Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Prüfung **(Lösung)**

Datenstrukturen und Algorithmen (D-MATH RW)

Felix Friedrich, Aritra Dhar, Departement Informatik

ETH Zürich, 21.8.2020.

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: 150 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: 150 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	6	7	Total
Points:	26	19	18	14	14	14	15	120
Score:								

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 (26P)

/8P (a) Kreuzen Sie an, ob die folgenden Aussagen wahr oder falsch sind. / *Mark if the following statements are true or false.*

Während die O-Notation die asymptotische Laufzeit eines Algorithmus im schlechtesten Fall charakterisiert, verwendet man die Θ-Notation, um gleichzeitig auch die Laufzeit im besten Fall zu charakterisieren. / *While the O-Notation is used to characterize the asymptotic runtime of an algorithm in the worst case, the Θ-Notation is used in order to characterize the best case runtime as well.*

◯Wahr / *True*
√Falsch / *False*

---

n vergleichbare Datenpunkte in einem binären Suchbaum zu speichern, kostet $\Theta(n)$ Speicherplatz. / *To store n comparable data points in a binary search tree, $\Theta(n)$ memory space is required.*

√Wahr / *True*
◯Falsch / *False*

Die Adjazenzmatrix eines Graphen mit n Knoten und m Kanten benötigt $\Theta(m)$ Speicherplatz. / *The adjacency matrix of a graph with n nodes and m edges requires $\Theta(m)$ memory.*

◯Wahr / *True*
√Falsch / *False*

Der Bellman-Ford Algorithmus auf einem Graphen mit n Knoten und m Kanten kann innerhalb einer Laufzeit von $O(m \cdot n)$ entscheiden, ob ein Graph negative Zyklen hat. / *The Bellman-Ford algorithm applied to a graph with n nodes and m edges, can determine the existence of negative cycles in a graph within an asymptotic running time of $O(n \cdot m)$*

√Wahr / *True*
◯Falsch / *False*

Die Angabe der Werte der Elemente eines binären Suchbaumes in Hauptreihenfolge bestimmt den dazugehörigen Baum eindeutig. / *The specification of the values of elements of a binary search tree in preorder uniquely determines the corresponding tree.*

√Wahr / *True*
◯Falsch / *False*

Wenn man in einem positiv gewichteten Graphen das Gewicht jeder Kante mit zwei multipliziert, bleibt jeder kürzester Weg zwischen zwei Knoten ein kürzester Weg. / *If in a positively weighted graph the weight of each edge is multiplied by two, every shortest path between two points remains a shortest path.*

√Wahr / *True*
◯Falsch / *False*

Die erlaubte Balance jedes Knotens eines AVL Baumes ist ein Wert zwischen -1 und 1. Daher darf sich die Höhe eines linken Teilbaumes von der Höhe des rechten Teilbaumes um 2 unterscheiden. / *The permitted balance of each node of an AVL tree is a value between -1 and 1. Therefore the height of a left subtree may differ from the height of the right subtree by 2.*

◯Wahr / *True*

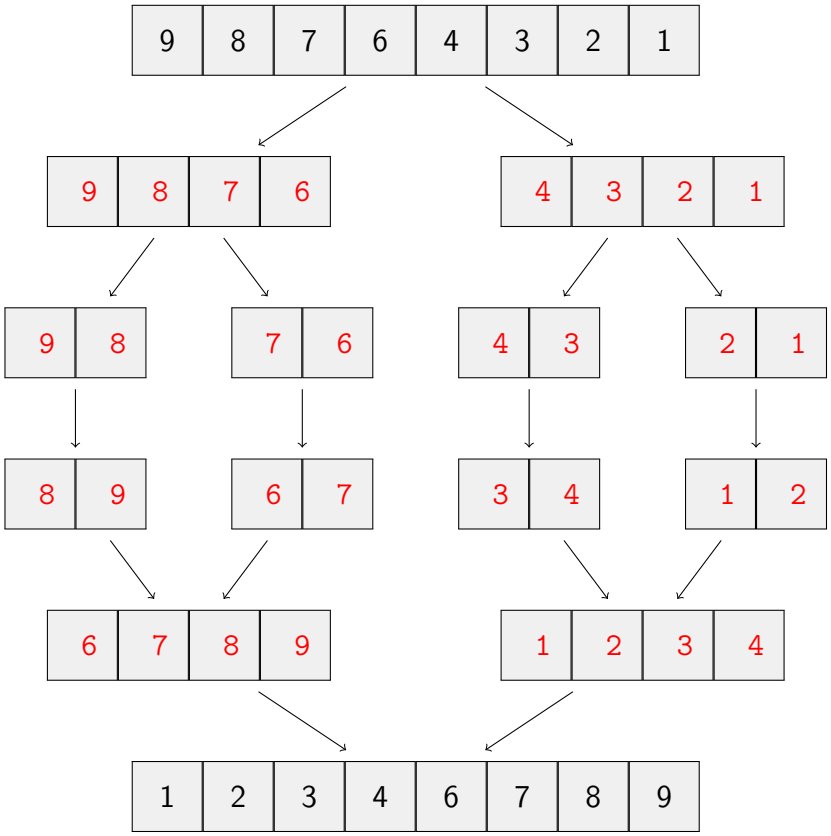
√Falsch / *False*

Die asymptotische Laufzeit, ein Element in einem binären balancierten Suchbaum mit $n \cdot 2^n$ Elementen zu finden ist $\Theta(n \log n)$ / *The worst case running time to search for an element in a balanced binary search tree with $n \cdot 2^n$ elements is $\Theta(n \log n)$*

◯Wahr / *True*

√Falsch / *False*

(b) Führen Sie auf dem folgenden Array den Mergesort-Algorithmus schematisch durch. *On the following array, perform the mergesort-algorithm using the scheme given.* /1P



/3P (c) Die Nebenreihenfolgeausgabe eines binären Suchbaumes ist

The post-order traversal output of a binary search tree is

$$1, 2, 4, 6, 5, 3$$

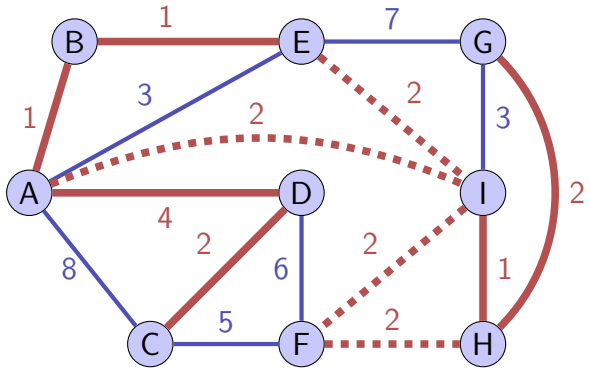
Finden Sie die Hauptreihenfolge.

Find the pre-order traversal.

Hauptreihenfolge / *pre-order traversal*: 3, 2, 1, 5, 4, 6

/3P (d) Markieren Sie in folgendem ungerichteten, gewichteten Graphen die Kanten, die zu jedem minimalen Spannbaum gehören. (Wenn Sie die Markierung von versehentlich markierte Kanten entfernen wollen, streichen Sie die Kanten durch). Wie viele mögliche minimale Spannbäume gibt es?

In the following undirected weighted graph, mark the edges that definitely belong to any minimum spanning tree. (If you want to remove the marks of marked edges again, you can cross the edges out). How many minimal spanning trees are possible here?



Anzahl Spannbäume / *Number Spanning Trees*: 4

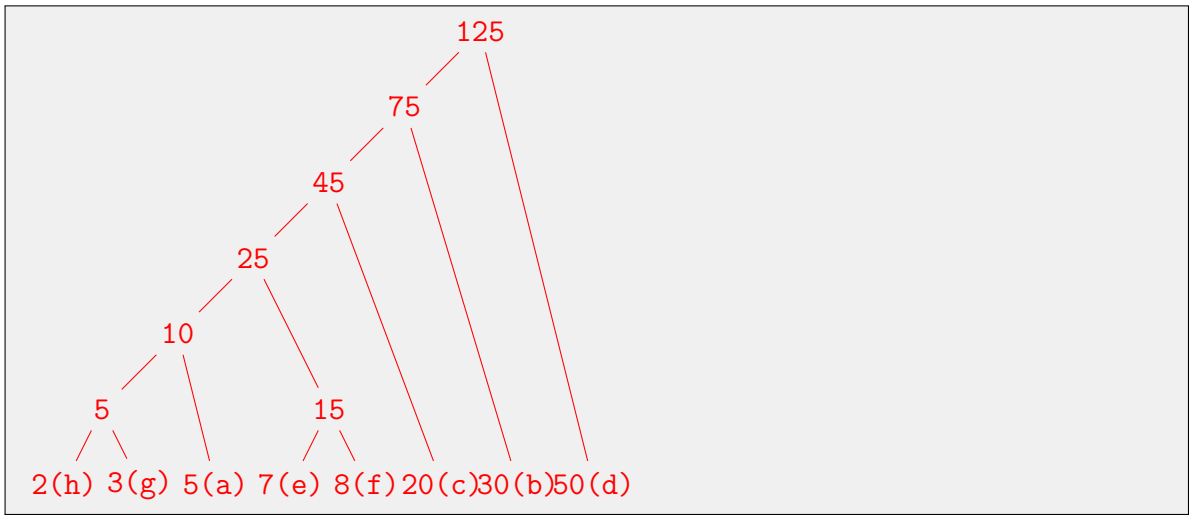
(e) Gegeben sind acht Buchstaben mit relativer Häufigkeit (Anzahl Zugriffe) wie folgt. Erstellen Sie mit Hilfe des Huffman-Algorithmus einen optimalen Codierungsbaum. Tragen Sie den resultierenden Code in der Tabelle ein.

Eight characters (keys) with relative frequency (number accesses) are given as follows. Using the Huffman algorithm provide an optimal code tree. Enter the corresponding code into the table.

/3P

char	a	b	c	d
freq	5	30	20	50
Code	00001	01	001	1

char	e	f	g	h
freq	7	8	3	2
code	00010	00011	000001	000000



(f) Was ist die asymptotische Laufzeit von Bubblesort im besten Fall für ein Array der Länge n? Für welches Array?

What is the asymptotic runtime of bubble sort in the best case for an array of size n? For which array?

/2P

(A) $\mathcal{O}(n^2)$
(B) $\mathcal{O}(n)$
(C) $\mathcal{O}n \log n$
(D) $\mathcal{O}(n \log^2 n)$

Option A, B, C oder / *or* D `(B)`
Szenario / *scenario*: `sorted data`

/2P (g) Eine Hashtabelle mit 10 Einträgen verwendet offene Adressierung mit der Hash-Funktion $h(k) = k \bmod 10$, mit linearer Sondierung (Sondierung geht nach rechts). Nachdem sechs Werte in die initial leere Hashtabelle eingefügt wurden, sieht die Hashtabelle wie folgt aus.

A hash table of length 10 uses open addressing with hash function $h(k) = k \bmod 10$, and linear probing (probing goes to the right). After inserting 6 values into an empty hash table, the table is as shown below.

0	1	2	3	4	5	6	7	8	9
		42	23	34	52	46	33		

Welche der folgenden Möglichkeiten bezeichnet eine Reihenfolge, in der die Schlüssel in die Hashtabelle eingefüllt werden konnten?

Which one of the following choices gives a possible order in which the key values could have been inserted in the table?

(A) 46, 42, 34, 52, 23, 33
(B) 34, 42, 23, 52, 33, 46
(C) 46, 34, 42, 23, 52, 33
(D) 42, 46, 33, 23, 34, 52

Option A,B,C oder / *or* D: `(C)`

/2P (h) Was ist die maximale Höhe eines AVL-Baumes mit 7 Knoten? Die Höhe eines Baumes mit einem einzigen Knoten sei 0.

What is the maximum height of any AVL-tree with 7 nodes? Assume that the height of a tree with a single node is 0.

(A) 2
(B) 3
(C) 4
(D) 5

Option A,B,C oder / *or* D: `(B)`

(i) Welche der folgenden Aussagen ist / sind wahr für ungerichtete Graphen?

Which of the following statements is/are TRUE for undirected graphs?

/2P

P: Die Anzahl Knoten ungeraden Grades ist gerade

P: Number of odd degree vertices is even.

Q: Die Summe der Grade aller Knoten ist gerade.

Q: Sum of degrees of all vertices is even.

(A) Nur P / *P only*
(B) Nur Q / *Q only*
(C) Beide: P und Q / *Both P and Q*
(D) Keine: weder P noch Q / *None of them*

Option A, B, C oder / *or* D: `(C): Both P and Q`

Aufgabe 2: Asymptotik (19P)

(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 entsprechenden Reihenfolge, da $n^3 \in \mathcal{O}(n^5)$ und $n^5 \in \mathcal{O}(n^7)$.

Provide an order for the following functions 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 respective order because $n^3 \in \mathcal{O}(n^5)$ and $n^5 \in \mathcal{O}(n^7)$.

/3P

$$\log_2^2 \sqrt{n} \;,\; 2^{\log_3^2 \sqrt{n}} \;,\; \sqrt{\log_2 n} \;,\; n! \;,\; \sum_{i=1}^{n} \mathcal{O}(n) \;,\; n^{1/3} \;,\; n^2 \log_2 n$$

$\sqrt{\log_2 n}$	$\log_2^2 \sqrt{n}$	$2^{\log_3^2 \sqrt{n}}$	$n^{1/3}$	$\sum_{i=1}^{n} \mathcal{O}(n)$	$n^2 \log_2 n$	$n!$

/6P (b) Gegeben sei die folgende Rekursionsgleichung:

Consider the following recursion equation:

$$T(n) = \begin{cases} 2T(\frac{n}{2}) + n\log_2 n, & n > 1 \\ 1 & n = 1 \end{cases}$$

Geben Sie eine geschlossene (nicht rekursive), einfache Formel für $T(n)$ an. Nehmen Sie an, dass es ein $m \in \mathbb{N}$ gibt mit $2^m = n$. Tipp: Schreiben Sie die Rekursionsformel zuerst in Abhängigkeit von m, lösen diese und transformieren dann zurück.

Specify a closed (non-recursive), simple formula for $T(n)$.
Hint: rewrite the recursion equation as a function of m, solve this recursion equation and then retransform.

$$\begin{aligned} T(2^m) &= 2T(2^{m-1}) + 2^m \log_2(2^m) \\ &= 2T(2^{m-1}) + m2^m \end{aligned}$$

Calling $f(m) = T(2^m)$ we get

$$\begin{aligned} f(m) &= 2f(m-1) + m2^m \\ &= 2(2f(m-2) + (m-1)2^{m-1}) + m2^m \\ &= 4f(m-2) + (m-1)2^{m-1} + m2^m \\ &= 8f(m-3) + (m-2)2^{m-2} + (m-1)2^{m-1} + m2^m \\ &= \dots \end{aligned}$$

$$\begin{aligned} f(m) &= 2^m f(0) + 2^m(1 + 2 + \dots + m) = 2^m f(0) + \frac{m(m+1)}{2}2^m \\ &= 2^m f(0) + m(m+1)2^{m-1} \end{aligned}$$

$$T(n) = nT(1) + n\frac{\log_2 n(1 + \log_2 n)}{2}$$

$$T(n) = n + n\frac{\log_2 n(1 + \log_2 n)}{2}$$

Geben Sie die asymptotische Laufzeit von $f(n)$ in Abhängigkeit von n möglichst knapp und präzise an.

Provide the asymptotic running time of $f(n)$ as a function of n as tight and precise as possible.

(c) /2P

```
void f(int n){
  if(n==1)
        return 1;
    else
        return f(n-1) + f(n-1)
  }
```

Asymptotische Laufzeit von f / *Asymptotic Running time of f*

$\Theta(2^n)$

(d) /2P

```
void f(int n){
    if(n==1)
     return 1;
    else {
        int x = f(n-1);
        return x + x;
    }
}
```

Asymptotische Laufzeit von f / *Asymptotic Running time of f*

$\Theta(n)$

/6P (e) Wie viele Einsen werden von der folgenden Funktion ausgegeben, wenn man sie mit n aufruft? Schreiben Sie die **Rekurrenz** hin, geben Sie eine **geschlossene einfache Formel** an und geben Sie die **asymptotische Anzahl in Θ Notation** an.

*How many ones does the following procedure print when run with input n? Write the corresponding **recurrence relation**. Provide a **closed, simple formula**. And provide the **asymptotic number in Θ notation**.*

```
void Ones(int n){
  if(n==0)
       printf("1");
    else {
       for(int i = 0; i < pow(2,n); i++) //2^n iterations
           Ones(n-1);
    }
}
```

$$T(n) = \begin{cases} 2^n T(n-1) & n > 0 \\ 1 & \text{n=0} \end{cases}$$

.

$$T(n) = \prod_{i=1}^{n} 2^i = 2^{\sum_{i=1}^{n} i} = 2^{\frac{n(n+1)}{2}}$$

.

Note that this quantity can be written as $\Theta(2^{n(n+1)/2})$ or $2^{\Theta(n^2)}$ but not as $\Theta(2^{n^2})$, since this last quantity is roughly the square of the correct answer and is not bounded by a constant multiple of it.

Aufgabe 3: Dynamic Programming: Palindromes (18P)

Für eine gegebene Zeichenkette s berechnen sie die minimale Anzahl Stücke in die s geschnitten werden kann, so dass jeder Teilstring ein Palindrom ist.
Beispiele:

- $BABABCBADCD$
  Minimale Anzahl Stücke 3 mit $BAB|ABCBA|DCD$
- $ABCBA$
  Minimale Anzahl Stücke 1 da $ABCBA$ bereits ein Palindrom ist.
- $ABCD$
  Minimale Anzahl Stücke 4 mit $A|B|C|D$.

Given a string s, find the minimum number of pieces into which s can be cut so that each piece is a palindrome.
For example.

- $BABABCBADCD$
  The minimum number pieces is 3 with $BAB|ABCBA|DCD$
- $ABCBA$
  The minimum number pieces is 1 as $ABCBA$ is already a palindrome.
- $ABCD$
  The minimum number pieces is 4 with $A|B|C|D$.

(a) Geben Sie zuerst die asymptotische Laufzeit an für die effiziente Berechnung, ob eine gegebene Zeichenkette der Länge n ein Palindrom ist. /1P

First provide the asymptotic running time to efficiently determine if a given string of length n is a palindrome.

$\Theta(n)$

Geben Sie nun einen Algorithmus an, der nach dem Prinzip der dynamischen Programmierung arbeitet und die minimale Anzahl Teilstrings berechnet.

Now provide a dynamic programming algorithm that computes the minimal number of substrings.

(b) Was ist die Bedeutung eines Tabelleneintrags, und welche Dimension und Grösse hat die DP-Tabelle T? /2P

What is the meaning of a table entry and what is the dimension and size of the DP-table T?

Tabellengrösse / *table size*:
either: $n \times n$ Table A (only entries in upper right triangle are used)
or: one dimensional table A with n entries.

Bedeutung Eintrag / *entry meaning*:
either: Entry at (i, j): Minimal number partitions for $s[i..j]$
or: Entry at i: Minimal number partitions for $s[i..n]$

/3P (c) Wie berechnet sich ein Eintrag der Tabelle aus den vorher berechneten Einträgen (Rekursionsgleichtung)?

How can an entry be computed from the values of previously computed entries (recursion equation)?

either:

$$A_{i,j} = \begin{cases} 1 & i = j \\ 1 & s[i..j] \text{ is palindrome} \\ A_{i,j} = \min_{i<k\leq j}\{A_{i,k-1} + A_{k,j}\}, & else \end{cases}$$

or:

$$A_i = \min\{1 + A_j \text{ for all } i < j \leq n \text{ with } s[i..j] \text{ is palindrome}\}$$

/2P (d) In welcher Reihenfolge können die Einträge berechnet werden?

In which order can the entries be computed?

```
either:
for s = 0 to n-1
for i = 1 to n-s
compute A[i,i+s]

or:
for i = n-1 down to 0
```

/1P (e) Wie kann der minimale Wert aus der Tabelle erhalten werden?

How can the minimal value be obtained from the DP table?

either: at $A[1,n]$, or: at A[0]

/2P (f) Geben Sie die Tabelle für folgenden Beispielstring an.

Provide the table for the following example expression

String : $ABBABBA$

1	2	2	1	2	2	1
	1	1	2	2	1	2
		1	2	1	2	2
			1	2	2	1
				1	1	2
					1	2
						1

$[1, 2, 2, 1, 2, 2, 1]$

(g) Zusätzlich zur minimalen Anzahl Stücke wollen Sie auch angeben, wie die Einteilung aussieht. Beschreiben Sie , wie Sie das bewerkstelligen.

In addition to the minimal number of pieces you want to provide the partitioning. Describe how this is done.

/2P

either:
First possibility: Keep a second matrix B where for each i, j we store the $\arg\min_{i<k\leq j}\{A_{i,k-1}\langle s_{k-1}\rangle A_{k,j}\}$. This matrix can be constructed together with the matrix of the minima. For $k = B_{i,j}$ we recursively walk back by considering entries $B_{i,k-1}$ and $B_{k,j}$.
Second possibility: We use the $\min$ identity from above and recursively walk back by checking the identities
or: Kepp a second array of length n with the successor for each partition.

(h) Geben Sie die asymptotichen Laufzeiten für die Berechnung der besten Unterteilung in Θ-Notation möglichst knapp mit Begründung an.

Provide the asymptotic running times for the computation of the minimal value and the corresponding partition in Θ notation as tight as possible with short explanation.

/5P

either:
Starting from the diagonal we need to compute $(n-1)+(n-1)\cdot 2+(n-2)\cdot 3+\cdots+(n-(n-1))\cdot(n-1)$ pairs of partitions, that is $\sum_{i=1}^{n-1} i\cdot(n-i) = n\cdot n\cdot(n-1)/2-\sum_{i=1}^{n-1} i^2 = n\cdot n\cdot(n-1)/2-(n-1)\cdot n\cdot(2n-1)/6 = \Theta(n^3)$
The walk back for the partition: With the first possibility (matrix B from above), the recursion equation reads $T(n) = T(n-q)+T(q)+1$ with the worst case being $q = 1$ yielding a $\Theta(n)$ runtime. If we need to search via matrix A each time, we have $T(n) = T(n-q)+T(q)+n$ yielding a $\Theta(n^2)$ runtime.
or: Compute A from right to left (n steps), for each entry we need $\sum_{j=i}^{n}\Theta(j-i)+1$ (compute if palindrome for each substring). Summed up we have $\Theta(n^3)$. To reconstruct the partitioning: $\Theta(n)$.
Both results can be improved by an order n by creating a boolean $n\times n$ table of palindromes first. This can be integrated into the first algorithm or done separately for the the first or second algorithm.
Use that: isPalindrome(i,j) = s[i]=s[k] and isPalindrome(i+1,j-1).
With memoization / DP this can be achieved in $\Theta(n^2)$. Overalruntime then $\Theta(n^2)$

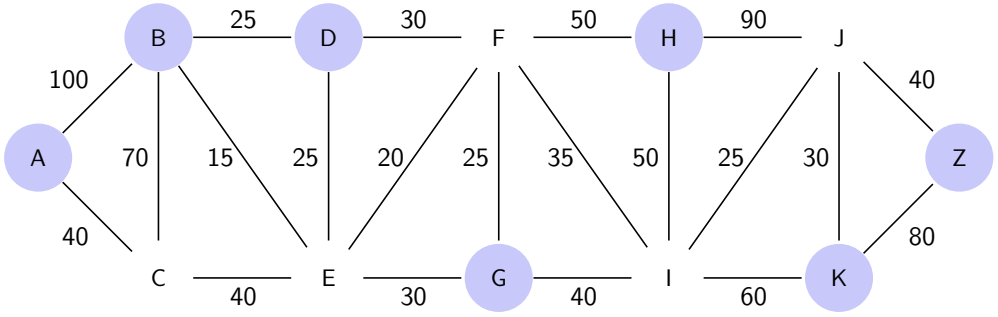
Aufgabe 4: Wasserstoffauto (14P)

Sie haben ein Auto, welches mit Wasserstoff fährt und planen eine Tour von A nach Z. Es gibt verschiedene Wege und Sie wollen den kürzesten Weg wählen. Allerdings darf der Abstand zwischen zwei besuchten Wasserstofftankstellen (markierten Knoten) einen gewissen Wert m nicht überschreiten.
Gegeben ist also ein Graph $G = (V, E, w)$ und eine Teilmenge $C \subset V$ von markierten Knoten. Jedes Gewicht w bezeichnet den Abstand zwischen zwei Knoten (in km). Ziel ist also, einen kürzesten Weg zu finden zwischen A und Z, so dass der Weg zwischen Zwei Knoten aus C den Wert m nicht übersteigt. Es gilt $A \in C$ und $Z \in C$.

You have a hydrogen car and plan a tour from A to Z. There are different paths and you want to choose the shortest path. But the distance between two points with a hydrogen filling station (marked nodes) may not exceed a certain value m.
Thus, consider a graph $G = (V, E, w)$ and a subset $C \subset V$ of marked nodes. Edge weights w determine the distance between nodes (in km). Goal is to find a shortest path from A to Z such that the distance between nodes from C does not exceed the value of m. It holds that $A \in C$ and $Z \in C$.

/2P (a) Betrachten Sie den folgenden Graph und bestimmen Sie den kürzesten Weg von A nach Z wenn $m = \infty$.

Consider the following graph and determine the shortest path from A to Z with $m = \infty$.



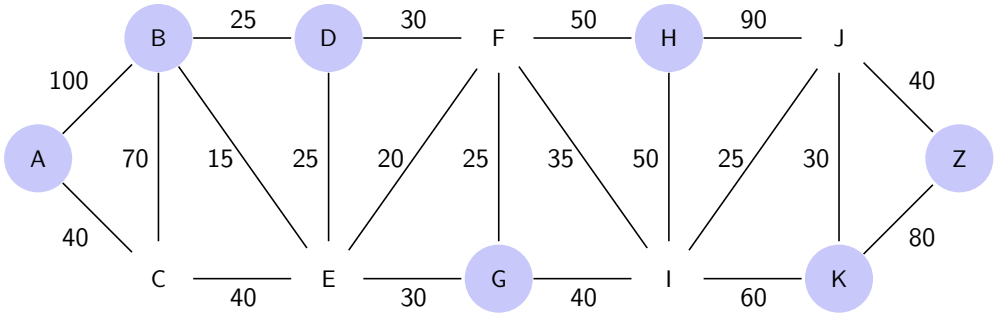
Kürzester Weg mit Länge / *Shortest path with length*

A-C-E-F-I-J-Z : 200

/3P (b) Benennen Sie den Algorithmus, den Sie hier für $m = \infty$ anwenden. Geben Sie die allgemeine Laufzeit dieses Algorithmus in Abhängigkeit von Anzahl Kanten und Knoten an.

Specify the algorithm that you choose for $m = \infty$ in order to solve the problem as efficient as possible. State the running time of this algorithm as a function of number nodes and edges.

Dijkstra Algorithm, $\Theta(|E| \log |V|)$ or, with Fibonacci Heap, $\Theta(|E| + |V| \log |V|)$.



(c) Bestimmen Sie nun die Länge des kürzesten Pfades, wobei auf dem Weg zwischen zwei Wasserstofftankstellen (markierten Knoten) der Abstand den Wert $m = 100$ nicht überschreiten darf.

Now determine the length of a shortest path where between to hydrogen filling stations (marked nodes) the distance of $m = 100$ may not be exceeded.

/3P

Kürzester Weg für $m = 100$ mit Länge / *Shortest path for $m = 100$ with length*

A-C-E-B-E-G-I-K-J-Z: 305

(d) Beschreiben Sie nun eine Algorithmus, mit dem Sie den kürzesten Pfad in einem solchen positiv gewichteten Graphen für endliches m möglichst effizient berechnen können.

Now describe an algorithm that you can use in order to compute the shortest path on a positively weighted graph for finite m, as efficient as possible.

/6P

Using the algorithm of Johnson, we first compute the shortest path for all point pairs.
We build a new graph on $G' = (C, E, w')$ with edges between $u \in C$ and $v \in C$ with weight $w'(u, v) = \delta_G(u, v)$ if and only if $\delta_G(u, v) \leq m$.
Now we run Dijkstra on G' to find the shortest path between A and Z.
The running time of Johnson is $O(|V| \cdot |E| \cdot \log |V|)$ (or, if using a Fibonacci Heap: $O(|V|^2 \cdot \log |V| + |V| \cdot |E|)$).
(Point deduction for using Floyd-Warshall with running time $O(|V|^3)$.)

Aufgabe 5: Course Scheduling (14P)

Sie haben $n \in \mathbb{N}$ Studenten und sollen diese den $m \in \mathbb{N}$ zur Verfügung stehenden Kursen zuordnen. Dabei muss jeder Student genau $k \leq m$ Kurse belegen. Kurse haben Kapazitäten C_i $(1 \leq i \leq m)$. Jeder Student gibt eine Liste von f $(k \leq f \leq m)$ bevorzugten Kursen an.
Sie suchen eine Zuteilung so dass jeder Student von den f angegebenen Kursen exakt k Kurse belegen kann und die Kurskapazitäten nicht überschritten werden.

You need to schedule courses for students. There are $n \in \mathbb{N}$ students, $m \in \mathbb{N}$ courses and each student needs to visit exactly $k \leq m$ courses. Courses have capacities C_i $(1 \leq i \leq m)$. Each student provides a list of f $(k \leq f \leq m)$ favourite courses. You are looking for an assignment such that every student can visit exactly k of his f favourite courses and such that no course capacity is exceeded.

/5P (a) Sie müssen entscheiden, ob ein zulässige Zuteilung der n Studenten zu den m Kursen überhaupt möglich ist. Sie modellieren diese Frage als ein Problem des maximalen Flusses. Beschreiben / Skizzieren Sie ein passendes Flussnetzwerk $G = (V, E, c)$. Geben Sie Knoten, Kanten und Kapazitäten des Netzwerks and und beschreiben Sie, wie Sie vom maximalen Wert des Flusses darauf schliessen können, ob es eine Zuteilung gibt oder nicht.

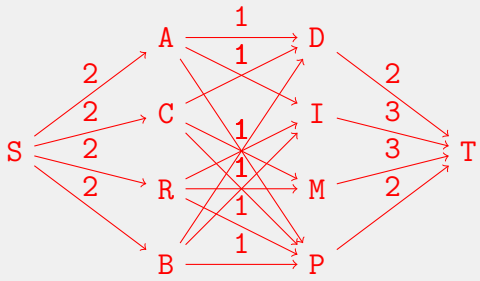
You have to decide if an acceptable assignment of the n students to the m courses is possible at all. You model that question as a network flow problem. Describe / sketch a suitable flow network $G = (V, E, c)$. Specify nodes, edges and capacities of the network and describe how you can deduce from the maximal value of the flow if a suitable assignment exists or not.

Flow network with source s and sink t. One node for each student, one node for each course. Edges with capacity k from source to each n. f Edges with capacity 1 from each student to each favourite course. Edges with capacity c_i from each course to sink t.
Determine a maximum flow from s to t. If that max-flow is $n \cdot k$, we know it worked out.

/3P (b) Zeichnen Sie das Flussnetzwerk für $k = 2$, $f = 3$ und die folgende Tabelle der bevorzugten Kurse. Die Zahlen in Klammern sind die Kurskapazitäten.

Draw the flow network for $k = 2$, $f = 3$ and the following table of favourite courses. The numbers in parentheses are course capacities.

Student	D&A (2)	Inf(2)	Math(3)	Phys(2)
Anna	x	x		x
Clara	x		x	x
Ralf		x	x	x
Bob	x	x		x



(c) Nennen Sie einen möglichst effizienten Algortihmus, mit dem Sie bestimmen können, ob eine passende Kurszuteilung existiert und bestimmen Sie die Laufzeit abhängig von den gegeben Parametern n, m, k und f.

State an algorithm, as efficient as possible, that you can use in order to determine if the suitable course assignment exists and specify its asymptotic running time as function of parameters n. m, k and f.

/4P

Ford-Fulkerson algorithm with runtime $O(f_{max} \cdot |E|) = O(k \cdot n \cdot (n \cdot f + m))$.
(Edmonds Karps Algorithm general runtime estimation would be worse:
$O(|V| \cdot |E|^2) = O((n+m) \cdot (n \cdot f + m)^2)$)

(d) Sie haben das Flussnetzwerk aus (a) bestimmt und den maximalen Fluss berechnet. Wie berechnen Sie die Zuteilung der Studenten zu Kursen. Was ist die Laufzeit dieses Algorithmus.

You have determined the flow network from (a) and computed the maximal flow. How do you compute the the assignment of students to courses? hat is the asymptotic running time of this algorithm.

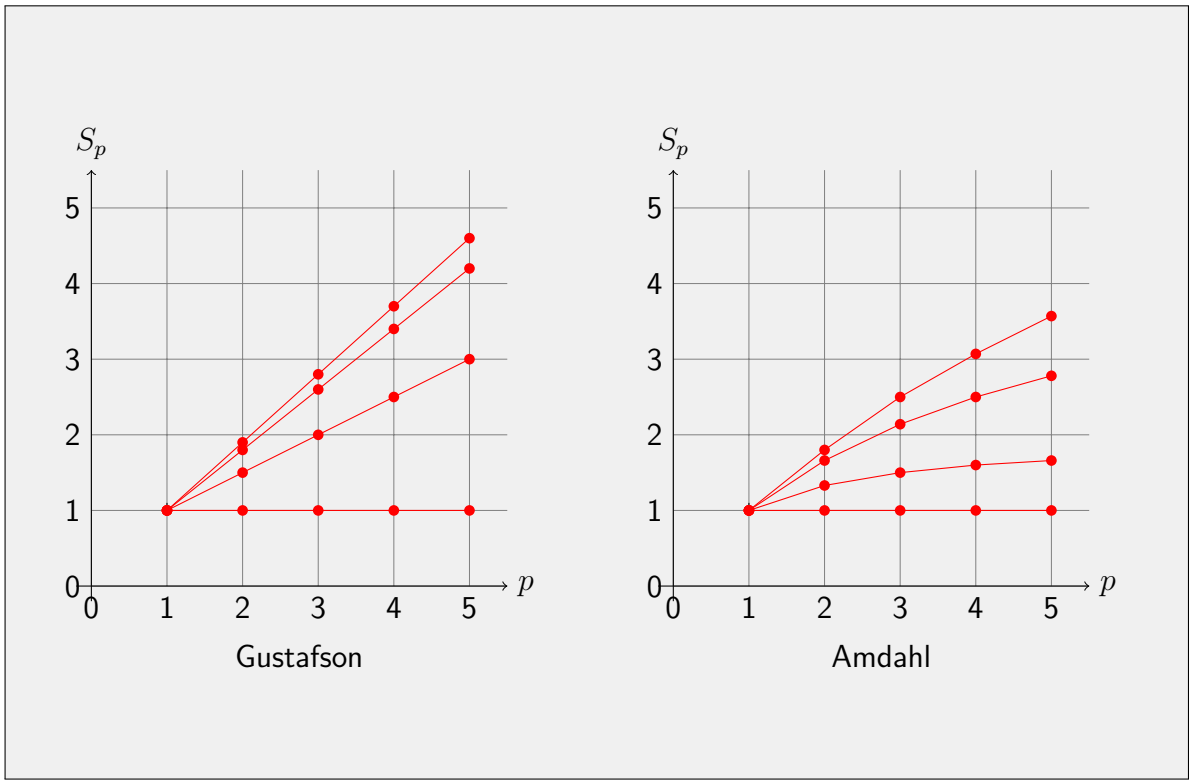
/2P

Breadth-First Search from s to all students. Assignment of student to course for all edges from student to course with positive flow. Running time of $O(|V| + |E|) = O(n + f \cdot n)$.

Aufgabe 6: Parallele Programmierung (14P)

/4P (a) Stellen Sie den Speedup nach dem Gustafsons Gesetz und nach dem Amdahl-Gesetz mit 1 bis 5 Prozessoren in einem Diagramm dar. Erstellen Sie Graphen für einen nicht-parallelisierbaren Anteil des Programmes von 10%, 20%, 50% und 100%.

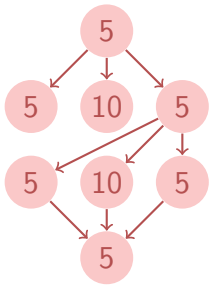
Plot the speedup when using 1 to 5 processors according to Gustafson's law and according to Amdahl's law for a program with a relative portion of the code that cannot be parallelized of 10%, 20%, 50% *and* 100%.



(b) Die folgende Abbildung zeigt einen Task-Graphen eines Algorithmus. Die Zahl an den Knoten bezeichnet die Ausführungszeit für den jeweiligen Berechnungsschritt.

The following figure shows a task-graph of some algorithm. The number in each of the nodes denotes the execution time per task step.

/2P



Markieren Sie den kritischen Pfad des Task-Graphen. Wie lang ist die Ausführungszeit des kritischen Pfades T_∞? Wie lang ist die sequentielle Ausführungszeit T_1?

Mark the critical path of the task graph. What is the execution time of the critical path T_∞? What is the sequential execution time T_1?

T_∞ =	T_1 =
25	50

(c) Was ist nach dem Greedy-Scheduling Theorem eine obere Schranke für die Ausführungszeit T_p des Taskgraphen, wenn ein Greedy-Scheduler auf p Prozessoren verwendet wird?

What is, according to the greedy scheduling theoem an upper bound for the execution time T_p of the task graph, when a greedy schediler on p processors is used.

/2P

p	2	4	5	∞
T_p	50	37.5	35	25

Wir verwenden im folgenden Teil der Aufgabe C++ Code und nehmen an, dass Sie die jeweilige Syntax und Semantik verstehen. Es geht um das theoretische Verständnis der nebenläufigen Vorgänge. Treffen Sie insbesondere keine aussergewöhnliche Annahmen über die verwendete Architektur, das Speichermodell oder das Verhalten des Compilers.

For the following part of the task, we use C++ code and assume that you understand the syntax and semantics. But this task is more about the theoretical understanding of what can happen in a concurrent setup. Particularly do not make any exceptional assumptions about the architecture used, the memory model, or the behavior of the compiler.

Es werden Threads ausgeführt. Wir gehen jeweils davon aus, dass die Funktion `print` jeweils einen Buchstaben ausgibt. Geben Sie für die Programme jeweils alle möglichen Ausgaben an, allenfalls getrennt durch Semikolon. Bestimmen Sie, ob das Programm terminiert. Wenn das Programm nicht terminiert, geben Sie alle Ausgaben immer so weit an, wie sie auftreten können.

Threads are executed. We assume for each part that the function `print` outputs a character. For each of the programs, provide all possible outputs, separated by semicolons if necessary. Specify if the program terminates. If the program does not terminate, provide all possible outputs as far as they can occur.

/2P (d)

```
void print(char c); // output character c

void A(char value){
  print(value);
}

void B(char value){
  print(value);
  std::thread t1(A,value+1);
  t1.join();
}
int main(){
  std::thread t1(B,'A');
  std::thread t2(B,'B');
  t1.join();
  t2.join();
}
```

mögliche Ausgabe(n)/*possible output(s)*

ABBC ABCB BABC BACB BCAB

Das Programm terminiert/*the program terminates*

√ immer/*always* ◯ nie/*never* ◯ manchmal/*sometimes*

(e) /2P

```
std::mutex m1;
std::mutex m2;
void print(char c); // output character c

void A(){
  m1.lock();
  print('A');
  m2.lock();
  print('B');
  m1.unlock();
  print('C');
  m2.unlock();
}

int main(){
  std::thread t1(A);
  std::thread t2(A);
  t1.join();
  t2.join();
}
```

mögliche Ausgabe(n)/*possible output(s)*

ABCABC ABACBC

Das Programm terminiert/*the program terminates*

√ immer/*always* ◯ nie/*never* ◯ manchmal/*sometimes*

(bitte wenden / *turn page*)

/2P (f)

```
void print(char c); // output character c

using guard = std::unique_lock<std::mutex>;
std::mutex m;
std::condition_variable c;

void A(bool& mine, bool& other){
  guard g(m);
  print(mine ? 'A':'B'); // if mine == true: output A, else B
  c.notify_one();
  other = true;
  c.wait(g, [&]{return mine;});
}

int main(){
  bool one = false;
  bool two = false;
  std::thread t1(A,std::ref(one), std::ref(two));
  std::thread t2(A,std::ref(two), std::ref(one));
  t1.join();
  t2.join();
}
```

mögliche Ausgabe(n)/*possible output(s)*

```
BA
```

Das Programm terminiert/*the program terminates*
√ immer/*always* ◯ nie/*never* ◯ manchmal/*sometimes*

/15P

Aufgabe 7: Programmieraufgabe: Job Scheduler (15P)

Diese Aufgabe zum Thema Parallele Programmierung soll am Computer gelöst werden. Sie können sich die Zeit frei einteilen. Wir **empfehlen** aber, dass Sie **nicht mehr als 45 Minuten** für diese Aufgabe aufwenden.

Lösen Sie diese Aufgabe in der Online-Umgebung (Code Expert via Moodle).

This following part on parallel programming needs to be solved at the computer. You are free in how you divide the time. However, we **recommend to spend not more than 45 minutes** *on that problem.*

Solve this task in the online environment (Code Expert via Moodle).

Moodle-Passwort wird zu Beginnn der Prüfung bekannt gegeben / *Moodle password will be announced at the beginning of the exam*