Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Prüfung **(Lösung)**

Datenstrukturen und Algorithmen (D-MATH RW)

Felix Friedrich, Departement Informatik

ETH Zürich, 7.8.2017.

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: 120 Minuten.
2. Erlaubte Unterlagen: Wörterbuch (für 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: 120 minutes.*
2. *Permitted examination aids: dictionary (for spoken languages). 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 false 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	Total
Points:	10	7	10	8	7	7	49
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 reproducible.

Aufgabe 1: Verschiedenes (10P)

1) In dieser Aufgabe sollen nur Ergebnisse angegeben werden. Sie können sie direkt bei den Einzelteilen notieren.

   In this task only results have to be provided. You can note them down in the parts.

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 lexicographic order and for numbers we take the increasing order according to their sizes.

/1P (a) Gegeben seinen Daten der Länge n. Wie viele der folgenden Operationen führt der Algorithmus Sortieren durch Auswahl im schlechtesten Fall durch?

Given data with length n. How many of the following operations are executed by the algorithm Selection Sort in the worst case?

Anzahl Vergleiche/*Number Comparisons* : ◯ $\Theta(n)$ ◯ $\Theta(n \log n)$ √ $\Theta(n^2)$

Anzahl Vertauschungen/*Number Swaps* : √ $\Theta(n)$ ◯ $\Theta(n \log n)$ ◯ $\Theta(n^2)$

/1P (b) Führen Sie auf dem folgenden Array zwei weitere Iterationen des Algorithmus Quicksort aus. Als Pivot wird jeweils das erste Element des (Sub-)Arrays genommen.

Execute two further iterations of the algorithm Quicksort on the following array. The first element of the (sub-)array serves as the pivot.

8	7	10	15	3	6	9	5	2	13
2	7	5	6	3	**8**	9	15	10	13
2	7	5	6	3	**8**	**9**	15	10	13
2	3	5	6	**7**	**8**	**9**	13	10	**15**

(c) Im folgenden ist eine Hashtabelle dargestellt. Es wird offenes Hashing mit der Hashfunktion $h(k) = k \bmod 11$ verwendet. Kollisionen werden mit linearem Sondieren aufgelöst. Die Sondierung läuft immer nach links. Vervollständigen Sie die angegebene Einfügereihenfolge so, dass die Belegung entsteht, wenn man mit einer zu Anfang leeren Hashtabelle startet. Wie viele Kollisionen sind insgesamt aufgetreten?

In the following a hash table is displayed. We use open hashing with the hash function $h(k) = h \bmod 11$. Collisions are resolved using linear probing. Probing always goes left. Complete the given insertion order such that the hash table as given would be produced from an initially empty hash table. How many collisions took place overal?

/1P

		18	3	16	5	7	29			
0	1	2	3	4	5	6	7	8	9	10

Einfügereihenfolge/*Insertion Order*: 3, 5, 16, 29, 7, 18

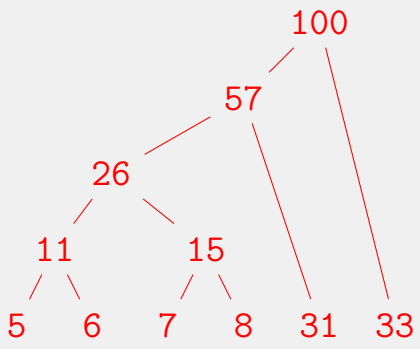
Anzahl Kollisionen/*Number collisions*: ◯ 3 ◯ 4 ◯ 5 ◯ 6 √ 7

(d) 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.

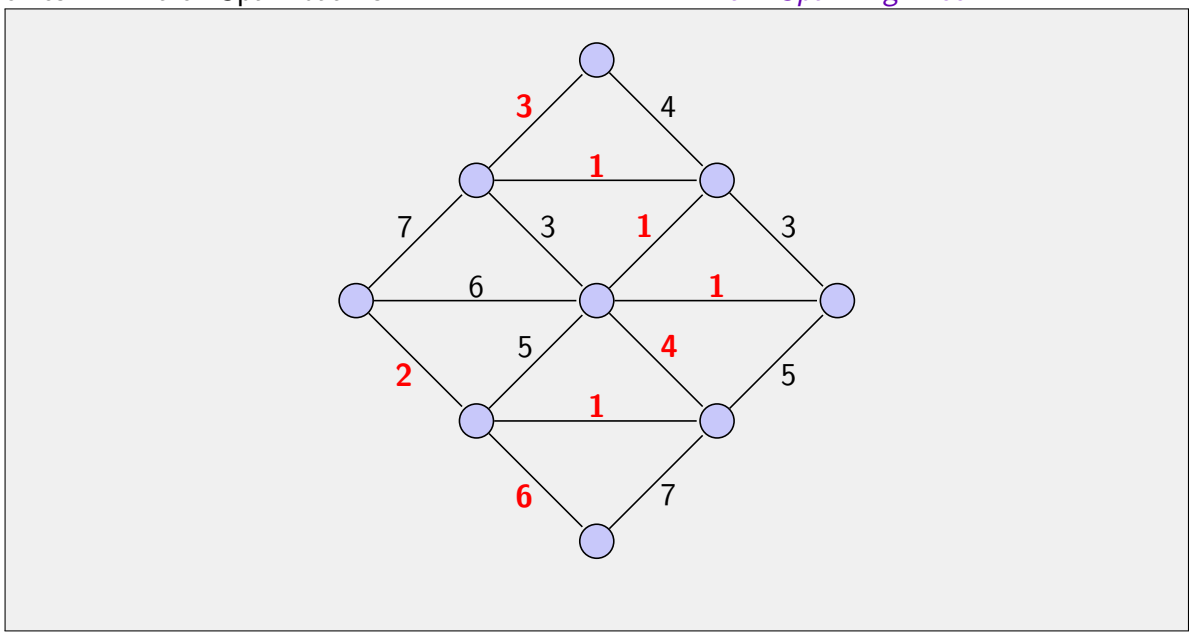
/1P

char	a	b	c	d	e	f
freq	5	6	31	33	7	8
Code	1111	1110	10	0	1101	1100



/1P (e) Markieren Sie im Graphen unten die Kanten eines minimalen Spannbaums.

In the graph below mark the edges of a Minimum Spanning Tree.



Gegeben seien nun folgende Daten, eine Sequenz von Zahlen

Now consider the following data, a sequence of numbers.

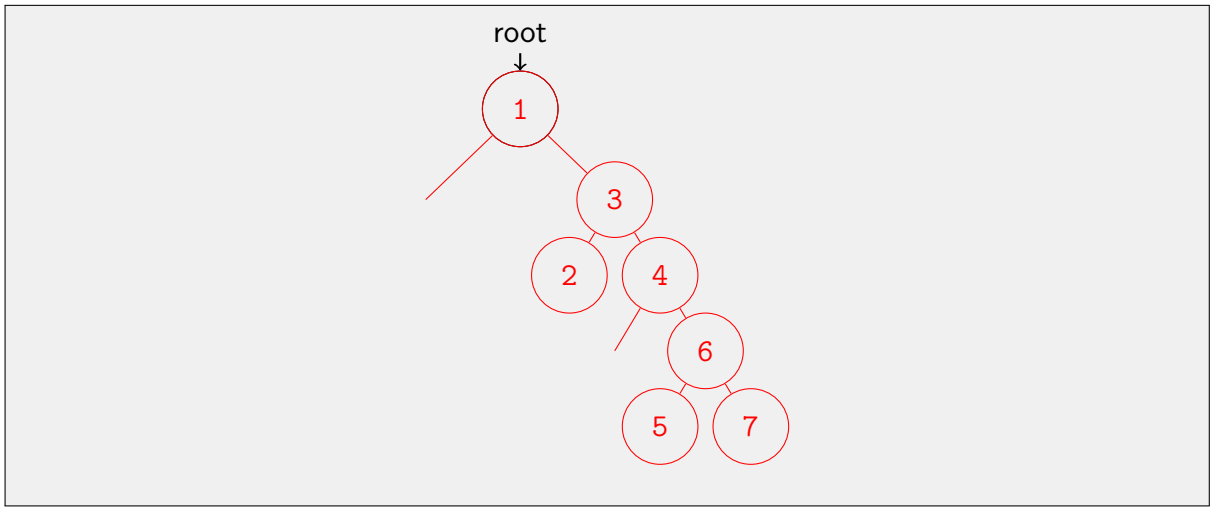
1, 3, 4, 2, 6, 5, 7

Vervollständigen Sie folgende Figuren, so dass sie deutlich machen, wie die jeweilige Datenstruktur – wie in Vorlesung und Übungen gezeigt – aussieht, wenn die Werte in obiger Reihenfolge (von links nach rechts) eingefügt werden.

Complete the following figures such that they illustrate the respective data structures – as presented in lectures and exercises – after insertion of the given data in the order above (from left to right).

/1P (f) Binärer (unbalancierter) Suchbaum

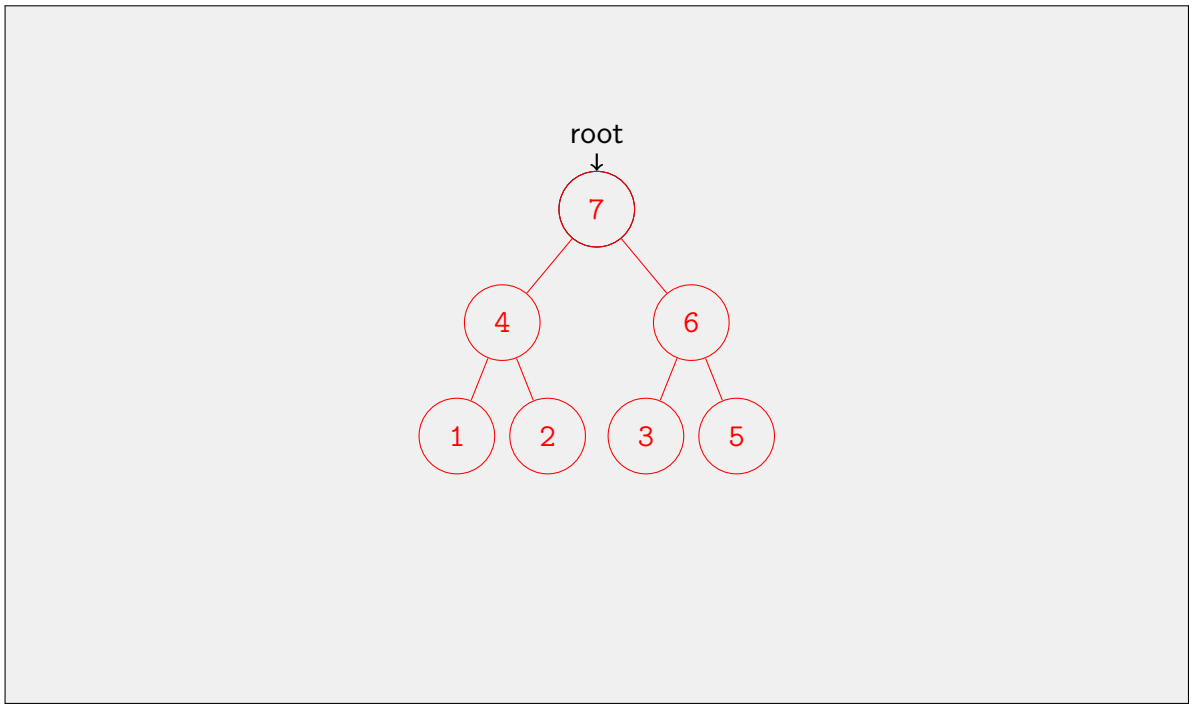
Binary (unbalanced) search tree



(g) Max-Heap (als Baum repräsentiert). Tipp: zeichnen Sie sich Zwischenschritte auf.

Max-Heap (represented as Tree). Hint: draw intermediate steps.

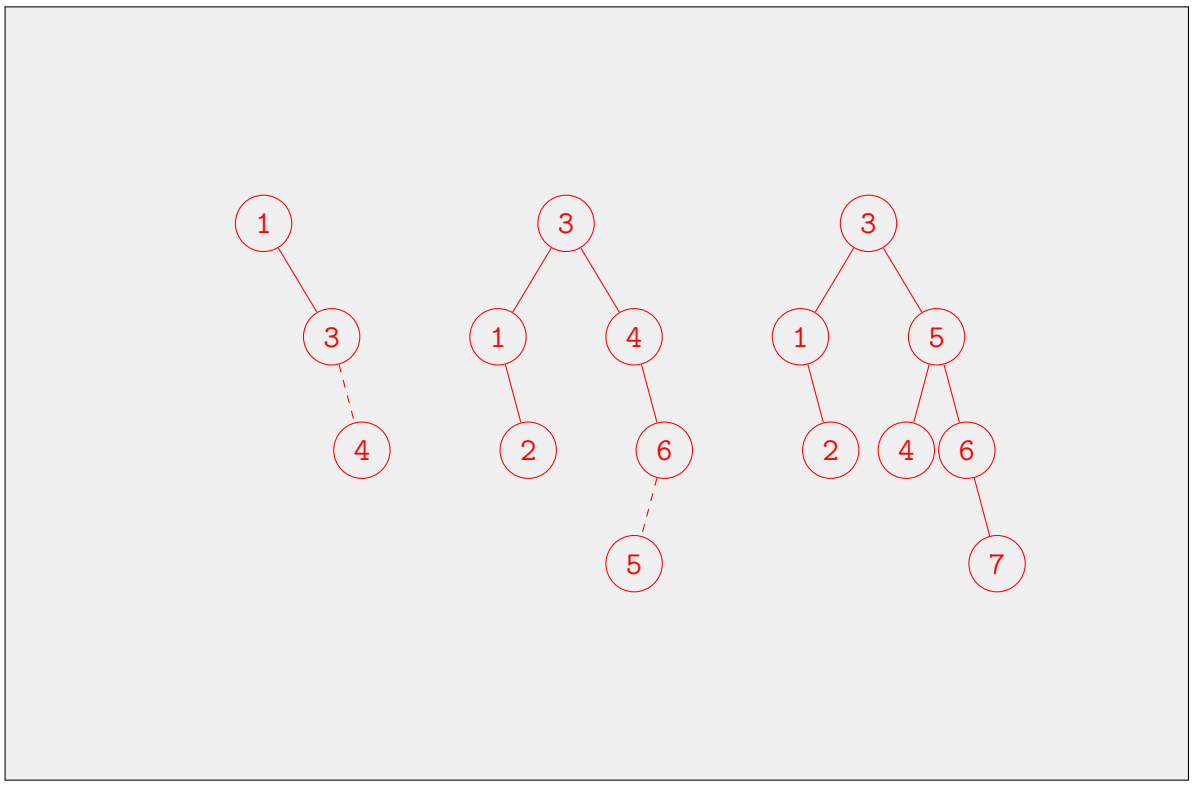
/1P



(h) AVL-Baum. Zeichnen Sie den Baum nach jeder Balancierung neu.

AVL-Tree. Draw a new tree after each balancing.

/1P



/1P (i) Betrachten Sie einen binären Suchbaum als gerichteten Graph. Kreuzen Sie an, was im allgemeinen stimmt.

Consider a binary search tree as a directed graph. Mark what is generally correct.

◯Die Traversierung in Hauptreihenfolge ist eine Breitensuche	*The preorder traversal is a breadth first search*
√Die Traversierung in Hauptreihenfolge ist eine Tiefensuche	*The preorder traversal is a depth first search*
◯Die Traversierung in Nebenreihenfolge ist eine Breitensuche	*The postorder traversal is a breadth first search*
√Die Traversierung in symmetrischer Reinhenfolge ist eine Tiefensuche	*The In-order traversal is a depth first search*

/1P (j) Ein Compiler muss eine grosse Menge Symbole (Zeichenketten) verwalten. Es muss möglich sein, während der Kompilation viele Namen neu einzutragen und effizient danach zu suchen. Löschen steht nicht im Vordergrund. Welche Datenstruktur schlagen Sie für die Verwaltung der Symbole vor?

A compiler has to manage a large number of symbols (strings). It has to be possible to add a lot of names during compilation and to search for a symbol efficiently. Deletion is not of importance. Which data structure do you propose to use for the management of the symbols?

Entweder eine Hashtabelle mit erwartet konstantem Lookup oder ein balancierender Baum, wie z.B. AVL Baum mit Einfüge- und Suchkosten $O(\log n)$.

Aufgabe 2: Asymptotik (7P)

/1P (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 $n^5 \in \mathcal{O}(n^7)$.

$$\sqrt{n}\log n, \quad n\log\sqrt{n}, \quad 1/n, \quad \log(\sum_{i=1}^{n} i), \quad n!, \quad n^n$$

$1/n, \quad \log(\sum_{i=1}^{n} i), \quad \sqrt{n}\log n, \quad n\log\sqrt{n}, \quad n!, \quad n^n$

(b) Gegeben Sei die folgende Rekursionsgleichung:

Consider the following recursion:

/3P

$$T(n) = \begin{cases} 3 \cdot T(n-1) + 2^n, & n > 0 \\ 1 & n = 0 \end{cases}$$

Geben Sie eine geschlossene (nicht rekursive), einfache Formel für $T(n)$ an und beweisen Sie diese mittels vollständiger Induktion.
Hinweis:
Für $q \neq 1$ gilt $\sum_{i=0}^{k} q^i = \frac{q^{k+1}-1}{q-1}$.

Provide a closed (non-recursive), simple formula for $T(n)$ and prove it using mathematical induction.
Hint:
For $q \neq 1$ it holds that $\sum_{i=0}^{k} q^i = \frac{q^{k+1}-1}{q-1}$.

$$\begin{aligned} T(n) &= 3T(n-1) + 2^n = 3(3T(n-2) + 2^{n-1}) + 2^n \\ &= 3(3(3T(n-3) + 2^{n-3}) + 2^{n-1}) + 2^n \\ &= 3^n \cdot T(0) + 3^{n-1} \cdot 2^1 + 3^{n-2} \cdot 2^2 + \cdots + 3 \cdot 2^{n-1} + 1 \cdot 2^n \\ &= \sum_{k=0}^{n} 3^k \cdot 2^{n-k} = 2^n \cdot \sum_{k=0}^{n} 3^k 2^{-k} \\ &= 2^n \cdot \frac{(3/2)^{n+1} - 1}{3/2 - 1} = 2^{n+1} \cdot ((3/2)^{n+1} - 1) \\ &= 3^{n+1} - 2^{n+1} \end{aligned}$$

Induktion:

1. Hypothese $T(n) = 3^{n+1} - 2^{n+1}$.

2. Anfang $T(0) = 3 - 2 = 1$.

3. Schritt $(n-1 \to n)$: $T(n) = 3T(n-1) + 2^n = 3(3^n - 2^n) + 2^n = 3^{n+1} - 3 \cdot 2^n + 2^n = 3^{n+1} - 2^{n+1}$

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 $f()$ as a function of $n \in \mathbb{N}$ using Θ notation as tight as possible. The function f does not call itself. You do not have to justify your answers.

/1P (c)

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

Anzahl Aufrufe von f / *Number Calls of f*

$\Theta(\log n)$

/1P (d)

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

Anzahl Aufrufe von f / *Number Calls of f*

$\Theta(n)$

/1P (e)

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

Anzahl Aufrufe von f / *Number Calls of f*

$\Theta(2^n)$

Aufgabe 3: Dynamic Programming (10P)

Es soll eine Stromleitung von einem bestehenden Startmast S zu einem bestehenden Endmast E mit Abstand n km durch unwegsames Gelände verlegt werden. Kosten verursachen zum einen das Aufstellen der Stromleitungsmasten und zum anderen das Verbinden mit Stromkabeln.
Eine Firma A bietet Ihnen an, Leitungsmasten aufzustellen. Sie gibt Ihnen zu Punkten, welche sich äquidistant mit jeweils 1 km Abstand zwischen S und E befinden, jeweils die Kosten m_i $(0 < i < n)$ für das Aufstellen eines Masts an. Die Kosten variieren wegen des unwegsamen Geländes.
Eine andere Firma B bietet Ihnen an, zwischen den Leitungsmasten Drähte zu spannen. Die Kosten sind jedoch (nichtlinear) vom Abstand der zu verbindenden Leitungsmasten abhängig. Es ist nun Ihre Aufgabe, eine möglichst günstige Konstellation von Leitungsmasten zu finden.
Nachfolgend finden Sie eine Beispieltabelle mit Angeboten.

A power cable from some existing power mast S to an existing end mast E with a distance of n km shall be installed in rough terrain. The costs are determined by the installation of the power masts and by the connection with power cables.
A company A offers to install the power masts. For the installation of a power mast, to given points that are placed equidistantly with a distance of 1 km each between S and E the company quotes a price m_i $(0 < i < n)$. The costs vary because of the rough terrain.
A different company B offers to install power cables between the masts. However, the costs depend (in a non-linear fashion) on the distance of the masts to be connected. Now, it is your task to find the cheapest constellation of power masts.
Below you see an example table with the company's offers.

Firma / *Company* A

Mast / *mast*	$M_0 = S$	M_1	M_2	M_3	M_4	M_5	$M_6 = E$
Punkt bei km / *Point at km*	0	1	2	3	4	5	6
Kosten / *costs* m	0	4	3	4	2	4	0

Firma / *Company* B

Abstand / *distance* d (km)	1	2	3	4	5	6	7
Kosten / *costs* l_d	1	4	9	16	25	36	49

Beispiel: eine Verlegung mit Masten an den Punkten 2, 3 und 5 kostet $4+3+4 = 11$ für die Masten plus $4 + 1 + 4 + 1 = 10$ für die Kabel (von 0 nach 2, 2 nach 3, 3 nach 5 und 5 nach 6), also insgesamt 21.

Example: an installation of masts at the points 2, 3 and 5 causes costs $4+3+4 = 11$ for the masts plus $4 + 1 + 4 + 1 = 10$ for the cables (from 0 to 2, 2 to 3, 3 to 5, and 5 to 6), thus overal costs of 21.

Geben Sie einen Algorithmus an, der nach dem Prinzip der dynamischen Programmierung arbeitet und die kleinstmöglichen Kosten (wie oben beschrieben) berechnet. Gehen Sie in Ihrer Lösung auf die folgenden Aspekte ein.
(Der triviale Algorithmus, der alle Lösungen aufzählt, ergibt keine Punkte.)

Provide a dynamic programming algorithm for computing the minimum cost (as described above). Address the following aspects in your solution.
(The trivial algorithm that simply enumerates all possible solutions does not give any points.)

(a) /4P

(I)
Was ist die Bedeutung eines Tabelleneintrags, und welche Grösse hat die DP-Tabelle T?

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

Tabellengrösse / *table size*: $1 \times n$ Tabelle

Bedeutung Eintrag / *entry meaning*: t_k Niedrigste Kosten bis zum Mast k

(II)
Wie berechnet sich ein Eintrag der Tabelle aus den vorher berechneten Einträgen?

How can an entry be computed from the values of previously computed entries?

$t_k = M(k) + \arg\min l \in 0..k\{L(l) + t_{k-l}\}$

(III)
In welcher Reihenfolge können die Einträge berechnet werden?

In which order can the entries be computed?

von links nach rechts mit aufsteigendem k

(IV) Wie kann der Wert des minimalen Preises aus der Tabelle erhalten werden?

How can the value of the minimum price be obtained from the DP table?

Die Lösung steht an der Stelle t_n

/3P (b) Zusätzlich zu den optimalen Kosten wollen Sie nun auch angeben, an welchen Orten die Masten aufgestellt werden sollen. Beschreiben Sie detailiert, wie Sie das bewerkstelligen.

In addition to the optimal costs you now want to provide the locations of the masts. Describe in detail how you can determine this.

Wir führen eine zweite $1 \times n$-Tabelle mit dem optimalen Vorgänger. Während der Berechnung des optimalen Preises, tragen wir den optimalen Vorgänger ein. Bestimmung der Masten durch Rückwärtsverfolgung der optimalen Vorgänger bis zum Vorgänger 0.

/1P (c) Geben Sie nun eine günstigste Verlegung für obiges Beispielszenario an. An welchen Orten würden Sie Masten aufstellen und wieviel kostet das?

Provide an optimal setup for the example scenario above. Where would you place masts and what are the costs?

Masten an den Stellen 2 und 4 mit Kosten 17

/2P (d) Geben Sie die Laufzeiten für die Berechnung der optimalen Kosten und die Berechnung der Orte in Θ-Notation möglichst knapp mit Begründung an.

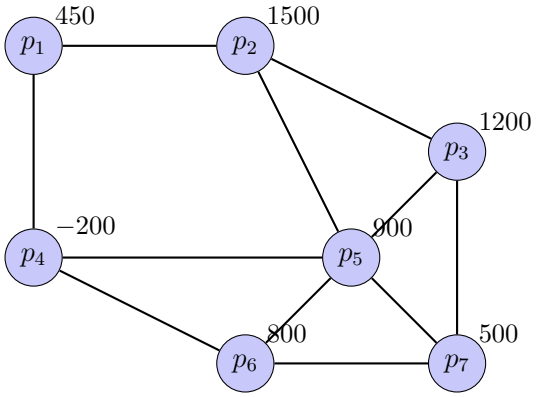
Provide the run-times for the computation of the optimal costs and the computation of the placements in Θ notation as tight as possible with short explanation.

Bestimmung der Tabelle: $\Theta(n^2)$, für jeden Eintrag müssen alle vorherigen Einträge durchlaufen werden, also $\sum_{i=1}^{n+1} i = (n+1) \cdot (n+2) = \Theta(n^2)$. Berechnung der optimalen Lösung durch Rückverfolgung im schlechtesten Fall $\Theta(n)$, da von jedem k auf ein kleineres k verwiesen wird.

Aufgabe 4. (8P)

Gegeben sei ein bergiges Wandergebiet mit n interessanten Zielpunkten $p_i (1 \leq i \leq n)$ welche jeweils auf Höhen $h(p_i) \in \mathbb{R} (1 \leq i \leq n)$ liegen. Ausserdem gibt es eine Menge $W \subset \{\{p_i, p_j\}, 1 \leq i, j \leq n\}$ von möglichen Wanderwegen zwischen den Punkten. Jeder interessante Punkt ist mit maximal fünf Wegen erreichbar. Ein Wanderer möchte von p_1 nach p_n wandern. Er möchte möglichst wenige Höhenmeter durchlaufen, sucht also nach einem Weg $(p_{i_j}), 1 \leq j \leq n$ mit minimaler Summe $\sum_{j=1}^{k-1} |h(p_{i_j}) - h(p_{i_{j+1}})|$.

Consider a hiking region in the mountains with n interesting destinations $p_i (1 \leq i \leq n)$ that are located at heights $h(p_i) \in \mathbb{R} (1 \leq i \leq n)$. Moreover, there is a set $W \subset \{\{p_i, p_j\}, 1 \leq i, j \leq n\}$ of possible paths between the destination points given. Every destination can be reached with a maximum of 5 different paths.
A wanderer wants to hike from p_1 to p_n. He wants to climb and descend as little as possible, thus he searches for a path $(p_{i_j}), 1 \leq j \leq n$ with minimal sum $\sum_{j=1}^{k-1} |h(p_{i_j}) - h(p_{i_{j+1}})|$.



(a) Geben Sie den verwendeten Graph und einen möglichst effizienten Algorithmus zur Bestimmung eines gewünschten Weges an. Geben Sie die Laufzeit Ihres Algorithmus (im schlechtesten Fall) in Abhängigkeit von der Anzahl n der interessanten Punkte an.

Provide the used graph and an efficient algorithm for the determination of the best possible path. Provide asymptotic running time of your algorithm (in the worst case) depending on the number n of destinations.

/3P

Suche nach kürzestem Weg auf dem Graph $G = (V, E, c)$ mit $V = \{p_1, \ldots, p_n\}$, $E = W$, $c_{ij} = |h(p_i) - h(p_j)|$. Alle Kanten positiv, daher bietet sich Dijkstra's Algorithmus an. Der Graph hat maximal $O(n)$ Kanten. Also ergibt sich eine Laufzeit von $O(|E| \log |V|) = O(n \log n)$.

/3P (b) Da das Wandergebiet sehr beliebt ist, stehen am Punkt p_1 immerzu beliebig viele Wanderer zur Wanderung bereit und laufen nach p_n. Die durchlaufene Höhe spielt nun keine Rolle mehr. Ein findiger Busunternehmer möchte genügend Busse zum Abholen der Wanderer am Punkt p_n bereitstellen. Zu jedem Wegstück zwischen zwei interessanten Punkten kennt er die maximale Anzahl Wanderer, die den Weg pro Stunde durchlaufen können. Wie kann der Busunternehmer herausfinden, wie viele Wanderer beim Punkt p_n maximal pro Stunde abgeholt werden müssen?
Geben Sie auch hier die Laufzeit des Algorithmus in Abhängigkeit von der Anzahl interessanter Punkte n an.

Because the hiking area is very popular, at point p_1 there are always wanderer ready to start their hiking tour to p_n. The altitude difference is not of importance. A bus tour operator want to provide enough busses to pick up the wanderer at point p_n. For each way between two destinations the operator knows the maximal number of wanderers that can pass per hour. How can the bus operator find out how many wanderers have to be picked up at point p_n maximally per hour?
Again provide the asymptotic runtime of the algorithm depending on the number of destinations n.

Problem des maximalen Flusses. Ford-Fulkerson Algorithmus / Edmonds Karp Algorithmus. Anzahl Flusserhöhungen $O(|V| \cdot |E|) = O(n^2)$, also Gesamtlaufzeit des Algorithmus $O(|V| \cdot |E|^2) = O(n^3)$.

/2P (c) Ein Wanderer zeichnet die Höhendaten einer sehr langen Tour im Minutentakt auf. Hinterher möchte er den anstrengendsten Teil seiner Wanderung anzeichnen und sucht dafür den zusammenhängenden Teil der Strecke, in dem er am meisten Höhenmeter zurückgelegt hat. Der Teil der Strecke kann auch Gefälle enthalten, welches mit entsprechend negativer Steigung aber auch im Gegensatz zu (a) oben negativ zählt.
Geben Sie einen Algorithmus an, mit dem man den zusammenhängenden Teil der Strecke mit den meisten Höhenmetern möglichst effizient bestimmen kann. Wie viele Schritte macht der Algorithmus in Abhängigkeit von der Anzahl Datenpunkte m?

A wanderer records the height data of a very long tour, one sample per minute. Afterwards he wants to mark the most exhausting part of the tour and therefore is looking for the contiguous part with the largest (positive) height difference. The part can contain descending pieces, but in contrast to (a) above now they do count negative.
Provide an algorithm that efficiently can determine the piece of the tour with the largest height difference. How many steps does the algorithm require depending on the number data points m?

Er verwendet den Maximum-Subarray Algorithmus auf dem Array der Höhendifferenzen. Das geht in $\Theta(m)$

Aufgabe 5: Parallele Programmierung (7P)

(a) Angenommen ein Programm sei zu 80% parallelisierbar und erreiche auf mehreren Prozessorkernen einen Speedup von 3. Berechnen Sie mit Hilfe von Amdahls Gesetz die minimale Anzahl verwendeter Kerne.

Consider a program that is 80% parallelizable and reaches a speedup of a factor of 3 on multiple processor cores are used. Calculate the minimum number of used processor cores using Amdahl's law.

/1P

$$S_p = \frac{1}{\lambda + (1-\lambda)/p}$$

$$3 = \frac{1}{0.2 + 0.8/p} \Leftrightarrow 0.6 + 2.4/p = 1 \Leftrightarrow p = 6$$

(b) Wie gross ist unter denselben Voraussetzungen (80% Parallelisierbarkeit) der maximal erreichbare Speedup mit beliebig vielen Prozessorkernen?

What is the maximum possible speedup under the same conditions (80% parallelizability) with an unlimited number of processor cores?

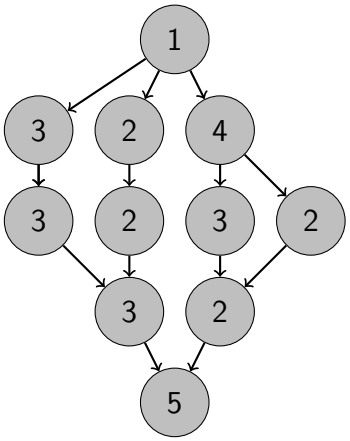
/1P

$$S_\infty = \frac{1}{\lambda + (1-\lambda)/\infty} = 5$$

(c) Geben Sie für folgenden Taskgraphen eine obere Schranke an für die Ausführungszeit bei Einsatz von beliebig vielen Prozessorkernen und bei drei Kernen unter Verwendung eines beliebigen gierigen Schedulers. Die Zahlen in den Knoten bezeichnen die jeweilige Ausführungszeit in Nanosekunden.

For the following task graph provide an upper bound for the execution time when an unlimited number of processor cores is used and the execution time when three cores are used and an arbitrary greedy scheduler is employed. The number in the nodes provide execution time in nanoseconds.

/2P



$T_\infty =$ 15

$T_3 \leq$ 15 + 30/3 = 25

Das Programm auf der nächsten Seite wird mit einem optimierenden Compiler kompiliert und auf moderner Hardware ausgeführt.

The program on the next page is compiled with an optimizing compiler and is executed on modern hardware.

/1P (d) Auf ein und demselben System wird manchmal "cold and dry", manchmal "warm and wet" und manchmal "warm and dry" ausgegeben. Wie nennt man dieses Verhalten und warum tritt es auf?

On the same system, sometimes you see the output "cold and dry", sometimes "'warm and wet" and sometimes "warm and dry". How ist this behavior called and why does it happen?

Race condition: Die Ausführungsreihenfolge der Threads spielt eine Rolle. Hier genauer: es genügt schon ein bad interleaving.

/1P (e) Auf einem anderen System bleibt die Funktion `report` sogar in einer Endlosschleife hängen, haben Sie auch dafür eine Erklärung?

On a different system the function `report` is even blocked in an endless loop. Can you explain this?

Datarace: compiler optimiert das data = true weg (Compiler) oder die lokale Änderung wird für andere Threads nicht sichtbar (Cache).

/1P (f) Beschreiben Sie ganz kurz / stichwortartig, wie Sie das Verhalten der Klasse `Weather` unter Beibehaltung ihrer beabsichtigten Funktionalität und Verwendung mehrerer Threads deterministisch machen.

Describe, very shortly / in note form, how you would render the behavior of the class `Weather` deterministic such that it provides the intended functionality when multiple threads are used.

Verwendung von Locks und einer condition variable für die Daten.
[Alternativ: lock-freie Implementation mit atomaren Variablen. Volatile ist auch ok]

```
#include <iostream>
#include <thread>

class Weather{
  int temp = 0;
  int rain = 0;
  bool data = false;

public:

  // report the current weather data in words
  // as soon as data are available
  void report(){
      while (!data);
      if (temp < 20)
        std::cout << "cold and ";
      else
        std::cout << "warm and ";
      if (rain > 0)
        std::cout << "wet";
      else
        std::cout << "dry";
  }

  // provide weather data
  void update(int t, int r){
      data = true;
      temp = t;
      rain = r;
  }
};

int main()
{
    Weather w;
    std::thread t1([&]{w.report();});
    std::thread t2([&]{w.update(20,10);});
    t2.join(); t1.join();
    return 0;
}
```

Aufgabe 6: Parallele Programmierung (7P)

Eine Verkehrssimulation enthält eine Menge von Objekten, die eigenständig (als Threads implementiert) agieren. Darunter auch Autos und Kinder. Nun soll ein Zebrastreifen zur Simulation hinzugenommen werden, mit folgender Semantik:

(A) Zu jeder Zeit können maximal zwei Autos den Zebrastreifen befahren.

(B) Zu jeder Zeit können bis zu 20 Kinder den Zebrastreifen betreten.

(C) Zu keiner Zeit können Autos und Kinder gemeinsam auf dem Zebrastreifen sein.

Wir setzen voraus, dass Autos und Kinder sofort anhalten (können), wenn die Strasse belegt ist.

A traffic simulation comprises a set of objects that act autonomously (implemented as threads). Amongst such objects there are cars and children. Now a zebra crossing shall be implemented providing the following semantics:

(A) At each point in time a maximal number of two cars can be on the zebra crossing.

(B) At each point in time a number of at most 20 children can be on the zebra crossing.

(C) At no point in time cars and children can be on the zebra crossing together.

For simplicity we assume that children and cars (can) stop immediately when the crossing is already occupied.

/4P (a) Vervollständigen Sie nachfolgenden Code so, dass obige Semantik implementiert wird. Die Autos und Kinder betreten und verlassen den Zebrastreifen jeweils mit den Funktionen `car_enter`, `kid_enter`, `car_leave` und `kid_leave` der Klasse `Zebra`. Es ist nicht gefordert, dass Ihre Implementation die Autos den Zebrastreifen in der Reihenfolge ihrer Ankunft passieren lassen. Ebenso gibt es (noch) keine Vorrangregeln.

Complement the following code such that the semantics from above are implemented. Cars and children enter and leave the zebra crossing calling the functions `car_enter`, `kid_enter`, `car_leave` und `kid_leave` of the class `Zebra`, respectively. It is not required that cars pass the zebra crossing in the order of their arrival. Moreover, there are no priority rules (yet).

```
// includes and shortcuts on this page in order to save space

#include <mutex> // for std::mutex and std::unique_lock
#include <condition_variable> // for std::condition_variable

using unique_lock = std::unique_lock<std::mutex>;

// program continues on the right hand side
```

```
class Zebra{
 private:
  int kids = 0;
  int cars = 0;
    std::mutex mtx;
    std::condition_variable cv;
 public:
  void car_enter(){
    unique_lock lock(mtx);
    cv.wait(lock, [&]{return cars <2 && kids == 0});
    cars++;
  }
  void car_leave(){
    unique_lock lock(mtx);
    cv.notify_all(); // order of notify / cars-- is not crucial
    cars--;
  }
  void kid_enter(){
    unique_lock lock(mtx);
    cv.wait(lock, [&]{return cars == 0 && kids < 20};
    kids++;
  }
  void kid_leave(){
    unique_lock lock(mtx);
    cv.notify_all(); // order of notify / kids-- is not crucial
    kids--;
  }
}
```

/3P (b) Es wird nun eine Vorrangregel eingeführt: Kinder müssen nie auf Autos warten (welche nicht bereits auf dem Zebrastreifen sind). Wie implementieren sie das genau?

Now a priority rule is introduced: Children may not have to wait for cars (that are not already on the zebra crossing). How exactly do you implement this?

```
Zusätzliche Variable waiting_kids = 0;
Autos warten mit
  cv.wait(lock, kids == 0 && waitingKids == 0 && cars < 2);
Kinder warten mit
  waitingKids++;
  cv.wait(l, [&]{return cars == 0 && kids < 20};
  waitingKids--;
```