Algorithmen und Datenstrukturen 1

SoSe 2021 · Prof. Dr. H. Dell
9. August 2021
Klausurnummer: **0000**

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Nachname (Druckschrift)**:** ____________________

Vorname (Druckschrift)**:** ____________________

Matrikelnummer: ____________________

Studiengang: ____________________

Bitte Hinweise beachten:

- Schreiben Sie Ihren Namen **nur auf dieses Titelblatt**. Sollten Sie Zusatzpapier bekommen, vermerken Sie darauf unbedingt die Klausurnummer <u>**0000**</u>.
- Merken oder notieren Sie sich Ihre Klausurnummer <u>**0000**</u>, da nur unter dieser Nummer die Ergebnisse veröffentlicht werden.
- Es dürfen nur **dokumentenechte Stifte** in den Farben blau und schwarz verwendet werden. Insbesondere ist die Nutzung von Tintenlöschern und Tipp-Ex untersagt. Zugelassene Hilfsmittel:
  1 Blatt DIN A4 mit handschriftlichen Notizen (beidseitig).
- Das Mitbringen nicht zugelassener Hilfsmittel stellt eine Täuschung dar und führt zum Nichtbestehen der Klausur. **Schalten Sie bitte deshalb alle elektronischen Geräte, insbesondere Handys und Smartwatches, vor Beginn der Klausur aus und packen Sie diese weg.**
- Bitte benutzen Sie Rückseiten und die beigefügten Zusatzblätter. Weitere Blätter sind bei Bedarf erhältlich. Das Benutzen eigens mitgebrachter Blätter ist untersagt.
- Wenn sich Ihre Lösung zu einer Aufgabe teilweise oder ganz auf Rückseiten oder Zusatzblättern befindet, vermerken Sie dies entsprechend bei der Aufgabe.
- 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 ist mit Sicherheit bestanden, wenn mindestens **50%** der Höchstpunktzahl erreicht wird. Die Klausur dauert 180 Minuten.

2 111000 000001

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

Aufgabe	1	2	3	4	5	6	7	8
Erreichbar	16	9	7	13	12	16	11	16
Erreicht								

Klausur	Bonifikation	Summe	Note

a) Sei n eine natürliche Zahl. Wie viele Sterne (*) gibt der folgende Code aus?

```
for i = 1, 2, ..., 2n do
    if i ist gerade then
        print "**"
    else
        print "*"
```

Es werden genau __________ Sterne ausgegeben. (Gesucht ist die genaue Anzahl in Abhängigkeit von n, nicht die asymptotische Anzahl in O- oder Θ-Notation.)

⇑⇑⇑ ______ / 2 Punkte ⇑⇑⇑

b) Geben Sie für die folgenden Algorithmen jeweils die Laufzeit in Θ-Notation abhängig von n an.

```
for i = 1, 2, ..., n do
    for j = 1, 2, ..., n do
        if i == j then
            print "*"
```

```
s = n
while s ≥ 0 do
    for i = 0, 1, ..., s do
        print "*"
    s = s − 5
```

```
s = 1
while s ≤ (log n)^2 do
    s = 3s
```

$\Theta\big($ ________________ $\big)$ $\Theta\big($ ________________ $\big)$ $\Theta\big($ ________________ $\big)$

⇑⇑⇑ ______ / 6 Punkte ⇑⇑⇑

c) Sei $f(n) = 100n\log n + n^2\log n + \frac{1}{2}n + n/\log\log n$. Welches asymptotische Wachstum ist für diese Funktion richtig? Kreuzen Sie genau eine Box an.

□ $\Theta(n/\log\log n)$ □ $\Theta(n\log n)$ □ $\Theta(n)$ □ $\Theta(n^2\log n)$

⇑⇑⇑ ______ / 2 Punkte ⇑⇑⇑

d) Sei $f(n) = \sqrt{n}\log n + n^{2/3} - 99\sqrt{n} - \log\log n$. Welches asymptotische Wachstum ist für diese Funktion richtig? Kreuzen Sie genau eine Box an.

□ $\Theta(\sqrt{n})$ □ $\Theta(n^{2/3})$ □ $\Theta(\log\log n)$ □ $\Theta(\sqrt{n}\log n)$

⇑⇑⇑ ______ / 2 Punkte ⇑⇑⇑

e) Welche der folgenden Funktionen von n ist asymptotisch **am Größten**? Kreuzen Sie genau eine Box an.

□ $\left(\sqrt{n} + 19\,(\log_4 n)^2\right)^3$ □ $0.1 \cdot n/\sqrt{\log_{10} n}$ □ $63 \cdot \sqrt{n} + \log_2(999^n)$

⇑⇑⇑ ______ / 2 Punkte ⇑⇑⇑

f) Welche der folgenden Funktionen von n ist asymptotisch **am Kleinsten**? Kreuzen Sie genau eine Box an.

□ $\sum_{i=1}^{n}(i+10)$ □ $\log_2(\log_2(\log_2(n)))$ □ $\sum_{i=1}^{n^{99}} \frac{1}{7^i}$

⇑⇑⇑ ______ / 2 Punkte ⇑⇑⇑

a) Betrachten Sie die folgende Funktion Foo.

function FOO(n)
  if $n \leq 0$ **then**
    return 42
  else
    return $2 \cdot \text{FOO}(n-1) + 5$

Geben Sie eine Rekursionsgleichung für die **genaue** Anzahl $A(n)$ von arithmetischen Operationen (Additionen, Subtraktionen, Multiplikationen und Divisionen) an, die ein Aufruf von FOO(n) verursacht. Der Basisfall ist $A(0) = 0$.

$A(n) =$ ____________________

⇑⇑⇑ ______ / 3 Punkte ⇑⇑⇑

b) Geben Sie für folgende Rekursionsgleichungen eine geschlossene Form an.

Sie können bei B und C davon ausgehen, dass $n = 7^k$ für eine natürliche Zahl $k > 0$ gilt, und bei D, dass $n = 7k$ für eine natürliche Zahl $k > 0$ gilt. Geben Sie das Ergebnis in Θ-Notation abhängig von n an.

- $B(n) = 7 \cdot B(\frac{n}{7}) + n, \quad B(1) = 1.$

  $B(n) = \Theta\Big($ ____________________ $\Big)$

- $C(n) = C(\frac{n}{7}) + \log_7 n, \quad C(1) = 1.$

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

- $D(n) = 77 \cdot D(n-7) + 77, \quad D(0) = 77.$

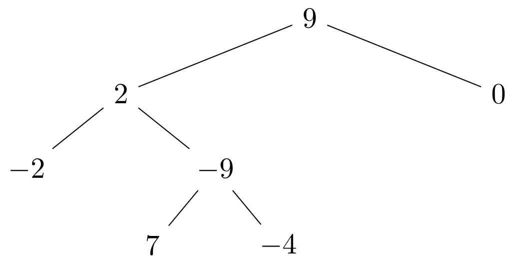
  $D(n) = \Theta\Big($ ____________________ $\Big)$

⇑⇑⇑ ______ / 6 Punkte ⇑⇑⇑

Aufgabe 3: Algorithmen auf Bäumen — $2+3+2=7$ Punkte

Sei T ein Binärbaum. Jeder Knoten x von T hat die Eigenschaften x.`parent`, x.`left` und x.`right`, welche auf den Elternknoten sowie auf das linke und rechte Kind von x verweisen. Wenn der Knoten keine Kinder hat (z.B. die Blätter) oder keinen Elternknoten hat (die Wurzel), wird der jeweilige Wert auf `null` gesetzt. Des Weiteren hat jeder Knoten x eine Eigenschaft x.`number`, die eine einzelne Zahl speichert. Betrachten Sie die folgende Funktion FUNC und den abgebildeten Beispielbaum, in dem jeder Knoten mit x.`number` markiert ist.

```
function FUNC(x)
    if x ≠ null then
        if x.number > 0 then
            return x.number + FUNC(x.left) + FUNC(x.right)
        else
            FUNC(x.left)
            print x.number
            FUNC(x.right)
            return 0
    else
        return 0
```



a) Welche Zahl liefert FUNC(v) über die **return** Anweisung als Rückgabewert, wenn v die Wurzel des Beispielbaums ist?

(________________)

⇑⇑⇑ ______ / 2 Punkte ⇑⇑⇑

b) Welche Zahlen gibt FUNC(v) über die **print** Anweisungen aus, wenn v die Wurzel des Beispielbaums ist? Geben Sie die Zahlen in derselben Reihenfolge an, in der sie ausgegeben werden.

(________________________________)

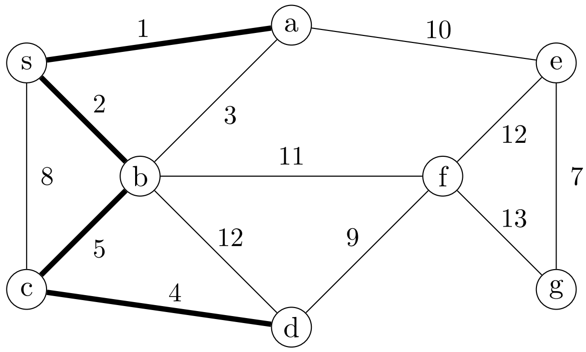
⇑⇑⇑ ______ / 3 Punkte ⇑⇑⇑

c) Welche Laufzeit benötigt der Aufruf FUNC(x), wenn x die Wurzel eines beliebigen Binärbaums mit n Knoten ist? Geben Sie die Laufzeit in Θ-Notation abhängig von n an.

Θ(________________)

⇑⇑⇑ ______ / 2 Punkte ⇑⇑⇑

Wir lassen Prims, Kruskals und Dijkstras Algorithmus auf diesem ungerichteten, gewichteten Graphen laufen:



a) Prims Algorithmus wurde mit Wurzel s gestartet. Die Abbildung zeigt den Zustand von Prims Algorithmus, nachdem vier Kanten in den Baum eingefügt wurden (fett gezeichnet). Welche Kante wird Prims Algorithmus als Nächstes einfügen?

Antwort: __________

⇑⇑⇑ ______ / 3 Punkte ⇑⇑⇑

b) Dieselbe Abbildung zeigt den Zustand von Kruskals Algorithmus, nachdem vier Kanten in den Wald eingefügt wurden (fett gezeichnet). Welche Kante wird Kruskals Algorithmus als Nächstes einfügen?

Antwort: __________

⇑⇑⇑ ______ / 3 Punkte ⇑⇑⇑

c) Dijkstras Algorithmus wurde mit Wurzel s gestartet. Dieselbe Abbildung zeigt jetzt den Zustand von Dijkstras Algorithmus, nachdem vier Kanten in den Baum eingefügt wurden (fett gezeichnet). Welche Kante wird Dijkstras Algorithmus als Nächstes einfügen?

Antwort: __________

⇑⇑⇑ ______ / 3 Punkte ⇑⇑⇑

d) Wir entfernen uns jetzt von dem Beispiel. Beschreiben Sie **in Prosa** einen effizienten Algorithmus, der für einen gegebenen ungerichteten und gewichteten Graphen G und einen Knoten s ermittelt, ob es einen minimalen Spannbaum in G gibt, der gleichzeitig ein Baum kürzester Wege von s ist. Sie dürfen hierbei annehmen, dass alle Kantengewichte verschieden sind. Begründen Sie, warum der Algorithmus korrekt ist.

⇑⇑⇑ ______ / 4 Punkte ⇑⇑⇑

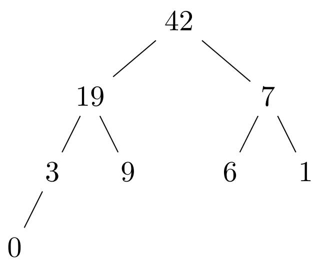
a) **Union-Find** ist eine abstrakte Datenstruktur, die eine dynamische Familie von disjunkten Mengen verwaltet, deren Vereinigung $\{0, \ldots, n-1\}$ ist. Wir verwenden die Operationen der Union-Find Datenstruktur wie folgt:

$\textsc{Init}(10)$
$\textsc{Union}(0, 9)$
$\textsc{Union}(7, 4)$
$\textsc{Union}(9, 0)$
$\textsc{Union}(2, 3)$
$\textsc{Union}(4, 5)$
$\textsc{Union}(6, 5)$
$\textsc{Union}(7, 6)$
$\textsc{Union}(3, 4)$

Wie viele Mengen enthält die Familie jetzt? ____________________

⇑⇑⇑ ______ / 2 Punkte ⇑⇑⇑

b) Der **Max-Heap** ist eine konkrete Datenstruktur, die eine Prioritätswarteschlange implementiert. Hier ist ein Max-Heap mit acht Elementen abgebildet:

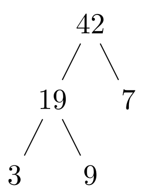


Wir stellen den Max-Heap wie in der Vorlesung beschrieben durch ein Feld dar. Welche Einträge hat das Feld? Schreiben Sie die Einträge in die folgende Tabelle.

0	1	2	3	4	5	6	7	8
–								

⇑⇑⇑ ______ / 2 Punkte ⇑⇑⇑

c) Hier ist ein **Max-Heap** (mit fünf Elementen abgebildet:

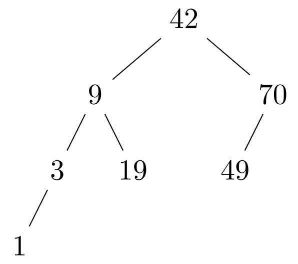


Endergebnis hier zeichnen:

Wir rufen jetzt $\textsc{ExtractMax}()$ und $\textsc{Insert}(10)$ in dieser Reihenfolge auf. Zeichnen Sie im selben Stil den Baum, der dadurch am Ende entsteht.

⇑⇑⇑ ______ / 2 Punkte ⇑⇑⇑

d) Ein **binärer Suchbaum** ist eine konkrete Datenstruktur, die eine Menge von sortierten Daten verwaltet. Hier ist ein binärer Suchbaum mit sieben Elementen abgebildet:

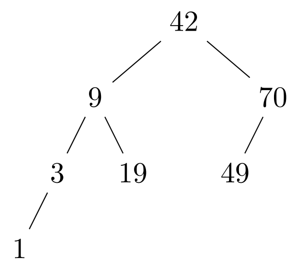


Endergebnis hier zeichnen:

Wir rufen jetzt Delete(42) und Insert(10) in dieser Reihenfolge auf. Zeichnen Sie im selben Stil den binären Suchbaum, der dadurch am Ende entsteht.

⇑⇑⇑ ______ / 2 Punkte ⇑⇑⇑

e) Ein **AVL-Baum** ist eine konkrete Datenstruktur, die eine Menge von sortierten Daten verwaltet. Hier ist ein AVL-Baum mit sieben Elementen abgebildet:



Endergebnis hier zeichnen:

Wir rufen jetzt Delete(42) und Insert(10) in dieser Reihenfolge auf. Zeichnen Sie im selben Stil den AVL-Baum, der dadurch am Ende entsteht.

⇑⇑⇑ ______ / 2 Punkte ⇑⇑⇑

f) **Hashing mit linearem Sondieren** ist eine konkrete Datenstruktur, die eine dynamische Menge von Elementen verwaltet. Als Hashfunktion benutzen wir die Funktion $h\colon \mathbb{N} \to \{0, 1, \ldots, 10\}$ mit

$$h(x) = (2x) \bmod 11 \,.$$

Die Hash-Tabelle hat Platz für elf Einträge und ist derzeit wie folgt gefüllt:

0	1	2	3	4	5	6	7	8	9	10
33					30		42			

Fügen Sie nacheinander 5, 6, 19, 16 in dieser Reihenfolge in die Hashtabelle ein.

⇑⇑⇑ ______ / 2 Punkte ⇑⇑⇑

Der kleine Fahrradladen AlgoBike macht Schrotträder wieder fit, kann aber immer nur ein Fahrrad gleichzeitig reparieren. AlgoBike ist sehr beliebt, daher haben die Eigentümerinnen Schwierigkeiten, den Überblick über die n Aufträge zu behalten. Jeder Auftrag x, der erfüllt wird, liefert einen Gewinn von x`.gewinn`. Zu jedem Zeitpunkt will AlgoBike den Auftrag mit dem größten Gewinn bearbeiten.

a) Sei n die Zahl der Aufträge. Wir möchten die beiden Funktionen Insert und NextOrder implementieren. Hierbei fügt Insert(x) den Auftrag x in die Datenstruktur ein, und NextOrder liefert über eine "return"-Anweisung den Auftrag zurück, der den größten Gewinn hat, und löscht ihn aus der Datenstruktur. Welche aus der Vorlesung bekannte **konkrete** Datenstruktur eignet sich, um die beiden Funktionen in Zeit $O(\log n)$ zu unterstützen?

Antwort: ______________________________

⇑⇑⇑ ______ / 2 Punkte ⇑⇑⇑

b) Implementieren Sie die beiden Funktionen unter Zuhilfenahme der Operationen der Datenstruktur. (z.B., falls die Datenstruktur ein Stapel ist, dürfen Sie die Funktionen Push und Pop verwenden.)

function Insert(x)

function NextOrder

⇑⇑⇑ ______ / 3 Punkte ⇑⇑⇑

c) Begründen Sie, warum die Laufzeit Ihres Insert-Algorithmus im schlimmsten Fall $\Omega(\log n)$ ist.

⇑⇑⇑ ______ / 3 Punkte ⇑⇑⇑

d) Die Eigentümerinnen von AlgoBike können sich besser motivieren, wenn sie zu jedem Zeitpunkt den noch möglichen, zukünftigen Gesamtgewinn vor Augen haben. Sie möchten eine Funktion MOTIVATION nutzen, die die Summe der Gewinne aller noch nicht bearbeiteten Aufträge zurückliefert. Wie muss die Datenstruktur aus b) modifiziert werden, damit MOTIVATION in Zeit $O(1)$ abläuft? Beschreiben Sie, was die Datenstruktur jetzt noch zusätzlich speichern soll und welche der beiden existierenden Funktionen wie geändert werden müssen. Hierbei darf sich die asymptotische Laufzeit der beiden existierenden Funktionen nicht ändern.

$\Uparrow\Uparrow\Uparrow$ ______ / 4 Punkte $\Uparrow\Uparrow\Uparrow$

e) Wir entfernen uns jetzt von dem Fahrradladen. Wir betrachten einfach verkettete Listen. Jedes Element x einer verketteten Liste hat eine Eigenschaft x`.next`, die auf das nächste Element zeigt. Wenn x das letzte Element der Liste ist, gilt x`.next == null`. Beschreiben Sie **in Pseudocode** einen Algorithmus DECIMATE, der jedes zehnte Element aus der Liste löscht. Das heißt, wenn x das erste Element der Liste ist und DECIMATE(x) ausgeführt wurde, dann wird das zehnte Element und das zwanzigste Element und das dreißigste Element etc. gelöscht.

function DECIMATE(x)

$\Uparrow\Uparrow\Uparrow$ ______ / 4 Punkte $\Uparrow\Uparrow\Uparrow$

2 111000 000001

Gegeben ist ein Feld `A[1..n]` der Länge $n \geq 3$, welches positive ganze Zahlen enthält. Das Feld stellt das Höhenprofil einer Landschaft dar. Das Ziel ist es, einen Hügel zu finden, der eine möglichst große symmetrische Umgebung um sich hat, sodass die Aussicht besonders schön ist.

Eine Position h ist hierbei ein *Hügel*, wenn $A[h]$ größer gleich ist als seine beiden Nachbarpositionen $A[h-1]$ und $A[h+1]$, sofern diese existieren. Eine *Hügelumgebung* der Größe i ist ein Teilfeld $A[h-i..h+i]$ für ein i, sodass h ein Hügel ist und $h - i \geq 1$ und $h + i \leq n$ gilt. Eine Hügelumgebung $A[h-i..h+i]$ ist *symmetrisch*, wenn sie ein Palindrom ist, das heißt, $A[h-j] = A[h+j]$ gilt für alle j mit $0 \leq j \leq i$.

a) Als Beispiel sei folgende Landschaft `A[1..9]` gegeben:

3 1 9 20 9 1 9 20 9

Welche Größe i hat in diesem Beispiel die größte symmetrische Hügelumgebung?

Antwort: __________

⇑⇑⇑ ______ / 2 Punkte ⇑⇑⇑

b) Beschreiben Sie kurz und präzise **in Pseudocode** einen rekursiven Algorithmus RecUmgebung. Hierbei soll RecUmgebung(`A, x, y`) die Größe i einer größten symmetrischen Hügelumgebung liefern für einen Hügel h mit $h - i \geq x$ und $h + i \leq y$. Falls kein Hügel zwischen x und y existiert, soll RecUmgebung(`A, x, y`) minus unendlich zurückliefern. Der Algorithmus muss korrekt sein, aber darf beliebig ineffizient sein. Die Basisfälle sind bereits angegeben.

```
function RecUmgebung(A, x, y)
    if y < x then
        return −∞
    else if x == y then
        if x ist ein Hügel then
            return 0
        else
            return −∞
    else
```

⇑⇑⇑ ______ / 4 Punkte ⇑⇑⇑

c) Beschreiben Sie **in Pseudocode** einen iterativen Algorithmus, der die Größe einer größten symmetrischen Hügelumgebung findet. Der Algorithmus muss Laufzeit $O(n^2)$ haben.

function ITERUMGEBUNG(`A, n`)

⇑⇑⇑ ______ / 5 Punkte ⇑⇑⇑

Die Welt `W`$[1..Z][1..S]$ ist ein zweidimensionales Feld, das aus Z Zeilen und S Spalten besteht. Jede Position `W`$[i][j]$ ist das Zeichen `.` ("Wasser"), `O` ("Land"), `V` ("Virus"), oder `M` ("Mensch"). Anfangs gibt es genau ein `V` und ein `M`, dann verbreitet sich das Virus mit der Zeit. In jedem Schritt verbreitet sich das Virus zu allen benachbarten Positionen, die nicht Wasser sind.

Zum Beispiel entwickelt sich eine kleine Welt mit $Z = 1$ und $S = 10$ so:

`.O.OVOO.M.` $\longrightarrow$ `.O.VVVO.M.` $\longrightarrow$ `.O.VVVV.M.`

Der Prozess endet an dieser Stelle und der Mensch wird niemals infiziert.

Hier sind drei Entwicklungsschritte in einer Welt mit $Z = 4$ und $S = 6$ abgebildet:

```
OOO..O         OOV..O         OVV..O         VVV..O
OOV...    →    OVV...    →    VVV...    →    VVV...
.OOO.M         .OVO.M         .VVV.M         .VVV.M
O.OOOO         O.OOOO         O.VOOO         O.VVOO
```

Der Prozess endet nach diesen drei Schritten nicht, sondern geht noch weiter. Sie können sich davon überzeugen, dass der Mensch irgendwann infiziert wird.

Genauer gesagt verhält sich der Prozess wie folgt: Eine Position, die mit `O` oder `M` markiert ist, wird zu `V` in Runde i genau dann, wenn mindestens eine der bis zu vier benachbarten Positionen (im Norden, Süden, Osten oder Westen) in Runde $i - 1$ mit `V` markiert ist. In diesem Fall sagen wir, dass die Position in Runde i infiziert wird. Im größeren Beispiel kann man an der Position unten links erkennen, dass die Infektion nicht diagonal verläuft. Keine Position kann jemals von `V` zu etwas anderem werden, und Wasser (`.`) ändert sich nie.

a) Wir betrachten zunächst den Spezialfall $Z = 1$. Dann ist die Welt ein eindimensionales Feld `W`$[1..S]$. Beschreiben Sie kurz und präzise **in Pseudocode** einen effizienten Algorithmus IslandInfection, der feststellt, ob der Mensch irgendwann infiziert wird. Das heißt, IslandInfection(`W`, S) soll `True` zurückliefern genau dann, wenn der Mensch irgendwann infiziert wird. (Nur Pseudocode wird bewertet, Prosa nicht. Der Algorithmus muss korrekt und effizient sein. Schreiben Sie den Entwurf auf ein Schmierblatt und kopieren Sie Ihre Lösung sauber hier hin.)

function IslandInfection(`W`$[1..S]$, S)

⇑⇑⇑ ______ / 6 Punkte ⇑⇑⇑

b) Welche Laufzeit hat Ihr Algorithmus aus a)? Geben Sie die Laufzeit ohne Beweis in Θ-Notation abhängig von S an.

$\Theta\Big($____________________$\Big)$

⇑⇑⇑ ______ / 2 Punkte ⇑⇑⇑

c) Beschreiben Sie kurz und präzise **in Prosa** einen Algorithmus, der das allgemeine Problem effizient löst. Also IslandInfection(`W`$[1..Z][1..S], Z, S$) soll `True` zurückliefern genau dann, wenn der Mensch irgendwann infiziert wird. (Der Algorithmus muss korrekt und effizient sein. Falls ein Algorithmus aus der Vorlesung verwendet wird, benennen Sie diesen und beschreiben Sie genau, wie er benutzt wird, auf welcher Eingabe er aufgerufen wird, und wie die Ausgabe interpretiert wird. Falls ein Algorithmus aus der Vorlesung verändert wird, benennen Sie diesen und beschreiben Sie genau, wie Sie ihn verändern. Schreiben Sie den Entwurf auf ein Schmierblatt und kopieren Sie Ihre Lösung sauber hier hin.)

⇑⇑⇑ ______ / 6 Punkte ⇑⇑⇑

d) Welche Laufzeit hat Ihr Algorithmus aus c)? Geben Sie die Laufzeit ohne Beweis in Θ-Notation abhängig von Z und S an.

$\Theta\Big($____________________$\Big)$

⇑⇑⇑ ______ / 2 Punkte ⇑⇑⇑

Wichtig: Lösungen auf dieser Seite werden nur dann berücksichtigt, wenn bei der entsprechenden Aufgabe ein Verweis zu Seite 15 platziert wurde.

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