Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Prüfung

Informatik II (D-BAUG)

Felix Friedrich, Hermann Lehner, Departement Informatik

ETH Zürich, 27.1.2020.

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:	19	10	8	12	11	60
Score:						

Generelle Anmerkung / *General Remark*

Verwenden Sie die Notation, Algorithmen und Datenstrukturen aus der Vorlesung. Falls Sie eine andere Herangehensweise wählen, erklären Sie Ihre Antworten in nachvollziehbarer Weise!

Use notation, algorithms and data structures from the course. If you use a different approach, explain your answers in a comprehensible way!

Aufgabe 1: Verschiedenes (19P)

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.

/6P (a) Kreuzen Sie an, ob die folgenden Aussagen wahr oder falsch sind.

Mark if the following statements are true or false.

Eine Inorder-Traversierung eines binären Suchbaumes erzeugt eine sortierte Liste der gespeicherten Schlüssel. / *An in-order traversal of a binary search tree generates a sorted list of the stored keys.*

◯Wahr / *True*

◯Falsch / *False*

Hat eine Folge von m Operationen im schlimmsten Fall Gesamtkosten $O(m)$, dann hat jede einzelne der Operationen im schlimmsten Fall Kosten $O(1)$. / *If a sequence of m operations has overal worst case costs of $O(m)$, then every single of the operations has worst cases costs of $O(1)$*

◯Wahr / *True*

◯Falsch / *False*

Jeder vergleichbasierte Sortieralgorithmus kann zu einem stabilen Sortieralgorithmus gemacht werden mit asymptotisch, bis auf einen konstanten Faktor, unveränderter Laufzeit. / *Any comparison based sorting algorithm can be made to be stable, without affecting the asymptotic running time by more than a constant factor.*

◯Wahr / *True*

◯Falsch / *False*

Sei $G = (V, E)$ ein Graph. Jeder Teilgraph von G mit $|V|-1$ Kanten ist ein Spannbaum von G. / *Let $G = (V, E)$ be a graph. Every subgraph of G with $|V| - 1$ edges is a spanning tree ot G.*

◯Wahr / *True*

◯Falsch / *False*

In einem Max-Heap mit n Schlüsseln ist die Laufzeit zur Extraktion des Minimums im schlechtesten Fall $O(\log n)$. / *In a Max-Heap with n keys the worst-case running time to extract the minimum is $O(\log n)$.*

◯Wahr / *True*

◯Falsch / *False*

In einem AVL-Baum dürfen sich die Anzahlen der Knoten im linken und im rechten Teilbaum maximal um 1 unterscheiden. / *In an AVL-Tree the number of nodes in the left and right subtree must not differ by more than 1.*

◯Wahr / *True*

◯Falsch / *False*

(b) Führen Sie auf dem folgenden Array einen Aufteilungsschritt des Sortieralgorithmus Quicksort durch. Benutzen Sie als Pivot das Element 5.

On the following array, perform a partitioning step of the sorting algorithm Quicksort. As pivot, use the element 5.

/2P

3	8	10	11	9	5	7	8	2	6	4
0	1	2	3	4	5	6	7	8	9	10

0	1	2	3	4	5	6	7	8	9	10

(c) Gegeben sei die folgende Schlüsselmenge:

Let the following set of keys be given:

/2P

$$K = \{5, 9, 11, 15, 7, 20\}$$

Zeichnen Sie die beiden binären Suchbäume, die genau die Schlüssel aus K verwalten und die unter allen möglichen Suchbäumen minimale bzw. maximale Höhe haben.

Draw the two binary search trees that contain the keys from K and that provide minimal / maximal height.

Minimal:

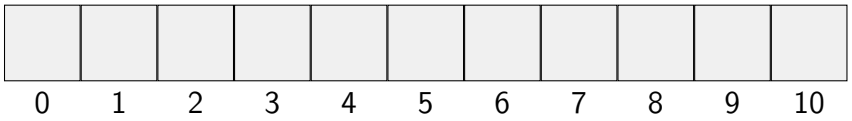
Maximal:

/2P (d) Fügen Sie die folgenden Schlüssel (in der angegebenen Reihenfolge) in die Hashtabelle ein. Verwenden Sie offene Addressierung und lineares Sondieren. Die verwendete Hash-Funktion $h(k)$ ist nachfolgend angegeben (es wird addiert, also nach rechts sondiert).

Enter the following keys (in the order provided) into the hash-table. Use open addressing and linear probing. The used hash function $h(k)$ is provided below (values are added, i.e. probing runs to the right).

Hashfunktion / *hash function* : $h(k) = k \bmod 11$

Schlüssel / *keys* : 2,13,24,40



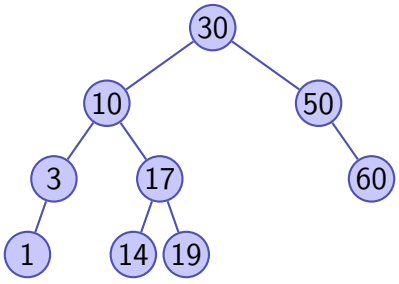
Wie viele Kollisionen treten bei der nachfolgenden (erfolglosen) Suche nach dem Schlüssel 100 auf? (Gezählt werden nur Kollisionen mit belegten Plätzen)

How many collisions occur when key 100 is searched now (unsuccessfully)? (We only count collisions with occupied spaces)

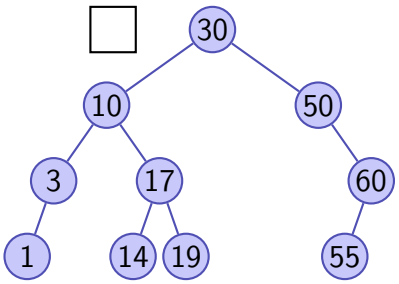
(e) Fügen Sie in folgendem AVL Baum den Schlüssel 55 ein und rebalancieren Sie. Wie sieht der AVL Baum nach dem in der Vorlesung gezeigten Algorithmus aus? Kreuzen Sie die richtige Antwort an.

In the following AVL tree, insert key 55 and rebalance. What does the AVL tree look like according to the algorithms that has been shown in class? Mark the correct answer.

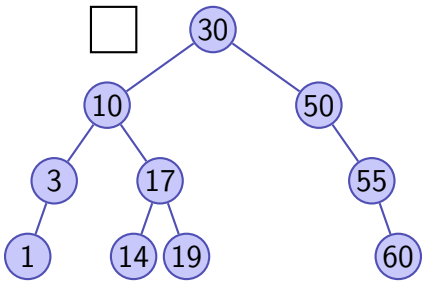
/2P



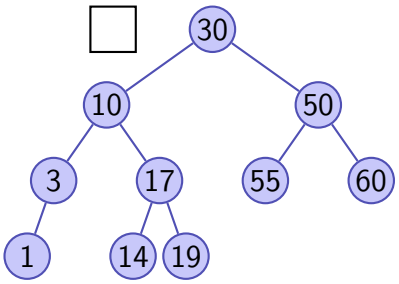
☐



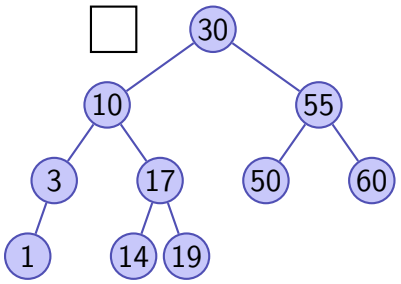
☐



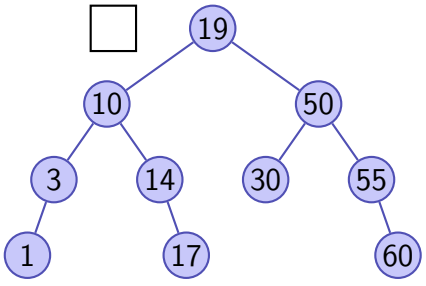
☐



☐

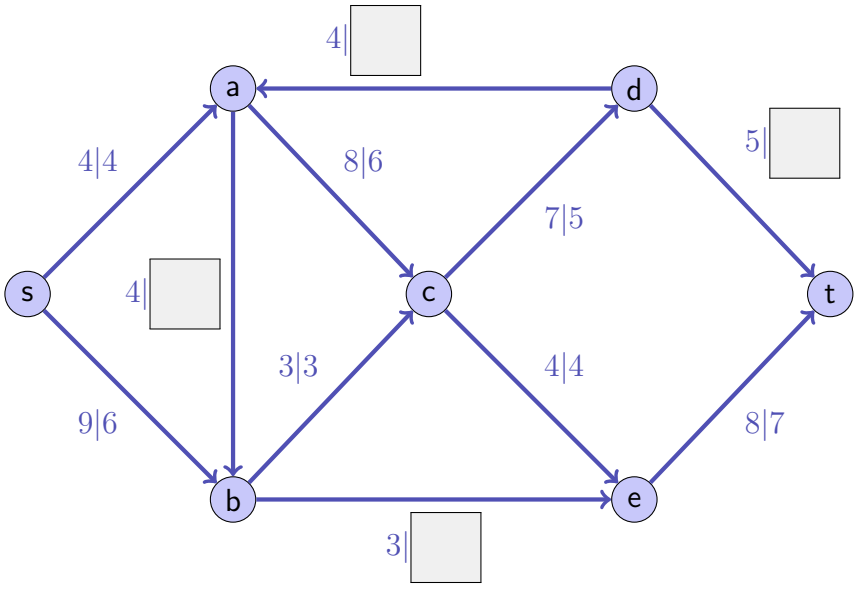


☐



/5P (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. Geben Sie den Wert des Flusses f an. 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 all cases. Provide the value of the flow f. Draw into the figure a cut that shows that ϕ is indeed maximal.



$f =$

Aufgabe 2: Asymptotik (10P)

/3P (a) Geben Sie für die untenstehenden Funktionen eine Reihenfolge an, so dass folgendes gilt: Wenn eine Funktion f links von einer Funktion g steht, dann gilt $f \in \mathcal{O}(g)$.
Beispiel: die drei Funktionen n^3, n^5 und n^7 sind bereits in der richtigen Reihenfolge, da $n^3 \in \mathcal{O}(n^5)$ und $n^5 \in \mathcal{O}(n^7)$.

Provide an order for the functions below such that the following holds: If a function f is left of a function g then it holds that $f \in \mathcal{O}(g)$. Example: the functions n^3, n^5 and n^7 are already in the correct order because $n^3 \in \mathcal{O}(n^5)$ and $n^5 \in \mathcal{O}(n^7)$.

$$\frac{n^2}{\log n}, \quad n\sum_{i=0}^{n} i, \quad n\sum_{i=0}^{\log n} 2^i, \quad n\log n^n, \quad n\sqrt{n}, \quad n\log n^2,$$

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 $g(n)$. Fill in the asymptotic number of calls of $f()$ depending on $n \in \mathbb{N}$ using Θ notation as succinct as possible. The function f does not call itself. You do not have to justify your answers.

(b) /2P

```
void g(int n){
    for (double i = 1; i < n; i*=3){
        f();
        i /= 2;
    }
}
```

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

(c) /2P

```
void g(int n){
    if (n > 0){
        g(n-1);
    }
    f();
}
```

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

(d) /3P

```
void g(int n){
    if (n>0){
        f();
        g(n/2);
        f();
        g(n/2);
    } else {
        f();
    }
}
```

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

Aufgabe 3: Python (8P)

/8P (a) Schreiben Sie eine Funktion, die zu einer gegebenen Liste von Worten ein Dictionary zurückgibt, welches die Zuordnung der Worte zu ihren Häufigkeiten enthält.
Anwendungsbeispiel:

Write a function that for a given list of words returns a dictionary containing the mapping from words to frequency of the words.
Application example:

Eingabe / *Input*

```
words = ["a","cat","is","a","cat"]
count = word_count(words)
for name in count:
    print name,":",count[name]
```

Ausgabe / *Output*

```
a : 2
is : 1
cat : 2
```

```
def word_count(word_list):
```

Aufgabe 4: Rekursion / Dynamische Programmierung (12P)

Es sei ein Stab der Länge $n \in \mathbb{N}$ gegeben. Ausserdem sei ein Array p von $k \in \mathbb{N}$ verschiedenen Längen $p_i \in \mathbb{N}$ $(0 \leq i < k)$ gegeben. Ziel ist, den Stab in möglichst viele (N) Stücke zu zersägen. Dabei muss allerdings jedes Stück eine der gegebenen Längen aus p aufweisen. Es darf kein Stück übrig bleiben. Beispiel:

Consider a rod (stick) of length $n \in \mathbb{N}$. Let an array of $k \in \mathbb{N}$ different lengths $p_i \in \mathbb{N}, 0 \leq i < k$ be given. Goal is to cut the rod into as many pieces (N) as possible. Each piece length must be of one of the given lengths of p. No piece may be left over
Example:

$$n = 5, k = (2, 3, 5) \Rightarrow N = 2 \quad (5 = 2 + 3),$$
$$n = 7, k = (2, 3, 5) \Rightarrow N = 3 \quad (7 = 2 + 2 + 3)$$

(a) Aufgabe: Vervollständigen Sie die folgende rekursive Funktion so, dass sie die maximale Anzahl Stücke zurückgibt. Wenn der Stab nicht ohne Rest zerteilbar ist, muss die Funktion -1 zurückgeben.

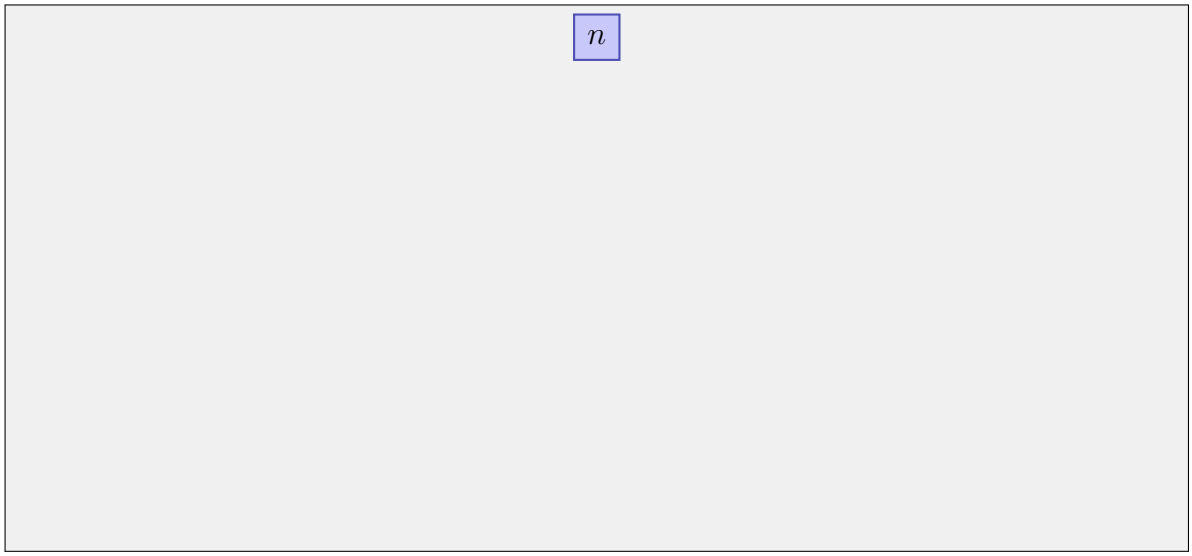
Task: complement the following recursive function such that it returns the maximum number of pieces. If the stick cannot be divided without rest, the function must return 0.

/2P

```
// return the maximum number of pieces to cut n with pieces of lengths from p
// if no possibility to cut without rest, -1 is returned
public static int solve(int n, int[] p){
  if (n<0){
    return -1;
  } else if (n == 0){
    return 0;
  } else {
    int max = -1;
    for (int i = 0; i < p.length; ++i){
      max = Math.max(max,                    );
    }
    if (max == -1) {
      return -1;
    }
    return                                   ;
  }
}
```

/3P (b) Sie stellen fest, dass Ihre Funktion für grosse n sehr lange benötigt. Skizzieren Sie den Rekursionsbaum (einen Teil davon, etwa drei Ebenen) für $k = 3$. Geben Sie die asymptotische Laufzeit des rekursiven Algorithmus in Abhängigkeit von n und k an.

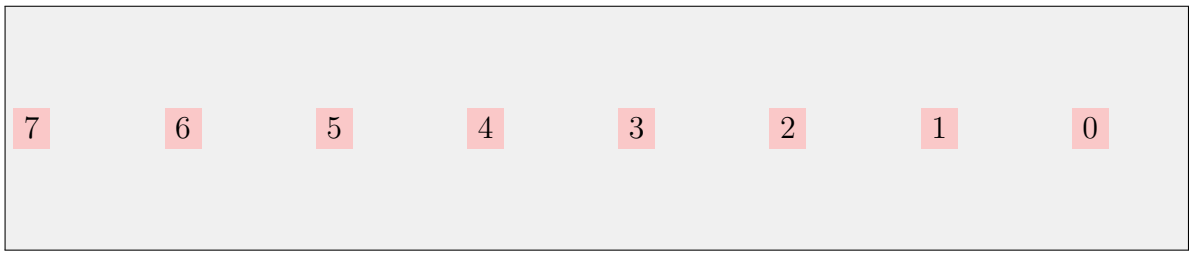
You realize that your function takes a lot of time for large n. Sketch the recursion tree (a part of it, about 3 levels) for $k = 3$. Provide the asymptotic running time of the recursive algorithm as a function of n and k.



Asymptotische Laufzeit / *Asymptotic running time*:

/2P (c) Überlegen Sie sich die Abhängigkeiten Ihres Problems. Zeichnen Sie in folgender kleinen Skizze (für $n = 7$, $p = (2, 3)$), die Abhängigkeiten ein. Sie zeigen damit dass es keine zirkulären Abhängigkeiten gibt.

Now think about the dependencies of your problem. Draw into the following small sketch (for $n = 7$, $p = (2, 3)$) the dependencies. Doing so, you show that there are no circular dependencies.



(d) Vervollständigen Sie nun die folgende (nicht-rekursive) Funktion so, dass sie das gegebene Problem effizient löst.
Tipp: verwenden Sie die Abhängigkeiten, die Sie sich soeben überlegt haben für die Berechnung der DP-Tabelle.

Now complement the following (non-recursive) function such that the given problem is solved in an efficient way.
Hint: use the dependencies that you have just developed in order to compute the DP-table.

/3P

```
// return the maximum number of pieces to cut n with pieces of lengths from p
// if no possibility to cut without rest, -1 is returned
public static int solve(int n, int[] p){
  int[] solution = new int[n+1];
  solution[0] = 0;
  for (int i = 1; i<=n; ++i){
    solution[i] = -1;
  }
  for (int i = 1; i<=n; ++i){
    for (int j = 0; j < p.length; ++j){
      if (i-p[j] >= 0){
        
      }
    }
    if (solution[i] != -1){
      
    }
  }
  return 
}
```

(e) Welche asymptotische Laufzeit hat Ihre Funktion nun in Abhängigkeit von k und n?

What is the asymptotic runtime of your function as a function of k and n?

/2P

Aufgabe 5: Algorithmen (11P)

/1P (a) Um einen kürzesten Pfade Algorithmus auf einem ungewichteten Graphen mit einer asymptotischen Laufzeit zu implementieren, welche linear in Anzahl Knoten und Anzahl Kanten ist, verwendet man im Algorithmus welche Datenstruktur?

In order to implement a shortest path algorithm on an unweighted graph with an asymptotic runtime that is linear in number of nodes and number of egdes, which data structure is used for the algorithm?

- ◯ Warteschlange / *Queue*
- ◯ Stapel / *Stack*
- ◯ Heap / *Heap*
- ◯ AVL Baum / *AVL Tree*

/1P (b) Was ist die asymptotische Laufzeit des Bellman-Ford-Algorithmus auf einem vollständigen Graphen mit n Knoten?

What is the asymptotic runtime of the Bellman-Ford algorithm on a complete graph with n nodes?

- ◯ $\Theta(n^2)$
- ◯ $\Theta(n \log n)$
- ◯ $\Theta(n^3)$
- ◯ $\Theta(n^4)$

/1P (c) Angenommen wir haben einen korrekt implementierten Kürzeste-Pfade Algorithmus auf einem Graphen mit ganzzahligen Kantengewichten. Wenn wir zu jeder Kante den Wert 1 hinzuaddieren ändern sich offensichtlich die Pfadlängen. Aber findet der Algorithmus stets dieselben kürzesten Pfade?

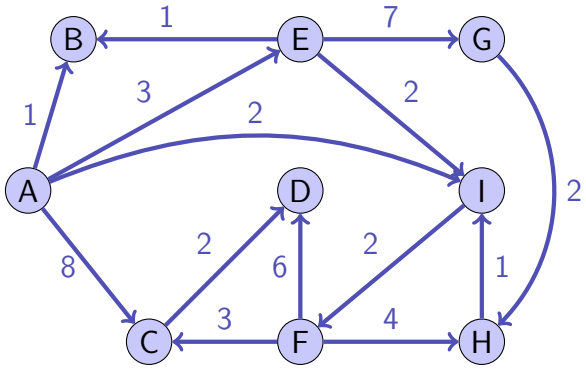
Assume you have a correctly implemented shortest-path algorithm on a graph with integer edge weights. If we add 1 to each edge, obviously all path lengths change. But does the algorithm always find the same shortest paths?

- ◯ Ja, bei allen Graphen / *Yes, with all graphs*
- ◯ Ja, bei allen positiv gewichteten Graphen / *Yes, with all positively weighted graphs*
- ◯ Nein / *No*

/2P

(d) Betrachten Sie folgenden gerichteten Graph. Geben Sie die Knoten in der Reihenfolge an, in der sie der Dijkstra-Algorithmus besuchen würde, wenn er vom Knoten A aus startet. Geben Sie zu jedem Knoten die kürzeste Pfadlänge von A aus an.

Consider the following directed graph. Provide the nodes in the order in which the Dijkstra algorithm visits them when it starts from node A. To each node provide the shortest path length from A.



Knoten / *Node*	Distanz / *Distance*
A	0

/3P (e) Betrachten Sie folgende Adjazenzmatrix, welche zu einem ungerichteten Graphen mit vier Knoten gehört. Skizzieren Sie zuerst den Graph. Was ist die grösste ganze Zahl für x, so dass es ein Paar von Knoten a und b gibt, so dass die zu x zugehörige Kante zwingend auf einem kürzesten Pfad von a nach b liegt.

Consider the following adjacency matrix that corresponds to an undirected graph with four nodes. First sketch the graph. What is the largest integer number for x such that there is a pair of nodes a and b such that a shortest path from a to b must include the edge that corresponds to x?

$$\begin{pmatrix} 0 & 1 & 7 & 4 \\ 1 & 0 & 4 & 7 \\ 7 & 4 & 0 & \mathbf{x} \\ 4 & 7 & \mathbf{x} & 0 \end{pmatrix}$$

Graph

$x =$

/3P (f) Sie haben einen sehr grossen Datensatz aus n unterschiedlichen Zahlen und wollen das k-kleinste Element finden ($k \ll n$). Wie machen Sie das effizient? Benennen Sie verwendete Datenstrukturen und Laufzeit des Algorithmus.

You have a huge data set with n different numbers and you want to find the k-smallest element ($k \ll n$). How do you do this efficiently? Provide the used data structures and the runtime of the algorithm.