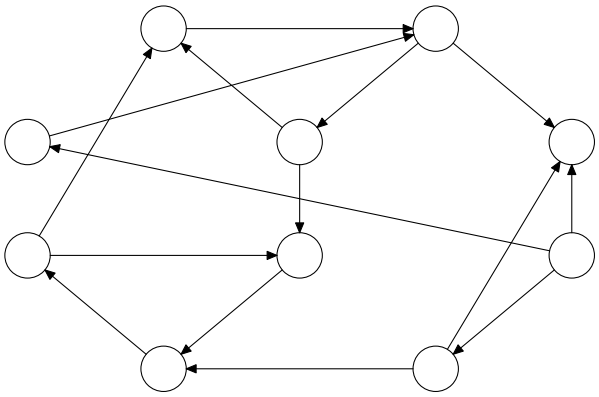
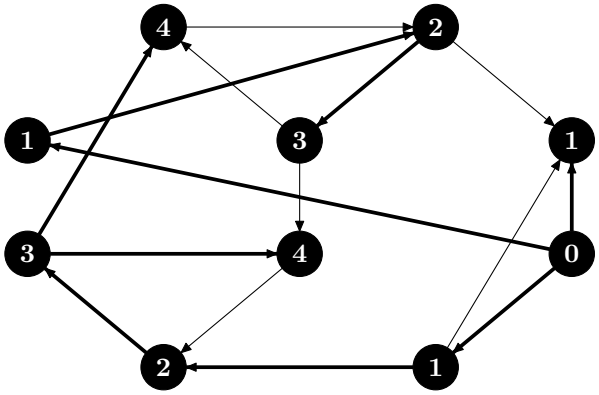
Übungen zur Vorlesung Datenstrukturen und Algorithmen

T26

Wenden Sie Breitensuche auf den folgenden Graphen an.

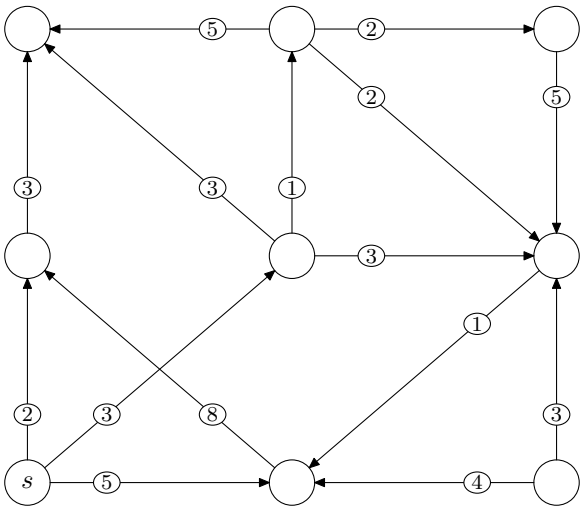


Lösungsvorschlag:

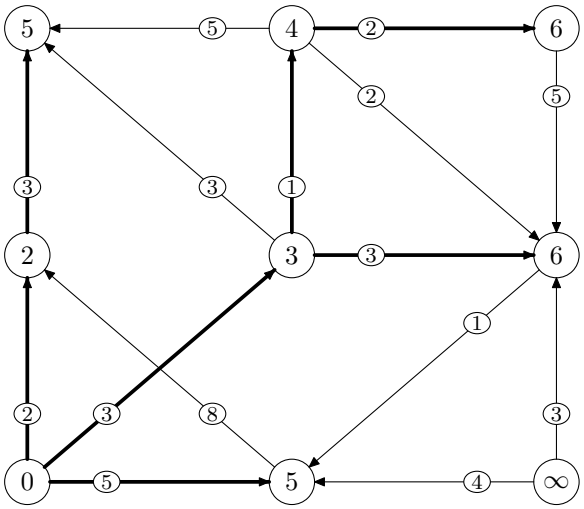


T27

Wenden Sie den Dijkstra-Algorithmus auf den folgenden Graphen an, wobei s die Quelle ist.



Lösungsvorschlag:



T28

Beweisen Sie den zweiten Punkt von Lemma A: $f(X,Y) = -f(Y,X)$ für $X, Y \subseteq V$ falls f ein Fluß für $G = (V,E)$ ist.

Lösungsvorschlag:

$$f(X,Y) \overset{Def.}{=} \sum_{x\in X}\sum_{y\in Y} f(x,y) \overset{Symm.}{=} \sum_{x\in X}\sum_{y\in Y} -f(y,x) =$$

$$\sum_{y\in Y}\sum_{x\in X} -f(y,x) = -\sum_{y\in Y}\sum_{x\in X} f(y,x) \overset{Def.}{=} -f(Y,X)$$

Aufgepasst: Welche Rolle spielen hier das Kommutativitäts-, das Assoziativ- und das Distributivgesetz?

T29

Es sei f ein Fluß in einem s-t-Netzwerk $G = (V,E)$ und $X, Y \subseteq V$, $X \cap Y = \emptyset$, mit $s \in X$ und $t \in Y$. Desweiteren gelte $|f| \neq 0$.
Beweisen oder widerlegen Sie die Aussage $f(X,Y) \geq 0$.

Lösungsvorschlag:
Wähle $V = \{s, t, v\}$ und $f(s, v) = 2$, $f(v, t) = 2$ und $f(t, s) = 1$. Dann ist $f(X, Y) = -1$ für $X = \{s\}$ und $Y = \{t\}$, aber $|f| = 1$.

H22 (10 Punkte)
Beweisen Sie den dritten Punkt von Lemma A: $f(X \cup Y, Z) = f(X, Z) + f(Y, Z)$ für $X, Y, Z \subseteq V$, $X \cap Y = \emptyset$, falls f ein Fluß für $G = (V, E)$ ist.

Lösungsvorschlag:

$$f(X \dot\cup Y, Z) \stackrel{Def.}{=} \sum_{v \in X \dot\cup Y} \sum_{z \in Z} f(v, z) =$$

$$\sum_{x \in X} \sum_{z \in Z} f(x, z) + \sum_{y \in Y} \sum_{z \in Z} f(y, z) \stackrel{Def.}{=} f(X, Z) + f(Y, Z)$$

H23 (10 Punkte)
Diese Aufgabe entstammt dem empfehlenswerten Buch „Introduction to Algorithms“ von Cormen, Leiserson, Rivest und Stein.
Es gibt zwei Arten von professionellen Wrestlern: die „good guys“ und die „bad boys.“ Natürlich gibt es auch Rivalitäten zwischen einzelnen Wrestlern. Nehmen wir an, wir haben n Wrestler und eine Liste von r Rivalitäten (eine Rivalität ist formal gesehen ein Paar von Wrestlern).
Geben Sie einen Algorithmus mit Zeitkomplexität $O(n + r)$ an, der entscheidet, ob die Wrestler so auf die beiden Gruppen verteilt werden können, daß nur Rivalitäten zwischen „good guys“ und „bad boys“ bestehen. Im positiven Fall sollte eine entsprechende Zuteilung auch ausgegeben werden.

Lösungsvorschlag:
Wenn wir die Wrestler als Knoten und Rivalitäten zwischen ihnen als Kanten repräsentieren, dann besteht die Aufgabe des Algorithmus darin, folgende Frage zu beantworten:
Können die Knoten des Rivalitätsgraphen $G = (V, E)$ so auf zwei Klassen V_1, V_2 aufgeteilt werden, daß keine Kante innerhalb von V_1 und keine Kante innerhalb von V_2 auftritt? (Graphen mit dieser Eigenschaft werden aus naheliegenden Gründen bipartit genannt.)
Mit Breitensuche und bei einzelner Betrachtung der Zusammenhangskomponenten von G ist diese Aufgabe schnell erledigt. Wir beginnen pro Komponente mit einem beliebigen Knoten, den wir ohne Beschränkung der Allgemeinheit als „bad boy“ markieren. Natürlich müssen dann alle Nachbarn des Startknotens zu „good guys“ erklärt werden, wenn es eine Lösung geben soll. Entsteht dadurch eine Kante zwischen zwei „good guys,“ müssen wir ohnehin aufgeben. In einem nächsten Schritt müssen die Nachbarn dieser „good guys“ wieder als „bad boys“ gekennzeichnet werden usw. (wir bewegen uns also vom Startknoten aus schichtenweise durch Knoten wachsender Distanz, wie es ja gerade typisch für die Breitensuche ist).
Lassen sich auf diese Weise alle Knoten markieren, ohne daß eine Kante zwischen Knoten mit gleicher Kennzeichnung entsteht, so liefert dies die gewünschte Bipartition. Ansonsten

haben wir einen Kreis ungerader Länge entdeckt, der als Beweis für die Undurchführbarkeit der Aufgabe offensichtlich ausreicht.

H24 (10 Punkte)
Eine Zahl darf durch folgende Operationen verändert werden:

1. Multiplikation mit drei.
2. Multiplikation mit zwei und anschließendes Abziehen von 323.
3. Addition von 27.
4. Subtraktion von 13.

Finden Sie eine kürzest mögliche Folge von Operationen, die die Zahl 1 in die Zahl 10000 verwandelt. Die Zahl darf dabei zwischendurch nicht negativ werden.

Hinweis: Wir können dieses Problem als ein Suchproblem auf gerichteten Graphen interpretieren. Der Graph hat die nicht-negativen ganzen Zahlen als Knoten und es gibt eine Kante von n nach m, wenn wir n durch eine der vier Operationen in m verwandeln können.
Dieser Graph ist unendlich groß. Es ist allerdings sehr einfach, die Nachbarn eines Knotens aufzuzählen. Es bietet sich daher an, auf diesem implizit gegebenen Graphen eine Breitensuche durchzuführen, die bei 1 beginnt.
Selbstverständlich können Sie die Datenstrukturen der Vorlesung in Ihrem Programm verwenden, die auf der Webseite zu finden sind, oder andere Programmbibliotheken. Ebenso können Sie statt Java auch eine andere Programmiersprache verwenden, falls das für Sie einfacher sein sollte.
Geben Sie als Lösung sowohl das Programm als auch seine Ausgabe ab. Die Ausgabe des Programms sollte den Weg von 1 zu 10000 in einer nachvollziehbaren Weise enthalten.

Lösungsvorschlag:
Wir verwenden Breitensuche:

```
public class H24 {
  static Map<Integer,Integer> dist;
  static Map<Integer,Integer> parent;
  static Queue<Integer> active;

  static void insert(Integer n, Integer m) {
    if(m ≥ 0 ∧!dist.iselement(m)) {
      dist.insert(m, dist.find(n)+1);
      active.enqueue(m);
      parent.insert(m, n);
    }
  }

  public static void main(String args[]) {
    dist = new Hashtable<Integer,Integer>();
```

```
      parent = new Hashtable<Integer,Integer>();
      active = new Queue<Integer>();
      dist.insert(1,0);
      active.enqueue(1);
      while(!active.isempty()) {
        Integer n = active.dequeue();
        insert(n, n*3);
        insert(n, n*2-323);
        insert(n, n+27);
        insert(n, n-13);
        if(n ≡ 10000) {
          while(n ≠ 1) {
            System.out.println(n);
            n = parent.find(n);
          }
          return;
        }
      }
    }
}
```

Die Ausgabe des Programms ist:

```
10000
10013
10026
9999
3333
1111
717
239
252
84
28
```