Algorithmen und Datenstrukturen 1

SoSe 2023 · Prof. Dr. Dell
2. Oktober 2023
Klausurnummer: **0000**

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Nachname (Druckschrift)**:** _____

Vorname (Druckschrift)**:** _____

Matrikelnummer: _____

Studiengang: _____

Musterlösung

Bitte Hinweise beachten:

- Schreiben Sie Ihren Namen nur auf dieses Titelblatt.
- Zusatzpapier darf nicht mit abgegeben werden.
- Sie dürfen Stifte in den Farben blau und schwarz verwenden.
- Zugelassenes Hilfsmittel: 1 Blatt DIN A4 mit handschriftlichen Notizen (beidseitig).
- Nicht zugelassene Hilfsmittel (z.B. Handys, Smartwatches, andere Geräte) stellen eine Täuschung dar und führen zum Nichtbestehen der Klausur. Schalten Sie daher alle elektronischen Geräte aus und verstauen Sie diese in Ihrer Tasche.
- Werden zu einer Aufgabe zwei oder mehr Lösungen angegeben, so gilt die Aufgabe als nicht gelöst. Entscheiden Sie sich also immer für **eine** Lösung.
- Begründungen sind nur dann notwendig, wenn die Aufgabenformulierung dies verlangt.
- Die Klausur gilt mit 50% der Höchstpunktzahl als bestanden.
- Die Klausur dauert 180 Minuten.

Please note:

- *Write your name on this cover sheet only.*
- *Additional paper must not be submitted.*
- *You may use pens in the colors blue and black.*
- *Approved aid: 1 sheet DIN A4 with handwritten notes (on both sides).*
- *Non-approved aids (e.g. mobile phones, smartwatches, other devices) constitute deception and lead to failing the exam. Turn off your electronic devices before the exam and store them in your bag.*
- *If two or more solutions are given for a problem, the problem counts as unsolved. So always choose **one** solution.*
- *Justifications are only necessary if the problem wording requires it.*
- *The exam is passed with 50% of the maximum number of points.*
- *The exam lasts 180 minutes.*

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Musterlösung

Diese Seite ist für den internen Gebrauch.
Bitte leer lassen.

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Diese Seite ist für den internen Gebrauch.
Bitte leer lassen.

Musterlösung

Aufgabe	1	2	3	4	5
Erreichbar	20	8	12	20	40
Erreicht					

Summe	Note
/100	

Aufgabe 1: Landau-Notation und Laufzeit — $5+2+2+2+9=20$ Punkte

a) Welche der folgenden Aussagen sind korrekt? Es könnten zwischen 0 und 8 Antworten korrekt sein. Kreuzen Sie genau die korrekten Antworten an:
Which of the following statements are correct? There could be between 0 and 8 correct answers. Mark exactly the correct answers:

☐ $\log(2^z) = \Theta(z^2)$ ☑ $\log x = o((\log x)^2)$ ☑ $\log k = O(2^k)$ ☑ $2^m = \omega(m^2)$

☐ $m = \Omega(m^2)$ ☐ $(y/2) \in o(99 \cdot y)$ ☑ $2^{2t} = \Theta(4^t)$ ☑ $(\log n)^2 = \omega(\log(n^2))$

(Jedes falsche Kreuz oder Nicht-Kreuz $= -1$ Punkt.)
(Each incorrect mark or non-mark $= -1$ point.)

⇑⇑⇑ _____ / 5 Punkte ⇑⇑⇑

b) Welches Paar von Funktionen erfüllt $f(N) = O(g(N))$? Kreuzen Sie genau eine Antwort an. *Which pair of functions satisfies $f(N) = O(g(N))$? Mark exactly one answer.*

☑ $f(N) = (N + 2N) \cdot (\log(N) + \log(N))$; $g(N) = N \cdot \log(N)$

☐ $f(N) = N^9 - N^2$; $g(N) = N^9/N^2$

☐ $f(N) = \log(N)$; $g(N) = (\log(\log(N)))^{99}$

☐ $f(N) = N^2$; $g(N) = \sqrt{N(N+1)}$

(Nicht genau die richtige Antwort $= 0$ Punkte. *Not exactly the right answer $= 0$ points.*)

⇑⇑⇑ _____ / 2 Punkte ⇑⇑⇑

c) Geben Sie eine Funktion $f(n)$ an,sodass $f(n) = \omega(\frac{2}{5} \cdot n^{42})$ und $f(n) = o(\frac{7}{8} \cdot n^{44})$ gilt:
Enter a function $f(n)$ such that $f(n) = \omega(\frac{2}{5} \cdot n^{42})$ and $f(n) = o(\frac{7}{8} \cdot n^{44})$:

$f(n) =$ n^{43}

⇑⇑⇑ _____ / 2 Punkte ⇑⇑⇑

d) Geben Sie eine Funktion $H(k)$ an, sodass
$H(k) = o(10 \cdot k^2)$ und $H(k) = \omega(45 \cdot k^2/(\log(k))^{13})$ gilt:
Enter a function $H(k)$ such that $H(k) = o(10 \cdot k^2)$ and $H(k) = \omega(45 \cdot k^2/(\log(k))^{13})$:

$H(k) =$ $k^2/\log(k)$

⇑⇑⇑ _____ / 2 Punkte ⇑⇑⇑

e) Geben Sie für die folgenden Algorithmen jeweils die Laufzeit als möglichst einfache Funktion abhängig von n in Θ-Notation an. *For the following algorithms, provide the running time as a simple function in terms of n in Θ-notation.*

```
s = n * n
while s > 1 do
    s = s/2
```

$\Theta($ $\log n$ $)$

```
i = 1
while i < n do
    j = n
    while j > 0 do
        j = j - 1
    i = i + 2
```

$\Theta($ n^2 $)$

```
i = 1
while i < n do
    j = n
    while j > 0 do
        j = j - i
    i = i * 2
```

$\Theta($ n $)$

⇑⇑⇑ _____ / 9 Punkte ⇑⇑⇑

Musterlösung

a) Betrachten Sie die folgende Funktion FOO.
Consider the following function FOO.

```
function FOO(N)
    if N > 9 then
        return 3 * FOO(N − 5) + FOO(N − 9) + 2
    else
        return 0
```

Welches ist die korrekte Rekursionsgleichung für die genaue Anzahl $T(N)$ von arithmetischen Operationen (Additionen, Subtraktionen, Multiplikationen, Divisionen), die ein Aufruf FOO(N) verursacht? Der Basisfall ist $T(0) = T(1) = \cdots = T(9) = 0$. Kreuzen Sie nur die korrekte Antwort an:
Which is the correct recurrence equation for the exact number $T(N)$ of arithmetic operations (additions, subtractions, multiplications, divisions) that a call to FOO(N) causes? The base case is $T(0) = T(1) = \cdots = T(9) = 0$. Mark exactly the correct answer:

- ☐ $T(N) = 3 \cdot T(N-5) + T(N-9) + 2$
- ☐ $T(N) = 3 \cdot T(N-5) + T(N-9) + 5$
- ☐ $T(N) = T(N-5) + T(N-9) + 2$
- ☑ $T(N) = T(N-5) + T(N-9) + 5$
- ☐ $T(N) = 3 \cdot \text{FOO}(N-5) + \text{FOO}(N-9) + 2$

⇑⇑⇑ _____ / 2 Punkte ⇑⇑⇑

b) Geben Sie für folgende Rekursionsgleichungen eine geschlossene Form an. Sie können davon ausgehen, dass $n = 2^k$ für eine natürliche Zahl $k > 0$ gilt. Geben Sie das Ergebnis in Θ-Notation abhängig von n an.
For the following recurrence equations, give a closed form. You can assume $n = 2^k$ for a natural number $k > 0$. Give the result in Θ-notation depending on n.

- $B(n) = B(\frac{n}{2}) + 20, \quad B(1) = 9.$

  $B(n) = \Theta($ $\log n$ $)$

- $C(n) = 2 \cdot C(\frac{n}{2}) + (\log n)^9, \quad C(1) = 1.$

  $C(n) = \Theta($ n $)$

- $D(n) = D(n/2) + D(n/2) + 9n^3, \quad D(1) = 1.$

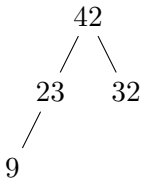
  $D(n) = \Theta($ n^3 $)$

⇑⇑⇑ _____ / 6 Punkte ⇑⇑⇑

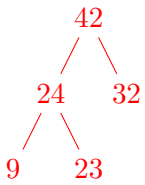
Musterlösung

Aufgabe 3: Anwendungsübungen $1+1+1+1+1+1+1+1+1+1+1+1=12$ Punkte

a) Links abgebildet ist ein **Max-Heap**. Wir rufen INSERT(24) auf. Zeichnen Sie rechts den Max-Heap, der dadurch entsteht.
On the left is a Max-Heap. We call INSERT(24). Draw the resulting Max-Heap on the right.

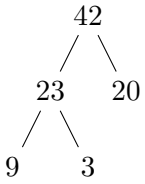


Endergebnis hier zeichnen:

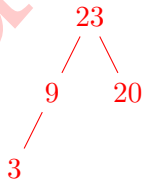


⇑⇑⇑ _____ / 1 Punkte ⇑⇑⇑

b) Links abgebildet ist ein **Max-Heap**. Wir rufen EXTRACTMAX auf. Zeichnen Sie rechts den Max-Heap, der dadurch entsteht.
On the left is a Max-Heap. We call EXTRACTMAX on this Max-Heap. Draw the resulting Max-Heap on the right.

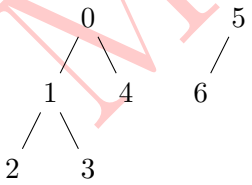


Endergebnis hier zeichnen:

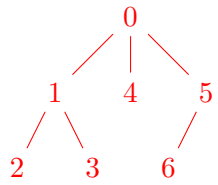


⇑⇑⇑ _____ / 1 Punkte ⇑⇑⇑

c) Links abgebildet ist ein Zustand der **Quick-Union** Datenstruktur. Wir nehmen an, dass die Operation UNION(i,j) immer den durch FIND(i) spezifizierten Knoten als Kind des von FIND(j) spezifierten Knoten anhängt. Wir rufen UNION$(6,2)$ auf. Zeichnen Sie rechts den Zustand, der dadurch entsteht.
*On the left is a state of the **Quick-Union** data structure. We assume that the operation UNION(i,j) always appends the node specified by FIND(i) as a child of the node specified by FIND(j). We call UNION$(6,2)$. Draw the resulting state on the right.*



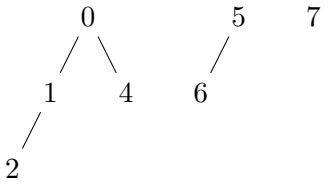
Endergebnis hier zeichnen:



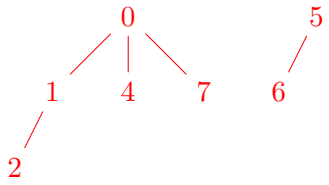
⇑⇑⇑ _____ / 1 Punkte ⇑⇑⇑

Musterlösung

d) Links abgebildet ist ein Zustand der **Weighted Quick-Union** Datenstruktur. Wir rufen UNION(2, 7) auf. Zeichnen Sie rechts den Zustand, der dadurch entsteht.
*On the left is a state of the **Weighted Quick-Union** data structure. We call UNION(2, 7). Draw the resulting state on the right.*

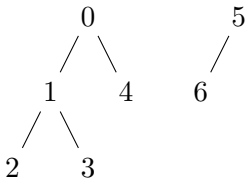


Endergebnis hier zeichnen:

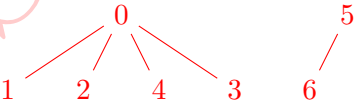


⇑⇑⇑ _____ / 1 Punkte ⇑⇑⇑

e) Links abgebildet ist ein Zustand der **Weighted Quick-Union** Datenstruktur mit Pfadverkürzung. Wir rufen UNION(3,2) auf. Zeichnen Sie rechts den Zustand, der dadurch entsteht.
*On the left is a state of the **Weighted Quick-Union** data structure with path compression. We call UNION(3,2). Draw the resulting state on the right.*



Endergebnis hier zeichnen:



⇑⇑⇑ _____ / 1 Punkte ⇑⇑⇑

f) Wir betrachten jetzt die Datenstruktur **Hashing mit linearem Sondieren**. Als Hashfunktion benutzen wir die Funktion $h\colon \mathbb{N} \to \{0, 1, \ldots, 10\}$ mit $h(x) = (2x) \bmod 11$.
*We now consider the data structure **Hashing with linear probing**. As a hash function we use the function $h\colon \mathbb{N} \to \{0, 1, \ldots, 10\}$ with $h(x) = (2x) \bmod 11$.*

Die Hash-Tabelle hat Platz für elf Einträge und ist derzeit wie folgt gefüllt:
The hash table has room for eleven entries and is currently filled as follows:

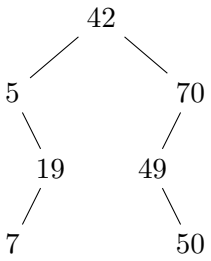
0	1	2	3	4	5	6	7	8	9	10
5								26	4	10

Fügen Sie die Zahlen 10 und 5 in dieser Reihenfolge in die Hashtabelle ein.
Add the numbers 10 and 5 in this order to the hash table.

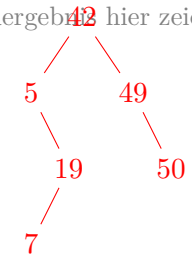
⇑⇑⇑ _____ / 1 Punkte ⇑⇑⇑

Musterlösung

g) Links abgebildet ist ein (nicht notwendigerweise balancierter) **binärer Suchbaum**. Wir rufen DELETE(70) auf. Zeichnen Sie rechts den Zustand, der dadurch entsteht.
*On the left is a (not necessarily balanced) **binary search tree**. We call DELETE(70). Draw the resulting state on the right.*

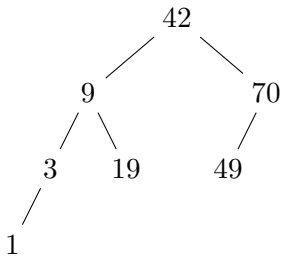


Endergebnis hier zeichnen:

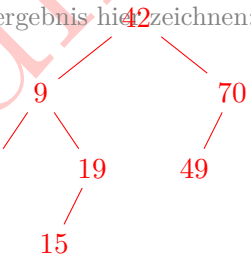


⇑⇑⇑ _____ / 1 Punkte ⇑⇑⇑

h) Links abgebildet ist ein (nicht notwendigerweise balancierter) **binärer Suchbaum**. Wir rufen INSERT(15) auf. Zeichnen Sie rechts den Zustand, der dadurch entsteht.
*On the left is a (not necessarily balanced) **binary search tree**. We call INSERT(15). Draw the resulting state on the right.*

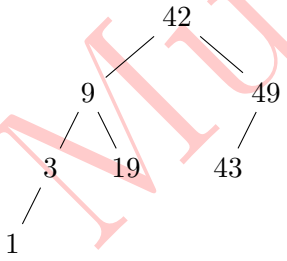


Endergebnis hier zeichnen:

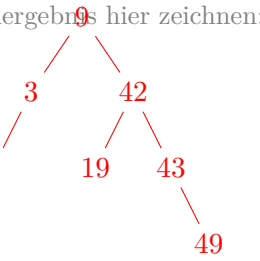


⇑⇑⇑ _____ / 1 Punkte ⇑⇑⇑

i) Links abgebildet ist ein **AVL-Baum**. Wir rufen DELETE(49) und INSERT(49) in dieser Reihenfolge auf. Zeichnen rechts den AVL-Baum, der dadurch entsteht.
*On the left is an **AVL tree**. We call DELETE(49) and INSERT(49) in this order. Draw the resulting AVL-tree on the right.*



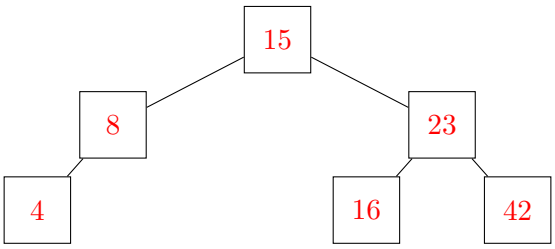
Endergebnis hier zeichnen:



⇑⇑⇑ _____ / 1 Punkte ⇑⇑⇑

Musterlösung

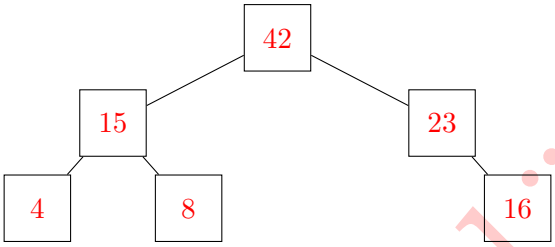
j) Hier ist die Struktur eines Baumes, an dessen Knoten Zahlen gespeichert sind:
Here is the structure of a tree whose nodes store numbers:



Die Inorder-Traversierung (*inorder traversal*) dieses Baums gibt die Zahlen 4, 8, 15, 16, 23, 42 in dieser Reihenfolge aus. Füllen Sie die Zahlen richtig in die Knoten ein.
The inorder traversal of this tree outputs the numbers 4, 8, 15, 16, 23, 42 in this order. Correctly fill in the numbers in the nodes.

⇑⇑⇑ _____ / 1 Punkte ⇑⇑⇑

k) Hier ist die Struktur eines Baumes, an dessen Knoten Zahlen gespeichert sind:
Here is the structure of a tree whose nodes store numbers:



Die Postorder-Traversierung (*postorder traversal*) dieses Baums gibt die Zahlen 4, 8, 15, 16, 23, 42 in dieser Reihenfolge aus. Füllen Sie die Zahlen richtig in die Knoten ein.

The postorder traversal of this tree outputs the numbers 4, 8, 15, 16, 23, 42 in this order. Correctly fill in the numbers in the nodes.

⇑⇑⇑ _____ / 1 Punkte ⇑⇑⇑

l) Gegeben ist ein Text, der aus den Buchstaben `a`, `b`, `c`, `d`, `e`, `f`, `g` besteht. Die Häufigkeit der Buchstaben ist wie folgt:
Given is a text consisting of the letters `a`, `b`, `c`, `d`, `e`, `f`, `g`. The frequency of the letters is as follows:

a	b	c	d	e	f	g
12	12	12	12	23	23	23

Geben Sie einen optimalen präfixfreien Code an, der diese Buchstaben in Bitfolgen übersetzt:
Give an optimal prefix-free code that translates these letters into bit strings:

a	b	c	d	e	f	g
000	001	100	101	110	111	01

⇑⇑⇑ _____ / 1 Punkte ⇑⇑⇑

Musterlösung

Jeder Knoten x in einem Binärbaum hat die Eigenschaften x.`parent`, x.`left` und x.`right`. Wenn der Knoten kein Kind oder keinen Elternknoten hat, ist der jeweilige Wert auf `null` gesetzt (zum Beispiel x.`parent` = `null`).

Schreiben Sie jetzt eine rekursive Funktion NoLeft(x): Die Funktion erhält einen Knoten x als Eingabe. Die Ausgabe der Funktion soll die Anzahl der Knoten im Unterbaum von x sein, die kein linkes Kind haben. (Beachten Sie, dass x die Wurzel des Unterbaums von x ist und also in diesem Unterbaum vorkommt.) **Die Funktion muss rekursiv sein, darf keine anderen Funktionen aufrufen und darf keine Schleifen benutzen.** Vervollständigen Sie den folgenden Code oder Pseudocode (Hinweise: Eine korrekte Lösung mit 3 Zeilen ist möglich. Ihr Code muss sowohl den rekursiven Fall als auch den Basisfall behandeln!):

Each node x in a binary tree has the properties x.`parent`, x.`left` and x.`right`. If the node has no child or parent, the respective value is set to `null` (for example x.`parent` = `null`).

Write a recursive function NoLeft(x): The function receives a node x as input. The output of the function should be the number of nodes in the subtree of x that do not have a left child. (Note that x is the root of the subtree of x and thus occurs in this subtree.) ***The function must be recursive, must not call other functions and must not use loops.*** *Complete the following code or pseudocode (Remarks: A correct solution with 3 lines is possible. Your code must handle both the recursive case and the base case!):*

```
function NoLeft(x)
    if x == null: return 0
    if x.left == null: return 1+NoLeft(x.right)
    return NoLeft(x.left)+NoLeft(x.right)
```

Musterlösung

Das Land *Seenland* wird unter den vielen Touristen, die jedes Jahr kommen, um die Natur und die Sicherheit der ausgezeichneten Infrastruktur zu genießen, zu Recht als *Land der n Seen* bezeichnet. Seenland besteht aus $r \times c$ Seen, die in einem gitterartigen Muster miteinander verbunden sind; die Gesamtzahl der Seen ist also $n = rc$. Seenland ist nicht nur voller Seen, sondern auch hügelig, und jeder See v hat eine Höhe über dem Meeresspiegel von $h(v)$ Metern. (Wir nehmen an, dass alle $h(v)$ ganzzahlig sind.)

Sie sind der neu gewählte Minister für Seesicherheit von Seenland. Sie sind stolz darauf, ausgehandelt zu haben, dass Sie das prestigeträchtige Boot "Seenschiff" als Ihr repräsentatives Gefährt erhalten. Leider kann das "Seenschiff" nicht auf einem See mit hoher Höhe eingesetzt werden.

Sie können annehmen, dass die Eingabe als Klartext in der folgenden Form gegeben ist: Die erste Zeile enthält die ganzen Zahlen r und c. Dann folgen r Zeilen, von denen jede c ganze Zahlen enthält: nämlich die Höhen der Seen in der entsprechenden Zeile. Wir vereinbaren, dass die Seen $v_1, \ldots, v_n$ von links nach rechts, von unten nach oben, wie in unserem Beispiel nummeriert sind.

The country of Seenland *is rightfully called the* Land of n Lakes *among the many tourists visiting every summer, who come to enjoy both nature and the safety of the excellent infrastructure. Seenland consists of $r \times c$ lakes connected in a grid-like fashion; so the total number of lakes is $n = rc$. Not only is Seenland full of lakes, it is also hilly, and each lake v has an elevation of $h(v)$ meters above sea level. (We assume that all $h(v)$ are integers.)*

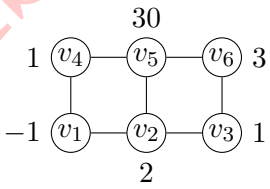
You are the newly-elected Minister of Lake Safety of Seenland. You pride yourself with having negotiated that you get the prestigious boat "Seenschiff" as your representative vehicle. Unfortunately, the "Seenschiff" cannot be used on a lake with high elevation.

You can assume that the input is given in plain text, in the following form: The first line contains the integers r and c. Then follow r lines containing c integers each: namely, the elevations of the lakes in the corresponding row. Let's agree that the lakes are numbered $v_1, \ldots, v_n$, left-to-right, bottom-to-top, as in our example.

—

Beispiel: Die Eingabe links stellt die Landkarte rechts dar:
Example: *The input on the left represents the map on the right:*

```
2 3
1 30 3
-1 2 1
```



—

Hinweis: Verwenden Sie in den folgenden Aufgabenteilen Ihnen bereits bekannte Algorithmen und Datenstrukturen so weit wie möglich, formulieren Sie Ihre Antworten klar und knapp. Beschreiben Sie Ihre Algorithmen mit geeignetem **Pseudocode**. Wenn Sie in mehreren Antworten denselben Algorithmus verwenden, so erwähnen Sie das einfach, etwa mit "siehe 5c", anstatt sich zu wiederholen.
Remark: *In the following questions, use existing algorithms and data structures as much as you can, write clearly, and be brief. Describe your algorithms using suitable **pseudocode**. If you use the same algorithm for some questions, just write that, like "see 5c", instead of repeating yourself.*

Musterlösung

a) Wir nehmen an, dass das "Seenschiff" auf Seen mit einer Höhe von höchstens 50 eingesetzt werden kann und dass $r = 1$ ist, d.h., die Landkarte besteht aus einer einzigen Zeile. (Treffen Sie keine Annahmen über c und h.) Entwerfen Sie einen effizienten Algorithmus, der entscheidet, ob das "Seenschiff" von See v_1, dem linksten See, zu See v_n, dem rechtesten See, fahren kann.

Geben Sie die asymptotische Laufzeit Ihres Algorithmus in Abhängigkeit von den gegebenen Parametern an; ignorieren Sie die Zeit für das Lesen der Eingabe. Schnellere Algorithmen geben mehr Punkte.

Assume that the "Seenschiff" can be used on lakes of elevation 50 or less, and assume that $r = 1$, i.e., the map consists of a single row. (Make no assumptions about c and h.) Design an efficient algorithm that decides if the "Seenschiff" can travel from lake v_1, the leftmost lake, to lake v_n, the rightmost lake.

State the asymptotic running time of your algorithm in terms of the given parameters; ignore the time for reading the input. Faster algorithms give more points.

Musterlösung: Unser Algorithmus muss lediglich überprüfen, dass das Maximum der Höhen aller Seen auf der Route von v_1 nach v_n kleiner oder gleich 50 ist. Als Pseudocode:

```
Lese c aus der zweiten Zahl in der ersten Zeile der Eingabe
Setze n ← c
Lese die Höhen h(v_1), ..., h(v_n) aus der zweiten Zeile der Eingabe
for all i ∈ {1, ..., n} do
    if h(v_i) > 50 then
        return False
return True
```

Die Laufzeit dieses Algorithmus beträgt: $\Theta(n)$.
Alternativ kann auch der Algorithmus aus Aufgabenteil b) für $\ell = 50$ verwendet werden mit der Beobachtung, dass Breitensuche auf dünnen Graphen (und insbesondere auf Pfadgraphen) nur Linearzeit benötigt.

Musterlösung

⇑⇑⇑ _____ / 8 Punkte ⇑⇑⇑

b) Sei ℓ die höchste Höhe, auf der das "Seenschiff" eingesetzt werden kann, und nehmen Sie weiter an, dass ℓ als zusätzlicher Eingabeparameter gegeben ist. Entwerfen Sie einen effizienten Algorithmus, der entscheidet, ob das "Seenschiff" von See v_1 zu See v_n fahren kann, wobei v_1 der See links unten und v_n der See rechts oben auf der Landkarte ist. (Treffen Sie keine Annahmen über r, c und h.) Im obigen Beispiel wäre die korrekte Antwort für $\ell = 2$ nein, für $\ell = 3$ ja.

Geben Sie die asymptotische Laufzeit Ihres Algorithmus in Abhängigkeit von den gegebenen Parametern an; ignorieren Sie die Zeit für das Lesen der Eingabe. Schnellere Algorithmen geben mehr Punkte.

Let ℓ (as in "limit") be the highest elevation the "Seenschiff" can be used at, and assume that ℓ is given as an additional input parameter. Design an efficient algorithm that decides whether the "Seenschiff" can travel from lake v_1, the lower left corner of the map, to lake v_n, the upper right corner of the map. (Make no assumptions about r, c and h.) In the above example, the correct answer for $\ell = 2$ would be no, for $\ell = 3$ it would be yes.

State the asymptotic running time of your algorithm in terms of the given parameters; ignore the time for reading the input. Faster algorithms give more points.

Musterlösung: Unser Algorithmus führt eine Breitensuche oder Tiefensuche auf dem Graphen durch, der die Seen mit Höhe kleiner als ℓ als Knoten und die Verbindungen zwischen diesen Seen als Kanten hat. Die Suche startet bei v_1 und liefert genau dann `True`, wenn sie v_n erreicht. Als Pseudocode:

- Lese r und c aus der ersten Zeile der Eingabe
- Setze $n \leftarrow rc$
- Lese die Höhen $h(v_1), \ldots, h(v_n)$ aus den folgenden r Zeilen der Eingabe
- Sei $V_{\text{all}} = \{v_1, \ldots, v_n\}$ die Menge aller Seen
- Sei G der ungerichtete Graph mit Knotenmenge $V(G) = \{v \in V_{\text{all}} \mid h(v) < \ell\}$ und Kantenmenge

$$E(G) = \left\{ \{v_i, v_j\} \in \binom{V(G)}{2} \;\middle|\; j = i + 1 \text{ oder } j = i + c \right\}.$$

  Das heißt, zwei Knoten v_i und v_j sind benachbart in G wenn sie benachbart im Gitter sind.
- Nun gilt: Es gibt eine gültige Route für das Seenschiff gdw. es einen Pfad von v_1 zu v_n in G gibt. Wir starten also eine Breitensuche von v_1. Falls diese irgendwann v_n erreicht, geben wir `True` zurück, ansonsten `False`.

Die Vorbereitung sowie Breitensuche haben eine Laufzeit von $\Theta(|V(G)| + |E(G)|)$. Da $|E(G)| \leq 2n$ gilt, ergibt sich eine Laufzeit von $\Theta(n)$.

Musterlösung

⇑⇑⇑ _____ / 16 Punkte ⇑⇑⇑

c) Ihre Ingenieure berichten stolz, dass sie eine Möglichkeit gefunden haben, die höchste Höhe ℓ zu ändern, auf der das "Seenschiff" eingesetzt werden kann. Die Kosten dafür hängen von ℓ ab, je höher, desto teurer. Entwerfen Sie einen effizienten Algorithmus, der das kleinste ℓ bestimmt, so dass das "Seenschiff" von See v_1 zu See v_n fahren kann, wobei v_1 der See links unten und v_n der See rechts oben auf der Landkarte ist. (Treffen Sie keine Annahmen über r, c und h.) Im obigen Beispiel wäre die korrekte Antwort 3.

Geben Sie die asymptotische Laufzeit Ihres Algorithmus in Abhängigkeit von den gegebenen Parametern an; ignorieren Sie die Zeit für das Lesen der Eingabe. Schnellere Algorithmen geben mehr Punkte.

Your engineers proudly report that they found a possibility to change the highest elevation ℓ the "Seenschiff" can operate on. The cost of doing so depends on ℓ, the higher, the more expensive. Design an efficient algorithm that determines the smallest ℓ such that the "Seenschiff" can travel from lake v_1, the lower left corner of the map, to lake v_n, the upper right corner of the map. (Make no assumptions about r, c and h.) In the above example, the correct answer would be 3.

State the asymptotic running time of your algorithm in terms of the given parameters; ignore the time for reading the input. Faster algorithms give more points.

Musterlösung

Musterlösung: Der folgende Algorithmus löst das Problem in Zeit $O(n \log n)$:

- Lies die Eingabe wie in Aufgabenteil b) ein. Falls $n = 1$ gilt, gib $h(v_1)$ zurück.
- Definiere den Graphen G mit $V(G) = \{v_1, \ldots, v_n\}$ und $E(G) = \{\{v_i, v_j\} \mid j = i + 1 \text{ oder } j = i + c\}$. Somit gibt es n Knoten und $\Theta(n)$ Kanten.
- Definiere die Kantengewichte $w(\{u, w\})$ mit $w(\{u, w\}) = \max(h(u), h(w))$ für alle $\{u, w\} \in E(G)$.
- Berechne den minimalen Spannbaum T von G bezüglich der Kantengewichte w. (z.B. mit Hilfe von Kruskal's Algorithmus in Zeit $O(n \log n)$).
- Bestimme mit BFS den eindeutigen Pfad von v_1 nach v_n in T und gib die maximale Höhe auf diesem Pfad zurück.

Die Gesamtlaufzeit wird dominiert von der Laufzeit von Kruskal's Algorithmus, also $O(n \log n)$.

— Ein noch effzienterer Algorithmus ist möglich: Definiere den Graphen G analog zu Aufgabenteil b), allerdings auf der gesamten Knotenmenge $V(G) = V_{\text{all}}$. Sei m die Anzahl *verschiedener* Höhen von Seen v_i. Wir berechnen nun zu den Höhen aller Knoten den Rang in der sortierten Folge der Höhen ohne Duplikate. Dies ist zum Beispiel möglich, indem die Höhen aller Knoten nacheinander mit Duplikatprüfung in einen AVL-Baum eingefügt werden und am Ende mit einem Inorder-Durchlauf dieses AVL-Baums die Ränge der verschiedenen Höhen bestimmt werden. Dabei wird für jeden Knoten von G der zugehörige Baumknoten in dem erstellten AVL-Baum gespeichert, sodass wir am Ende zu jedem Knoten $v \in V(G)$ den Rang $h'(v)$ der Höhe von v zur Verfügung haben. Die Laufzeit für das Einfügen aller Höhen in den AVL-Baum ohne Duplikate ist $\Theta(n \log m)$ und das anschließende Bestimmen der Ränge benötigt noch einmal $\Theta(n \log m)$.

Es gilt nun: Eine Route π von v_1 nach v_n für das Seenschiff hat minimale Höhe (maximales $h(v)$ für einen Knoten v auf der Route) unter allen solchen Routen gdw. π eine solche Route mit minimalem Höhenrang (maximales $h'(v)$ für einen Knoten v auf der Route) ist. Dies ist der Fall, da uns nur interessiert, welches die maximale Höhe auf der Route ist – dafür spielen die konkreten Höhen keine Rolle, sondern nur die Ränge der Höhen.

Die Routen entsprechen wie in b) genau den Pfaden im Graphen G. Ein v_1-v_n-Pfad von minimaler Höhe h' kann nun bestimmt werden, indem mit binärer Suche der kleinste Höhenrang gesucht wird, für den uns Breitensuche einen Pfad von höchstens diesem Höhenrang liefert. Genauer: In einer äußeren Schleife verdoppeln wir beginnend bei $\ell = 1$ jeweils den maximal erlaubten Höhenrang bis wir ein Intervall $\{2^{i-1}, \ldots, 2^i\}$ gefunden haben, in dem der benötigte Rang liegt. Anschließend suchen wir in diesem Intervall weiter mit binärer Suche. Für jeden Durchlauf dieser äußeren Schleife prüfen wir wie in Aufgabenteil b) mit Breitensuche, ob es einen entsprechenden Pfad gibt, der den aktuellen Höhenrang nicht überschreitet. Dabei benutzen wir nun aber die Höhen h' anstatt der Höhen h. Dieser Schritt benötigt Laufzeit $\Theta(n \log m)$.

Schließlich lesen wir in $\Theta(n)$ den gefundenen Pfad minimalen Rangs mit Backtracking aus dem Ergebnis der Breitensuche aus und bestimmen die minimale Höhe auf diesem Pfad.

Insgesamt ergibt sich damit eine Laufzeit von $O(n \log m) \subseteq O(n \log n) \cap O(n \log(\max_i\{h(v_i)\}))$.

Musterlösung

⇑⇑⇑ _____ / 16 Punkte ⇑⇑⇑

Notizseite

Wichtig: Lösungen auf dieser Seite werden nur dann berücksichtigt, wenn bei der entsprechenden Aufgabe ein Verweis zu Seite 16 platziert wurde.

Important: *Solutions on this page will only be considered if a reference to page 16 was placed at the corresponding task.*

Musterlösung

Wichtig: Lösungen auf dieser Seite werden nur dann berücksichtigt, wenn bei der entsprechenden Aufgabe ein Verweis zu Seite 17 platziert wurde.

Important: *Solutions on this page will only be considered if a reference to page 17 was placed at the corresponding task.*

Musterlösung

Notizseite

Wichtig: Lösungen auf dieser Seite werden nur dann berücksichtigt, wenn bei der entsprechenden Aufgabe ein Verweis zu Seite 18 platziert wurde.

Important: *Solutions on this page will only be considered if a reference to page 18 was placed at the corresponding task.*

Musterlösung

Notizseite

Wichtig: Lösungen auf dieser Seite werden nur dann berücksichtigt, wenn bei der entsprechenden Aufgabe ein Verweis zu Seite 19 platziert wurde.

Important: *Solutions on this page will only be considered if a reference to page 19 was placed at the corresponding task.*

Musterlösung