Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Prüfung **(Lösung)**

Informatik II (D-BAUG)

Felix Friedrich, Hermann Lehner, Departement Informatik

ETH Zürich, 28.1.2019.

Name, Vorname:	.....
Legi-Nummer:	.....

Ich bestätige mit meiner Unterschrift, dass ich diese Prüfung unter regulären Bedingungen ablegen konnte und dass ich die allgemeinen Richtlinien gelesen und verstanden habe.

I confirm with my signature that I was able to take this exam under regular conditions and that I have read and understood the general guidelines.

Unterschrift:

Allgemeine Richtlinien:

1. Dauer der Prüfung: 60 Minuten.
2. Erlaubte Unterlagen: Wörterbuch (für von Ihnen gesprochene Sprachen). 4 A4 Seiten handgeschrieben oder $\geq$ 11pt Schriftgrösse.
3. Benützen Sie einen Kugelschreiber (blau oder schwarz) und keinen Bleistift. Bitte schreiben Sie leserlich. Nur lesbare Resultate werden bewertet.
4. Lösungen sind direkt auf das Aufgabenblatt in die dafür vorgesehenen Boxen zu schreiben (und direkt darunter, falls mehr Platz benötigt wird). Ungültige Lösungen sind deutlich durchzustreichen! Korrekturen bei Multiple-Choice Aufgaben bitte unmissverständlich anbringen!
5. Es gibt keine Negativpunkte für falsche Antworten.
6. Störungen durch irgendjemanden oder irgendetwas melden Sie bitte sofort der Aufsichtsperson.
7. Wir sammeln die Prüfung zum Schluss ein. Wichtig: Stellen Sie unbedingt selbst sicher, dass Ihre Prüfung von einem Assistenten eingezogen wird. Stecken Sie keine Prüfung ein und lassen Sie Ihre Prüfung nicht einfach am Platz liegen. Dasselbe gilt, wenn Sie früher abgeben wollen: Bitte melden Sie sich lautlos, und wir holen die Prüfung ab. Vorzeitige Abgaben sind nur bis 15 Minuten vor Prüfungsende möglich.
8. Wenn Sie zur Toilette müssen, melden Sie dies einer Aufsichtsperson durch Handzeichen.
9. Wir beantworten keine inhaltlichen Fragen während der Prüfung. Kommentare zur Aufgabe schreiben Sie bitte auf das Aufgabenblatt.

General guidelines:

1. *Exam duration: 60 minutes.*
2. *Permitted examination aids: dictionary (for languages spoken by yourself). 4 A4 pages hand written or $\geq$ 11pt font size.*
3. *Use a pen (black or blue), not a pencil. Please write legibly. We will only consider solutions that we can read.*
4. *Solutions must be written directly onto the exam sheets in the provided boxes (and directly below, if more space is needed). Invalid solutions need to be crossed out clearly. Provide corrections to answers of multiple choice questions without any ambiguity!*
5. *There are no negative points for wrong answers.*
6. *If you feel disturbed by anyone or anything, let the supervisor of the exam know immediately.*
7. *We collect the exams at the end. Important: You must ensure that your exam has been collected by an assistant. Do not take any exam with you and do not leave your exam behind on your desk. The same applies when you want to finish early: Please contact us silently and we will collect the exam. Handing in your exam ahead of time is only possible until 15 minutes before the exam ends.*
8. *If you need to go to the toilet, raise your hand and wait for a supervisor.*
9. *We will not answer any content-related questions during the exam. Please write comments referring to the tasks on the exam sheets.*

Question:	1	2	3	4	5	Total
Points:	22	10	6	10	12	60
Score:						

Generelle Anmerkung / *General Remark*

Verwenden Sie die Notation, Algorithmen und Datenstrukturen aus der Vorlesung. Falls Sie andere Methoden verwenden, müssen Sie diese kurz so erklären, dass Ihre Ergebnisse nachvollziehbar sind.

Use notation, algorithms and data structures from the course. If you use different methods, you need to explain them such that your results are comprehensible.

Aufgabe 1: Verschiedenes (22P)

1) In dieser Aufgabe sollen nur Ergebnisse angegeben werden. Begründungen sind nicht notwendig.

   In this task only results have to be provided. Explanations are not required.

2) Als Ordnung verwenden wir für Buchstaben die alphabetische Reihenfolge, für Zahlen die aufsteigende Anordnung gemäss ihrer Grösse.

   As the order of characters we take the alphabetical order and for numbers we take the ascending order according to their sizes.

/4P (a) Teilen Sie folgende Algorithmen zu asymptotischen Laufzeiten in Abhängigkeit von der Problemgrösse n zu. Alle Laufzeiten sind für den **schlechtesten Fall**.

Assign the following algorithms to asymptotic running times as a function of the problem size n. All runtimes are for the **worst case**.

A Heap:ExtractMin
B HeapSort
C QuickSort
D MergeSort
E Sortieren durch Auswahl / *Selection Sort*
F Lineare Suche im unsortieren Array / *Linear search in unsorted array*
G Binäre Suche im sortierten Array / *Binary search in sorted array*
H Medianauswahl mit Quickselect / *Median selection with Quickselect*

$\Theta(1)$	
$\Theta(\log n)$	A, G
$\Theta(n)$	F
$\Theta(n \log n)$	B,D
$\Theta(n^2)$	C,E,H

(b) Im folgenden ist eine Hashtabelle dargestellt (nachdem die Schlüssel 7, 9, 10 und 20 bereits eingefügt wurden). Die Hashfunktion ist $h(k) = k \bmod 13$. Kollisionen werden mit linearem Sondieren aufgelöst. Die Sondierung läuft immer nach rechts. Fügen Sie die folgenden Schlüssel in die (teilweise schon belegte) Hashtabelle in. Wie viele Kollisionen treten beim Einfügen der folgenden drei Zahlen auf?

In the following a hash table is displayed (after keys 7, 9, 10 and 20 had been inserted previously). The hash function is $h(k) = k \bmod 13$. Collisions are resolved using linear probing. Probing always goes left. Insert the following keys into the (partially occupied) hash table. How many collisions take place during the insertion of the three keys?

/4P

Einzufügende Schlüssel/*Keys to be inserted*: 19,6,24

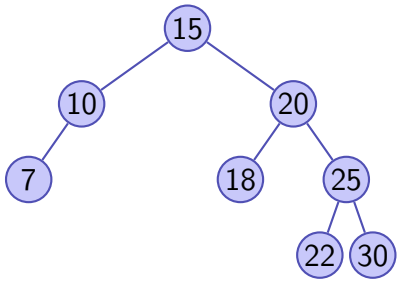
						19	7	20	9	10	**6**	**24**
0	1	2	3	4	5	6	7	8	9	10	11	12

Anzahl Kollisionen/*Number of collisions*: ◯ 4 ◯ 5 √ 6 ◯ 7 ◯ 8

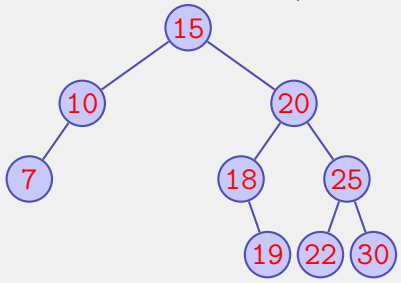
(c) Fügen Sie in untenstehendem binären Suchbaum zuerst den Schlüssel 19 ein und löschen Sie danach im entstandenen Suchbaum den Schlüssel 20.

First insert into the Binary Search Tree below the key 19. After insertion, delete the key 20 from the created binary search tree.

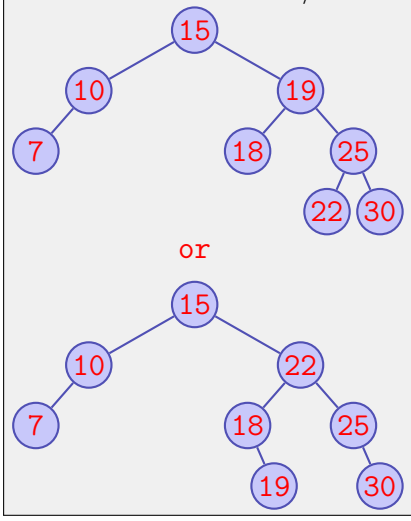
/4P



Nach Einfügen von 19 / *After insertion of 19*

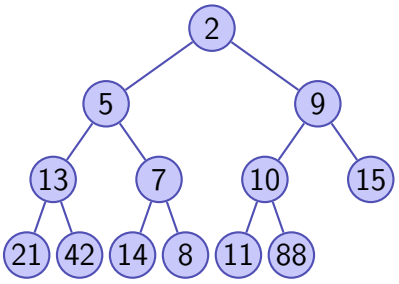


Nach Löschen von 20 / *After deletion of 20*

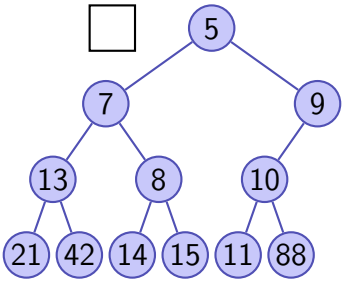


/2P (d) Führen Sie auf folgendem Min-Heap eine Extract-Min Operation aus, wie in der Vorlesung vorgestellt, einschliesslich der Wiederherstellung der Heap-Bedingung. Wie sieht der Heap nach der Operation aus? Kreuzen Sie die richtige Antwort an.

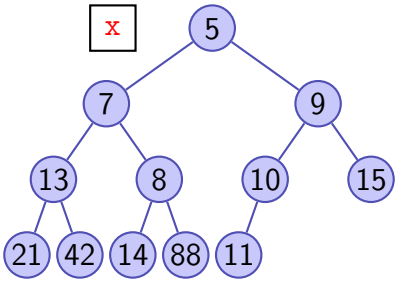
On the following Min-Heap, perform an extract-min operation, including re-establishing the heap-condition, as shown in class. What does the heap look like after the operation? Mark the correct answer.



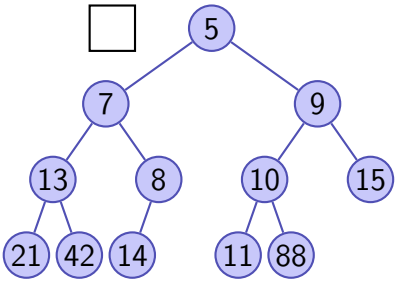
☐



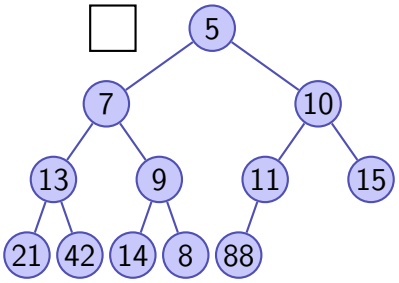
[x]



☐



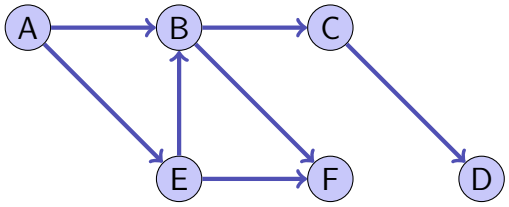
☐



(e) Auf wie viele Arten kann der folgende gerichtete Graph topologisch sortiert werden? Tipp: sortieren Sie den Graph topologisch nach dem Algorithmus der Vorlesung und überlegen Sie sich in jedem Schritt, welche Möglichkeiten Sie haben.

In how many ways can he following directed graph be sorted topologically? Hint: sort the graph topologically following the algorithm of the course and think at each step, what kind of possibilities you have.

/2P



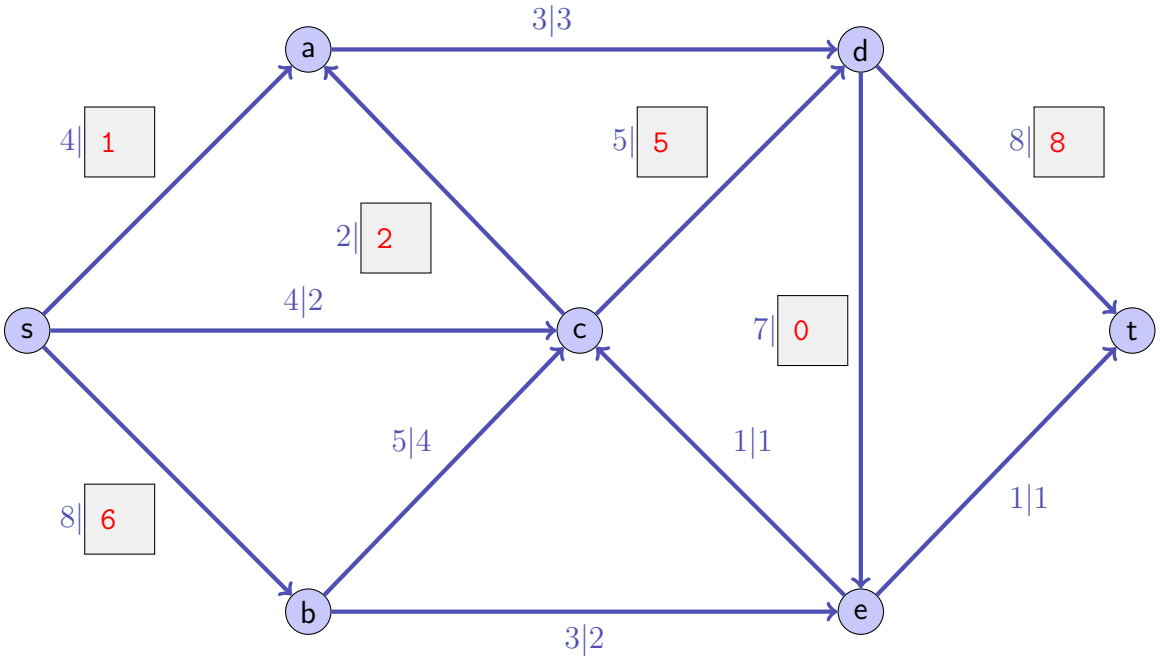
Anzahl Topologische Sortierungen/*Number Topological sortings*:

3

(f) Gegeben ist das folgende Flussnetzwerk mit Quelle s und Senke t. Die einzelnen Kapazitäten c_i und Flüsse ϕ_i sind an den Kanten angegeben als $c_i|\phi_i$. Ergänzen Sie die fehlenden Flusswerte auf den Kanten, so dass ϕ ein gültiger Fluss ist. Dieser wird (in diesem Beispiel) in jedem Falle maximal sein. Zeichnen Sie in der Abbildung einen Schnitt ein, der zeigt, dass ϕ maximal ist.

Provided in the following is a flow network with source s and sink t. Capacities c_i and flows ϕ_i are provided at the edges as $c_i|\phi_i$. Complete the missing flow values at the edges such that the overal flow ϕ is a valid flow. This will be maximal (in this example) in either case. Draw into the figure a cut that shows that ϕ is indeed maximal.

/3P



/3P (g) Geben Sie die Ausgabe der folgenden Anweisung in der Box unten an. *Provide the output of the following statement in the box below.*

```
Arrays.asList(10, 3, 8, 9, 2, 5)
    .stream()
    .map(i -> i*i % 10)
    .sorted()
    .filter(i -> {return i < 6;})
    .forEach(System.out::print);
```

0 1 4 4 5

Aufgabe 2: Asymptotik (10P)

/3P (a) Finden Sie aus der Liste der in der weissen Box angebenen Funktionen jeweils diejenige, so dass die Gleichheit gilt. *Find from the white boxed list of provided functions for each case below the function such that the equality holds.*

Beispiel/*Example*: $\Theta(n+5) = \Theta(\boxed{n})$

$1, \quad \log n, \quad \log^2 n, \quad n, \quad n\log n, \quad n^2, \quad n^4, \quad n^2\log n, \quad n\log^2 n, \quad n!, \quad n^n$

$\Theta(\Sigma_{i=0}^{10n} \log(n^n)) = \Theta(\boxed{n^2 \log n})$

$\Theta(\Sigma_{i=0}^{n^2} i) = \Theta(\boxed{n^4})$

$\Theta(3n + 5n\log n + 0.001n^2) = \Theta(\boxed{n^2})$

In den folgenden Aufgabenteilen wird jeweils angenommen, dass die Funktion g mit $g(n)$ aufgerufen wird. Geben Sie jeweils die asymptotische Anzahl von Aufrufen der Funktion $f()$ in Abhängigkeit von $n \in \mathbb{N}$ mit Θ-Notation möglichst knapp an. Die Funktion f ruft sich nicht selbst auf. Sie müssen Ihre Antworten nicht begründen.

In the following parts of this task we assume that the function g is called as $g(n)$. Provide the asymptotic number of calls of $f()$ depending on $n \in \mathbb{N}$ using Θ notation as tight as possible. The function f does not call itself. You do not have to justify your answers.

(b) /2P

```
void g(int n){
    for (int i = 0; i<100;++i)
        f();
    }
}
```

Anzahl Aufrufe von f / *Number of calls of f*

$\Theta(1)$

(c) /2P

```
void g(int n){
    if (n<100){
        g(n+1);
    }
    f();
}
```

Anzahl Aufrufe von f / *Number of calls of f*

$\Theta(1)$

(d) /3P

```
void g(int n){
    if (n > 1){
        g(n-1)
        g(n-1);
    }
    else{
        f();
    }
}
```

Anzahl Aufrufe von f / *Number of calls of f*

$\Theta(2^n)$

/6P

Aufgabe 3: Code Lesen & Schreiben (6P)

Im Folgenden sehen Sie die Datenstruktur eines Suchbaumknotens.

In the following you see the data structure of binary search tree node.

```
// Data structure to store a Binary Search Tree node
class Node{
    int value;
    Node left = null;
    Node right = null;

    Node(int value) {this.value = value; }
}
```

Ein binärer Baum mit Wurzelknoten `root` wurde aus solchen Knoten zusammengesetzt und Sie wollen wissen, ob er auch ein korrekter Suchbaum ist. Vervollständigen Sie folgenden Code, so dass ein Aufruf an `isBst(root)` zurückgibt, ob der Baum die Suchbaumeigenschaft hat.

A binary tree with root node `root` has previously been componsed of such nodes and you want to know, if it is indeed a correct binary search tree. Complete the follwing code such that a call to `isBst(root)` returns if the tree has the search tree property.

```
// Function to determine if given Binary Tree is a BST or not
public static boolean isBst(Node root){
  return isBstR(root, Integer.MIN_VALUE, Integer.MAX_VALUE);
}
// recursive function called by isBst
public static boolean isBstR(Node node, int min, int max){
  if (node == null) {
     return true;
  }
  if (node.value < min || node.value > max) {
     return false;
  }
  return isBstR( node.left, min, node.value ) &&
         isBstR( node.right, node.value, max );
}
```

Aufgabe 4: Datenbanken (10P)

Angenommen eine Datenbank zur Verwaltung von Wasserständen in Seen enthält die folgenden Tabellen zu Gebieten, Seen, Regenmengen und Wasserständen. Wir gehen davon aus, dass ein See nur in einem Gebiet liegen kann.

Consider a data base for managing water height data in lakes containing the following tables concerning rivers, sensor locations, areas and rainfall data.

area

id	name
1	Tal
2	Berg
3	Moor
4	Heide
5	Betonwüste

lake

id	name
1	Tiefensee
2	Hochtümpel
3	Talsee
4	Klarwasser
5	Trüblache

located_in

lake_id	area_id
1	1
2	2
3	1
4	4
5	3

time

id	date
1	1.1.2018
2	1.2.2018
3	25.2.2018
4	30.4.2018
5	1.5.2018
6	10.5.2018

water_level

time_id	lake_id	level
1	1	+0.2
1	3	+0.4
2	1	+0.1
2	2	+0.2
2	4	-0.2
3	5	-0.1
4	2	+0.2
5	4	+0.1
5	6	+0.8

rain

time_id	area_id	amount
1	1	10
1	2	12
2	4	15
3	1	2
4	2	10
5	1	0
5	2	0
5	3	0
5	5	0

(a) Schreiben Sie zu den folgenden Anfragen jeweils in Ihren Worten, was gesucht ist.

To the following queries write down in your own words what is looked for.

/6P

```
SELECT DISTINCT name
FROM area, rain
WHERE id = area_id AND amount > 9
```

Name aller Bereiche, in denen es irgendwann stark geregnet hat.

```
SELECT l.name
FROM area a, lake l, located_in
WHERE lake_id = l.id AND area_id = a.id AND a.name = "Tal"
```

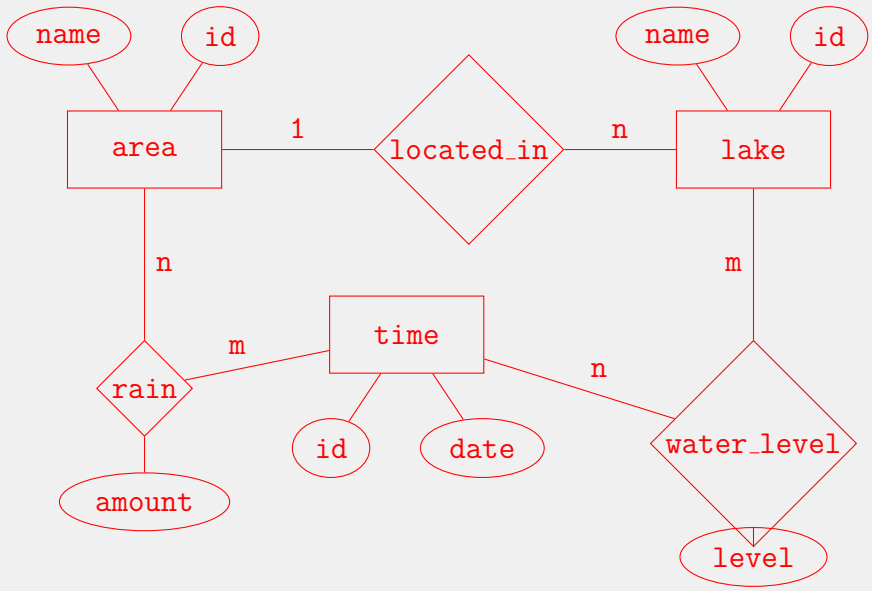
Name aller Seen, welche im Tal liegen.

```
SELECT DISTINCT date
FROM time JOIN waterlevel ON time.id = waterlevel.time_id
WHERE level < 0
```

Daten, an denen mindestens ein See Niedrigwasser hatte.

/3P (b) Zeichnen Sie ein Entity-Relationship-Diagramm zum obigen relationalen Modell. Geben Sie die Funktionalitäten (Kardinalitäten) an.

Draw an Entity-Relationship-diagram to the relational model from above. Provide the cardinalities.

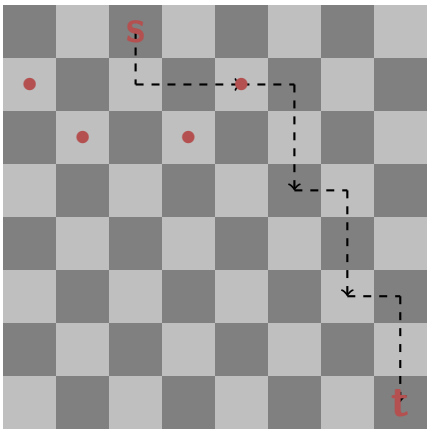


/1P (c) Geben Sie im folgenden an, ob sie die Tabellen hätten weiter zusammenfassen können. Begründen Sie Ihre Entscheidungl.

Provide in the following if the tables could have been merged any further. Give an explanation of your decision.

Die Tabelle Lake hätte als 1:n Tabelle mit der Area Tabelle vereint werden können: id / name / area_id

Aufgabe 5: Kürzeste Wege (12P)



Wir betrachten ein $n \times n$ Schachbrett ($n > 2$ gerade). Ein Pferd kann jeweils eine bestimmte Bewegung (einen „Sprung") ausführen: 1 Feld in eine Richtung, gefolgt von 2 Feldern orthogonal dazu.
Es geht nun darum, von einem Startfeld s aus eine Sequenz von möglichst wenigen Sprüngen zu finden, so dass das Pferd beim Zielfeld t landet.

We consider an $n \times n$ chess board ($n > 2$ even). The knight can make a certain movement (a "jump"): one square in one direction followed by two squares in an orthogonal direction.
Goal is now to find a sequence of jumps, as few as possible, to move from a given start square s to a target square t.

(a) Beschreiben Sie einen möglichst effizienten Algorithmus zum Finden des kürzesten Weges im Detail. Welche Datenstrukturen setzen Sie ein?

Describe an algorithm, as efficient as possible, to find the shortest path in detail. What kind of data structures do you employ?

/6P

Breitensuche [1]: wir führen ein ganzzahliges $n \times n$ **Array** x[1] für die Felder und initialisieren alle Felder mit -1 (für ,,nicht besucht‘‘).
Wir legen den Startindex in eine **Warteschlange (Queue)**[1] und setzen $s[x] = 0$.
Für alle Indizes i in der Queue durchlaufen wir jeweils alle Nachbarn[1] (im Sinne des Pferdsprungs) n. Sofern $s[n] = -1$ setzen wir $s[n] \leftarrow s[i] + 1$ [1] und fügen n der Queue hinzu[1]. So verfahren wir, bis $n = t$ gefunden (oder kein n mit $x[n] = -1$ mehr). [1]

/2P (b) Was ist die asymptotische Laufzeit Ihres Algorithmus in Abhängigkeit von der Seitenlänge n des Schachbrettes?

What is the asymptotic running time of your algorithm as a function of the side length n of the chess board?

Breitensuche $O((|E| + |V|)) = O(|V|) = O(n^2)$, denn die Anzahl der Nachbarn jedes Feldes ist beschränkt.

/2P (c) Uns wird das Spiel langweilig und wir erfinden eine neue Figur – den Zaubererer. Der Zauberer kann eine beschränkte Menge verschiedener Bewegungen (jeweils mit endlicher Reichweite) ausführen. Zum Beispiel so wie das Pferd springen, aber auch einzige Felder laufen. Allerdings haben die verschiedenen Bewegungen auch verschiedene Kosten. Zum Beispiel kostet der Pferdsprung 3 und das Gehen um ein Feld hat Kosten 2. Wir suchen nun den günstigsten Weg von s nach t im Sinne dieser Kosten. Welchen Algorithmus verwenden Sie nun?

Bored by the previous question, we invent a new chess piece – the magician. The magician can make a limited set of different movements (with finite distance, each). For example, it can jump like the knight, but also walk single squares. But now the different movemenents also incur different costs. For example, the jump of a knight costs 3 while moving by one square costs 2.
We now search for the cheapest path from s to t in terms of such costs. What kind of algorithm do you employ now?

We have a graph with different, but positive, weights now and are looking for a shortest path. A typical application of the Dijkstra algorithm.

/2P (d) Was ist die asymptotische Laufzeit des Algorithmus von (c) in Abhängigkeit von n?

What is the asymptotic running time of the algorithm of (c) as a function of n?

Laufzeit Dijkstra $O((|E| + |V|) \log |V|) = O((|V|) \log |V|) = O(n^2 \log n)$