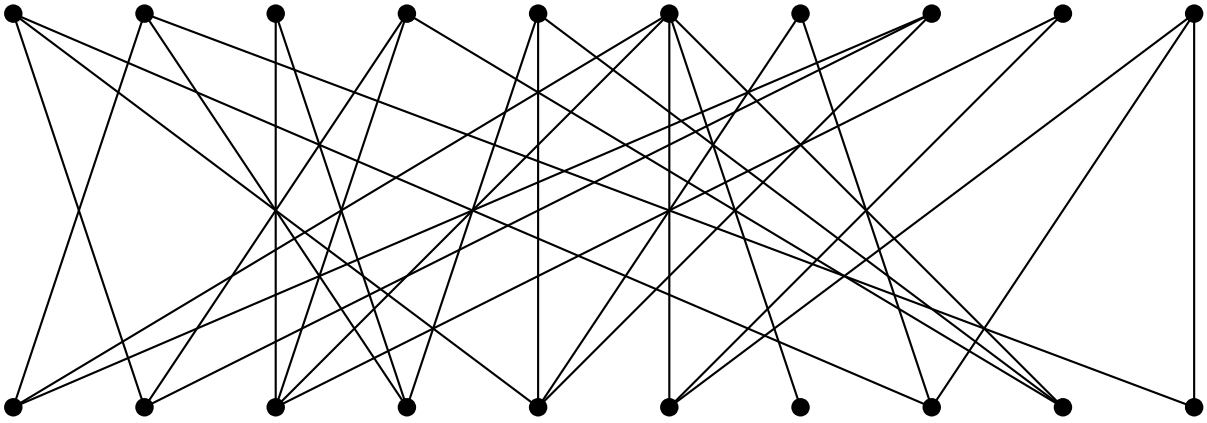
Aufgabe H20

Ermitteln Sie einen maximalen Fluß für das Netzwerk auf der folgenden Seite, wobei die Quellen links und die Senken rechts sind. Beweisen Sie die Maximalität durch Angabe eines gleichgroßen Schnitts. Alle Kapazitäten sind 1.



Aufgabe H21

Finden Sie ein Matching maximaler Kardinalität mit dem Verfahren der Vorlesung für folgenden Graphen:



RWTH Aachen **SS 2015**
Lehrgebiet Theoretische Informatik Blatt 12
Rossmanith–Reidl–Sánchez 8. Juli 2015

Übung zur Vorlesung Algorithmen und Datenstrukturen

Aufgabe T26

Manche Paare von Städten bilden Partnerschaften mit anderen Städten (Aachen zum Beispiel mit Liège, Toledo und weiteren). Diese Beziehungen lassen sich gut durch einen Graphen modellieren.

Nun soll im *Internationalen Jahr der Städtefreundschaften* je eine Feier zwischen allen befreundeten Städtepaaren durchgeführt werden. Wir gehen der Einfachheit halber davon aus, daß alle Feiern dasselbe kosten.

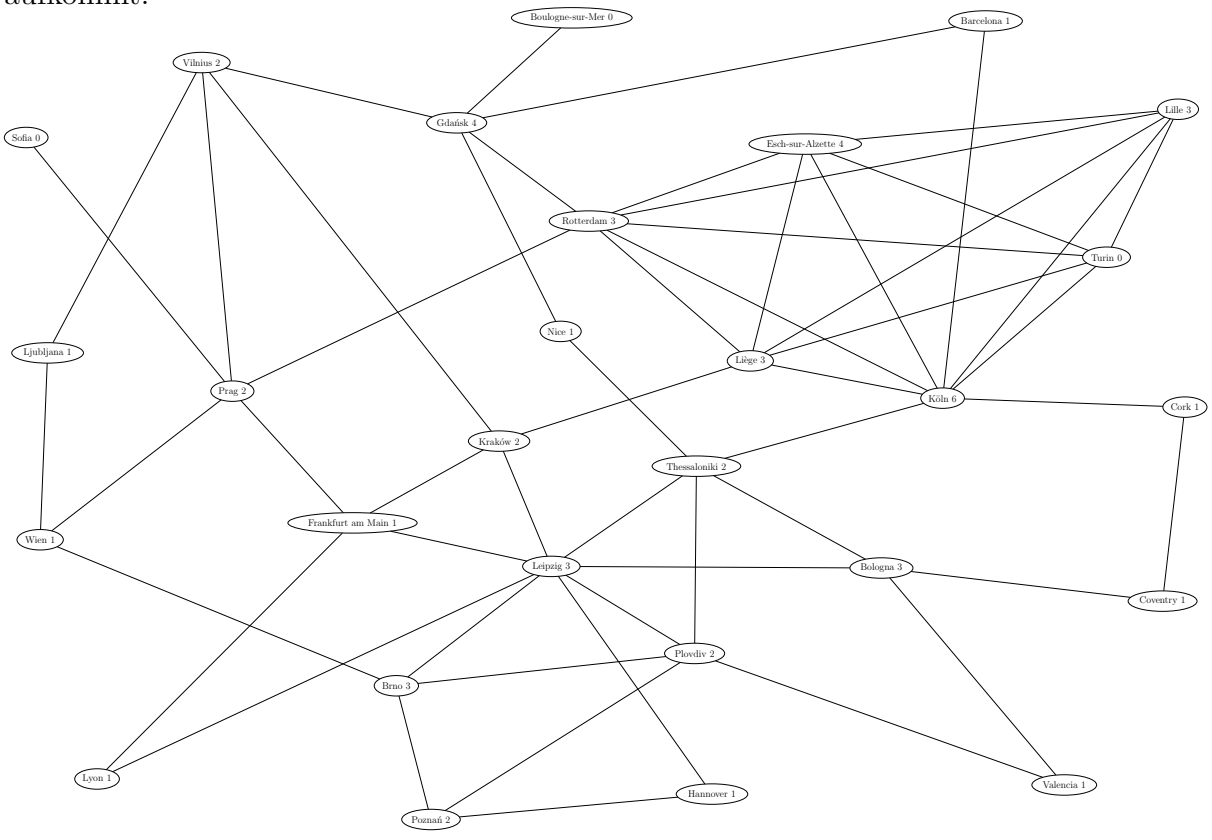
Nun hat jede Stadt ein Budget, welches besagt für wieviele eigene Feiern sie Geld spenden kann. Dies soll stets eine ganze Zahl sein.

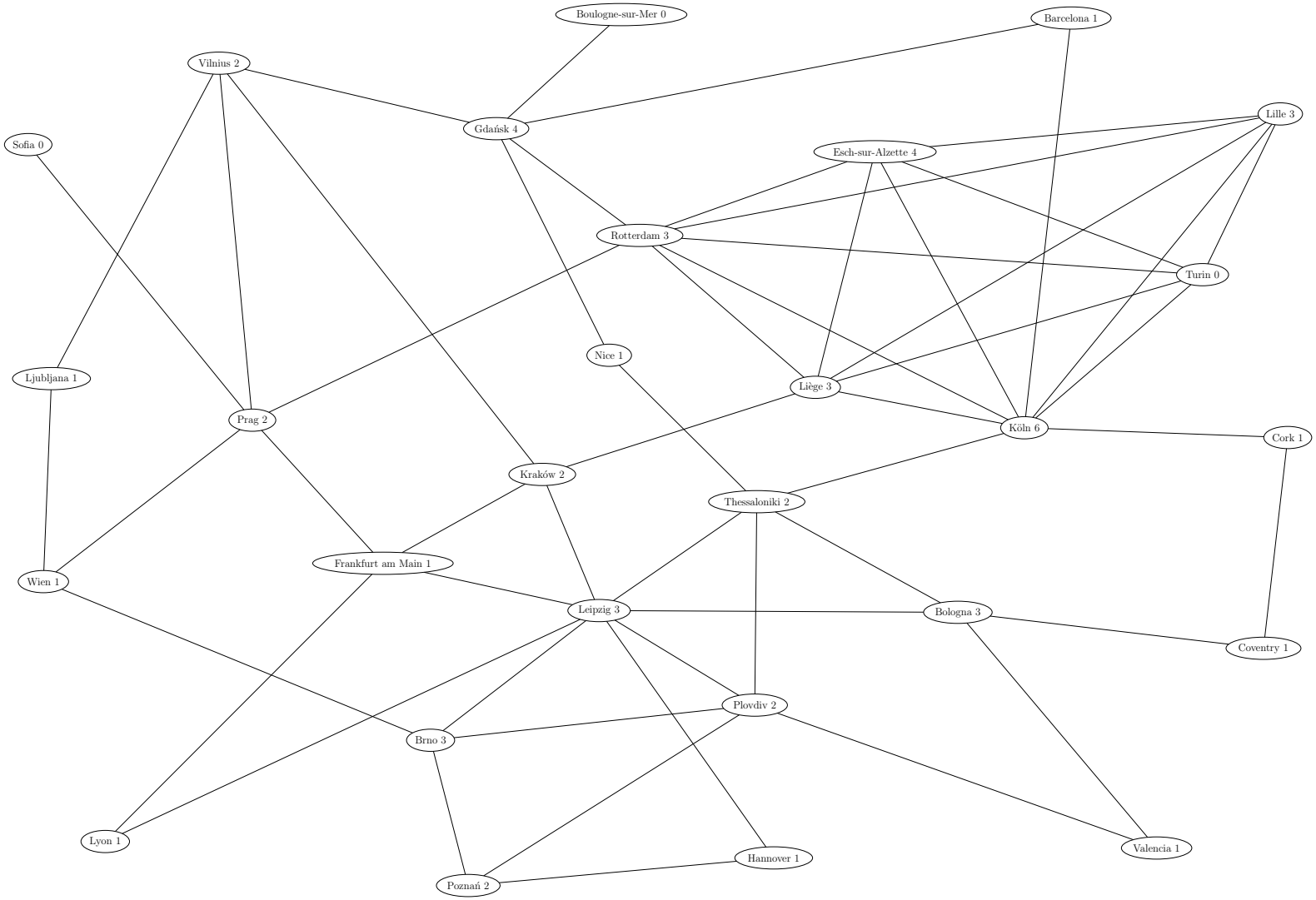
Ihr Problem ist es nun, einen Plan zu finden, der besagt welche Stadt für welche Feiern zahlen soll, ohne daß Budgets überschritten werden.

Können Sie dies durch Lösen eines Flußproblems lösen?

Aufgabe H22

Finden Sie einen Plan, analog zur Aufgabe oben, für das kleinere Städtepartnerschaftsmodell unten. Die Zahlen geben das Budget für jede Stadt wieder. Zeichnen Sie die gefundene Lösung ein, indem Sie für jede Kante markieren, welche der beiden Städte für diese Feier aufkommt.




Boulogne-sur-Mer 0
Barcelona 1
Vilnius 2
Lille 3
Gdańsk 4
Sofia 0
Esch-sur-Alzette 4
Rotterdam 3
Turin 0
Nice 1
Ljubljana 1
Liège 3
Prag 2
Köln 6
Cork 1
Kraków 2
Thessaloniki 2
Frankfurt am Main 1
Wien 1
Leipzig 3
Bologna 3
Coventry 1
Plovdiv 2
Brno 3
Lyon 1
Valencia 1
Hannover 1
Poznań 2