Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Prüfung **(Lösung)**

Datenstrukturen und Algorithmen (D-MATH RW)

Felix Friedrich, Departement Informatik

ETH Zürich, 6.8.2018.

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 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: 120 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:	19	15	18	15	14	17	9	107
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 (19P)

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.

/2P (a) Wie viele Vergleiche benötigt man asymptotisch **im schlechtesten Fall** für die Suche nach einem Element in einem Array der Länge n? Wählen Sie jeweils die schärfste Schranke für einen besten bekannten Algorithmus aus.

How many comparisons are asymptotically required **in the worst case** *when searching for an element in an array of length n? Choose the tightest bound for a best known algorithm in each case.*

Anzahl Vergleiche beim Suchen in **unsortiertem** Array/
Number comparisons when searching in an **unsorted** *array*:

◯ $O(\log n)$ ◯ $\Omega(\log n)$ ◯ $\Theta(\log n)$ ◯ $O(n)$ ◯ $\Omega(n)$ √ $\Theta(n)$

Anzahl Vergleiche beim Suchen in **sortiertem** Array/
Number comparisons when searching in a **sorted** *array*:

◯ $O(\log n)$ ◯ $\Omega(\log n)$ √ $\Theta(\log n)$ ◯ $O(n)$ ◯ $\Omega(n)$ ◯ $\Theta(n)$

/2P (b) Unter einer sehr grossen Anzahl (n) anwesender Studenten soll ein Preis vergeben werden. Der Student mit der Median Leginummer gewinnt. Es kommt zum Streit, welcher Algorithmus zur Ermittlung des Medians verwendet werden soll. Kreuzen Sie die richtigen Aussagen an.

Among a huge number (n) of students present, a price will be awarded to the student with the median Legi number. There is an argument what kind of algorithm shall be used to find this student. Mark the correct statements.

Um im schlechtesten Fall eine Laufzeit von $O(n \log n)$ zu erhalten, verwenden wir /
In order to have a worst case runtime of $O(n \log n)$, we use

◯ BubbleSort
◯ Sortieren durch Auswählen/*Selection Sort*
√ Mergesort
◯ Quicksort

Wir verwenden Quickselect mit zufälliger Auswahl des Pivots. Damit haben wir/
We use Quickselect with random pivot choice. Then we have

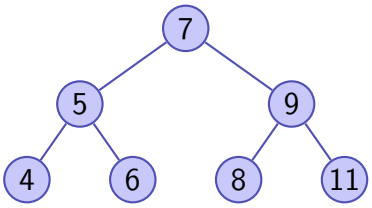
◯ im schlechtesten Fall eine Laufzeit von $O(n \log n)$/
a worst case running time of $O(n \log n)$

◯ im schlechtesten Fall eine Laufzeit von $O(n)$/*a worst case running time of* $O(n)$

◯ eine erwartete Laufzeit von $O(\log n)$/*an expected running time of* $O(\log n)$

√ eine erwartete Laufzeit von $O(n)$/*an expected running time of* $O(n)$

(c) Gegeben sei folgender binärer Suchbaum. *The following binary search tree is given.* /2P



Geben Sie an, welche Traversierungsart allenfalls zur angegeben Zahlensequenz passt.

Mark which traversal (if any) corresponds to the given number sequence.

$$7, 5, 9, 4, 6, 8, 11$$

◯ Hauptreihenfolge/*preorder* ◯ Nebenreihenfolge/*postorder*
◯ Symmetrische Reihenfolge/*inorder* √ Keine davon/*none of them*

$$7, 5, 4, 6, 9, 8, 11$$

√ Hauptreihenfolge/*preorder* ◯ Nebenreihenfolge/*postorder*
◯ Symmetrische Reihenfolge/*inorder* ◯ Keine davon/*none of them*

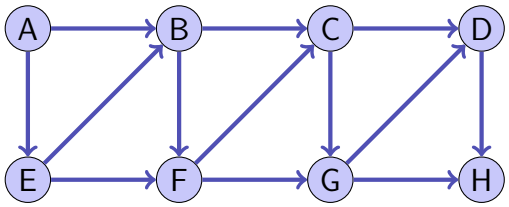
(d) Der folgende Array soll einen MinHeap speichern. Markieren Sie die beiden Elemente, welche aktuell die Heap-Eigenschaft verletzen.

The following array should store a MinHeap. Mark the two elements which currently violate the heap property. /4P

2	4	5	**11**	6	8	10	12	**9**	7	14	15	13
1	2	3	4	5	6	7	8	9	10	11	12	13

/2P (e) Geben Sie zu folgendem gerichteten Graphen eine topologische Sortierung der Knoten an.

Provide a topological sorting of the nodes of the following directed graph.



Topologische Sortierung/*Topological sorting*:

A E B F C G D H

/3P (f) Wir betrachten eine Hashtabelle der Länge $n = 7$. Angenommen den Buchstaben 'a', 'b', ... werden die Werte $1, 2, \ldots$ wie in folgender Tabelle zugeordnet.

We consider a hash table with length $n = 7$. Assume the characters 'a', 'b', ... correspond to values $1, 2, \ldots$ as provided with the following table.

c	$v(c)$	c	$v(c)$	c	$v(c)$	c	$v(c)$
		g	7	n	14	u	21
a	1	h	8	o	15	v	22
b	2	i	9	p	16	w	23
c	3	j	10	q	17	x	24
d	4	k	11	r	18	y	25
e	5	l	12	s	19	z	26
f	6	m	13	t	20		

Zu einem Namen $s = (s_i)_{i=1..k}$ wird der Hashwert wie folgt berechnet:

To a name $s = (s_i)_{i=1..k}$ we assign the following hash value

$$h(s) = \left(\sum_{i=1}^{k} v(s_i)\right) \bmod n$$

Beispiel/*Example*: $h('abc') = (1 + 2 + 3) \bmod 7 = 6$.

Tragen Sie die folgenden Namen (Schlüssel) in die Hashtablle unten ein. Es wird offen addressiert und linear (nach rechts) sondiert. In die Hashtabelle wurden vorgängig bereits die Namen 'inf', 'int' und 'jav' eingefügt.

Enter the following names (keys) into the hash table below. We use open addressing and linear probing (to the right). Previously the names 'inf', 'int' and 'jav' have been inserted into the hash table.

Einzufügende Namen/*Names to be inserted*: 'the', 'eth', 'big'

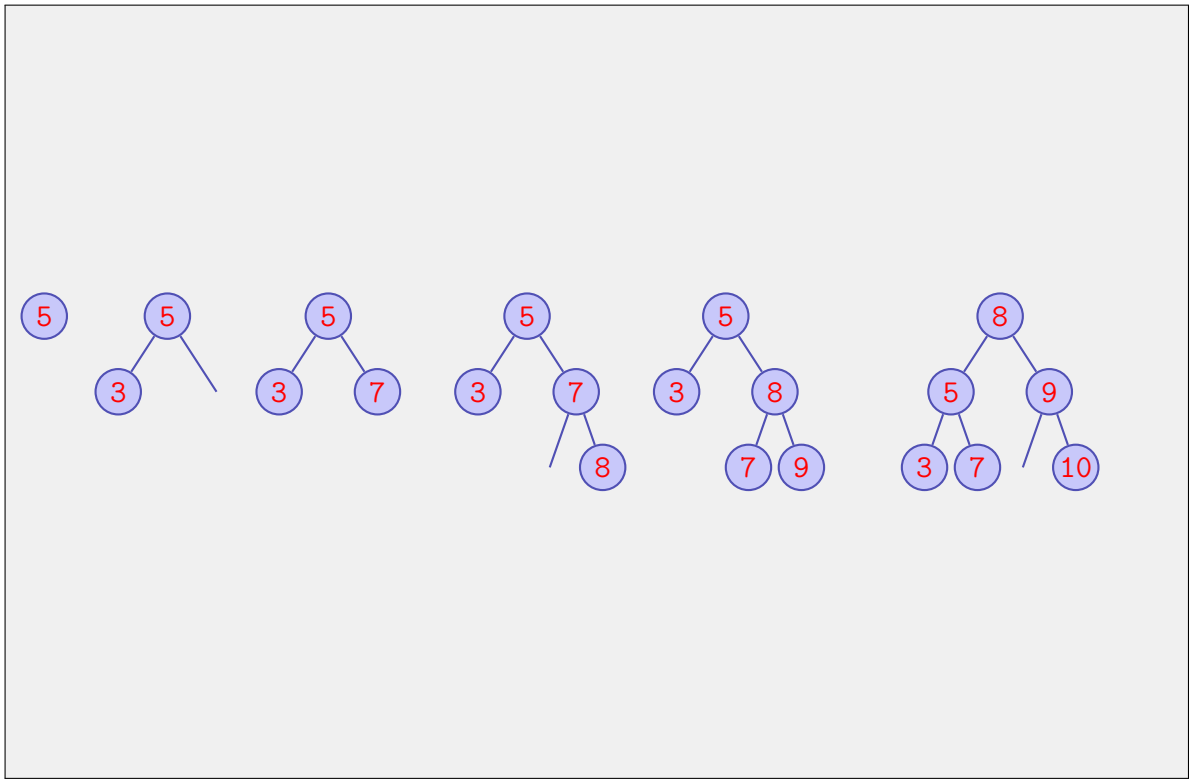
eth	inf	int		**big**	jav	**the**
0	1	2	3	4	5	6

(g) Fügen Sie folgende Zahlen in der angegebenen Reihenfolge in einen initial leeren AVL-Baum ein. Zeichnen Sie den Baum bei jedem Hinzufügen neu.

Insert the following numbers, in the given order, into an initially empty AVL Tree. Redraw the tree for each insertion.

/4P

5 3 7 8 9 10



Aufgabe 2: Asymptotik (15P)

(a) Finden Sie aus der Liste der in der weissen Box angebenen Funktionen jeweils diejenige, so dass die Gleichheit gilt.
Beispiel: $\Theta(n+5) = \Theta(\boxed{n})$

Find from the white boxed list of provided functions for each case below the function such that the equality holds.
Example: $\Theta(n+5) = \Theta(\boxed{n})$

/3P

$1, \quad \log n, \quad \log^2 n, \quad n, \quad n \log n, \quad n^2, \quad n^3, \quad n^2 \log n, \quad n \log^2 n, \quad n!, \quad n^n$

$\Theta(\Sigma_{i=0}^{n} i^2) = \Theta(\boxed{n^3})$

$\Theta(\log(n \cdot n^n)) = \Theta(\boxed{n \log n})$

$\Theta(|\sin(n)|) = \Theta(\boxed{1})$

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

Consider the following recursion:

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

Geben Sie eine geschlossene (nicht rekursive), einfache Formel für $T(n)$ an und beweisen Sie diese mittels vollständiger Induktion. Gehen Sie davon aus, dass n eine Potenz von 3 ist.

Provide a closed (non-recursive), simple formula for $T(n)$ and prove it using mathematical induction. Assume that n is a power of 3.

$$\begin{aligned} T(3^k) &= 3(T(3^{k-1}) + 3^{k-1}) \\ &= 3T(3^{k-1}) + 3^k = 3^2T(3^{k-2}) + 3^k + 3^k = ... \\ &= 3^kT(1) + \underbrace{3^k + ... + 3^k}_{k} \end{aligned}$$

`Assumption:` $T(n) = n \log_3 n$

`Induktion:`

`1.Hypothesis` $T(n) = f(n) := n \log_3 n.$

`2.Base Case` $T(1) = 0 = 1 \log_3 1 = f(1).$

`3.Step` $T(n) = f(n) \longrightarrow T(3 \cdot n) = f(3n)$ ($n = 3^k$ `for some` $k \in \mathbb{N}$):

$$\begin{aligned} T(3n) &= 3T(n) + 3n \\ &\overset{i.h.}{=} 3f(n) + 3n = 3(n \log_3 n) + 3n \\ &= 3n(1 + \log_3 n) \\ &= (3n) \log_3(3n) = f(3n) \end{aligned}$$

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.

(c) /1P

```
void g(int n){
    int s = 1;
    for (int i = 0; i<n ; ++i ){
        if (i==s){
            f();
            s += s;
        }
    }
}
```

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

$\Theta(\log n)$

(d) /2P

```
void g(int n){
    if (n > 1){
        g(n/2);
        g(n/2);
    }
    while (--n > 0){
        f();
    }
}
```

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

$\Theta(n \log(n))$

(e) /3P

```
void g(int n){
  for (int k = 1; k < n ; ++k){
    // call f for all bits that toggle
    // from k-1 to k:
    int prev = k-1;
    for (int num = k; num > 0; num /= 2){
        if (num % 2 != prev % 2){
            f()
        }
        prev /= 2;
    }
  }
}
```

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

$\Theta(n)$ [counter bit switches amortized analysis from class]

Aufgabe 3: Dynamic Programming (18P)

Gegeben sei ein Ausdruck

Consider an expression

$$a_1 \langle s_1 \rangle a_2 \langle s_2 \rangle \ \ldots \ \langle s_{n-1} \rangle a_n$$

bestehend aus n positiven Zahlen a_i und $n-1$ Operatoren s_i $(1 \leq i \leq n)$. Erlaubte Operatoren sind $+$ (Addition) und $\times$ (Multiplikation).

consisting of n positive numbers a_i und $n-1$ operators s_i $(1 \leq i \leq n)$. Permitted operators are $+$ (addition) and $\times$ (multiplication).

Beispiel/*Example*

$$\underset{a_1}{2} \underset{s_1}{+} \underset{a_2}{3} \underset{s_2}{\times} 0 + 2 \times 3$$

Man kann den Wert des Ausdrucks ändern, indem man Klammern einfügt.

You can change the value of the expression by insertion of parentheses.

Beispiele/*Examples*

$$(2 + (3 \times 0)) + (2 \times 3) = 8$$
$$(((2 + 3) \times 0) + 2) \times 3 = 6$$
$$((2 + 3) \times (0 + 2)) \times 3 = 30$$
$$(2 + (3 \times (0 + 2)) \times 3 = 24$$

Gesucht ist eine Klammerung, mit welcher der Ausdruck den kleinsten Wert ergibt (hier: 6).

Wanted is an insertion of parentheses that yields the smallest possible value (here: 6).

Geben Sie einen Algorithmus an, der nach dem Prinzip der dynamischen Programmierung arbeitet und den kleinstmöglichen Wert berechnet.
(Der triviale Algorithmus, welcher alle möglichen Lösungen auflistet, ergibt keinen Punkt).

Provide a dynamic programming algorithm for computing the smallest possible value. (The trivial algorithm that simply enumerates all possible solutions does not give any points.)

/8P

(a) (I)

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

What is the meaning of a table entry and what is the size of the (two-dimensional) DP-table T?

Tabellengrösse / *table size*:

$n \times n$ Table A (only entries in upper right triangle are used)

Bedeutung Eintrag / *entry meaning*:

Entry at row i, column j: smallest possible value of the sub-expression $a_i \langle s_i \rangle a_{i+1} \langle s_{i+1} \rangle \ldots \langle s_{j-1} \rangle a_j$.

(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?

$$A_{i,i} = a_i, 1 \leq i \leq n$$

$$A_{i,j} = \min_{i<k\leq j} \{A_{i,k-1} \langle s_{k-1} \rangle A_{k,j}\}, \quad 1 \leq i < j \leq n$$

(III)

In welcher Reihenfolge können die Einträge berechnet werden?

In which order can the entries be computed?

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

(IV) Wie kann der minimale Wert aus der Tabelle erhalten werden?

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

at $A[1, n]$

/3P (b) Geben Sie die Tabelle für den Beispielausdruck an.

Provide the table for the example expression.

$$2 + 3 \times 0 + 2 \times 3$$

	+		×		+		×	
2		5		0		2		6
		3		0		2		6
				0		2		6
						2		6
								3

/4P (c) Zusätzlich zum kleinsten Wert wollen Sie auch angeben, wie geklammert werden muss. Beschreiben Sie detailiert, wie Sie das bewerkstelligen.

In addition to the minimal value you want to provide the parentheses used. Describe in detail how this is done.

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 $A_{i,j} = A_{i,k} \langle s_{k-1}\rangle A_{i,j}$

/3P (d) Geben Sie die Laufzeiten für die Berechnung des minimalen Wertes und der dazugehörigen Klammerung in Θ-Notation möglichst knapp mit Begründung an.

Provide the run-times for the computation of the minimal value and the corresponding bracketing in Θ notation as tight as possible with short explanation.

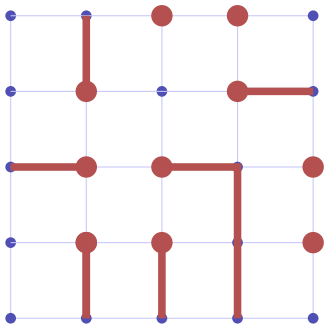
Starting from the diagonal we need to compute $(n-1)+(n-1)\cdot 2+(n-2)\cdot 3+\cdots+(n-(n-1))\cdot(n-1)$ pairs of expressions, that is $\sum_{i=1}^{n-1} i\cdot(n-i) = n\cdot n\cdot(n-1)/2-\sum_{i=1}^{n-1} i^2 = n\cdot n\cdot(n-1)/2-(n-1)\cdot n\cdot(2n-1)/6 = \Theta(n^3)$
The walk back for the best bracketing: 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.

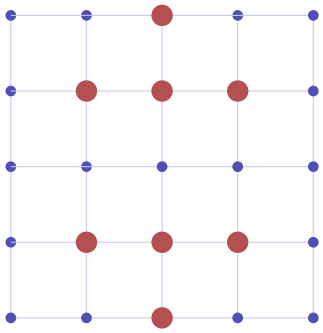
Diese Seite wurde bewusst freigelassen. *This page has been left blank intentionally*

Aufgabe 4: Flussprobleme (15P)

Gegeben sei ein $n \times n$ Gitter – ein *ungerichteter* Graph (V, E) der aus n Zeilen und n Spalten von Knoten $V = \{(i, j) : 1 \leq i, j \leq n\}$ besteht. Horizontal oder vertikal benachbarte Knoten sind mit einer Kante verbunden. Es seien $m < n$ Knoten als Startpunkte markiert. Das *Escape-Problem* besteht darin, festzustellen, ob m **knotendisjunkte** Pfade von den m Startpunkten zu paarweise disjunkten Randpunkten des Gitters existieren.
Nachfolgend sehen Sie auf der linken Seite eine Situation mit Fluchtmöglichkeit, während auf der rechten Seite keine m knotendisjunkte Pfade zum Rand existieren.

*Consider an $n \times n$ grid – an undirected graph (V, E) consisting of n rows and n columns of nodes $V = \{(i, j) : 1 \leq i, j \leq n\}$. Nodes neighbouring in horizontal or vertical direction are connected with an edge. There are $m \leq n$ nodes marked as starting points. The Escape Problem is to determine if there are m **vertex-disjoint** paths from the m starting points to pairweise different points on the boundary of the grid.*
Below on the left figure there is an escape. On the right figure there are no m vertex-disjoint paths possible.

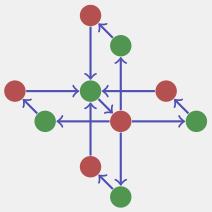




/8P (a) Modellieren Sie obiges Problem als Flussproblem. Beschreiben Sie die Konstruktion des Flussgraphen. Sie können zur Illustration einen (Ausschnitt des) Flussgraphen zeichnen.

Model the problem described above as a flow problem. Describe the construction of the flow-graph. You may draw (a part of) the flow-graph in order to illustrate your ideas.

1. make the graph directed by replacing all edge from E by two directed edges
2. replace each node $n \in V$ by an in- and an out-node n_i and n_o. Connect n_i to n_o by a directed edge and connect the in-node with incoming nodes and the out-node with outgoing nodes.
3. Add nodes s and a node t, connect s to all marked nodes and connect all boundary nodes to t.
4. provide capacity 1 to each edge.



(b) Geben Sie einen möglichst effizienten Algorithmus an, mit dem Sie bestimmen können, ob es m knotendisjunkte Pfade von den Startpunkten zu m beliebigen paarweise verschiedenen Randpunkten gibt.

Specify an efficient algorithm that can be used in order to determine, if there are m vertex-disjoint paths from the starting points to m pairwise different points on the boundary.

/2P

Use the Ford-Fulkerson Algorithm to determine the maximum flow from s to t. If the max-flow is m, then an escape path is found.

(c) Geben Sie die Laufzeit Ihres Algorithmus (im schlechtesten Fall) in Abhängigkeit von der Seitenlänge n des Gitters an.

Provide the running time of your algorithm (in the worst case) as a function of the side-length n of the grid.

/5P

Cost of the construction: $\Theta(n^2)$ for splitting the nodes and $\Theta(n^2)$ for the edges.
Running time of the Edmonds-Karp algorithm is $O(V \cdot E^2)$.
We have $O(E) = O(V) = O(n^2)$ and thus the running time is bounded by $O(n^6)$. However, a tighter estimation can be reached by considering the actual max flow that can happen, which is bounded by the $4n - 4$ nodes on the boundary. The running time of the Ford-Fulkerson Method is thus bounded by $O(E \cdot f^*) = O(n^2 \cdot (4n-4)) = O(n^3)$.

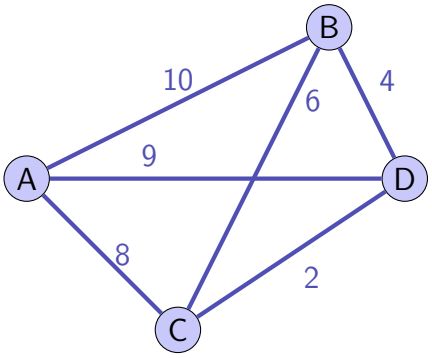
Aufgabe 5: Minimale Spannbäume (14P)

Zur Erinnerung: ein Minimaler Spannbaum minimiert die Summe der Kantengewichte unter allen möglichen Spannbäumen.
Wir nennen im Folgenden die Kante eines Spannbaumes, welche grösstes Gewicht, hat eine *Bottleneck-Kante*. Ein Minimaler Bottleneck-Spannbaum ist ein Spannbaum, dessen Bottleneck-Kante unter allen Spannbäumen minimal ist (so dass also kein Spannbaum existiert mit einer Bottleneck-Kante kleineren Gewichts).

Reminder: a Minimum Spanning Tree provides the minimal sum of edge weights for all possible spanning trees.
In the following we call the highest weighted edge of a spanning tree a bottleneck edge. A spanning tree is a minimum bottleneck spanning tree if the graph does not contain a spanning tree with smaller bottleneck edge weight.

/4P (a) Zeigen oder widerlegen Sie folgende Aussage: Jeder Minimum Bottleneck Spannbaum ist ein Minimaler Spannbaum. Als Hinweis schenken wir Ihnen folgenden Beispielgraphen.

Show or disprove the following statement: every minimum bottleneck spanning tree is a minimum spanning tree. As a hint we give you the following example graph.



In order to reach node A every minimum bottleneck spanning tree must be such that it contains the edge with weight 8. The tree with edges AC, CB and BD is a minimum bottleneck spanning tree, but not a minimum spanning tree (which would be AC,CB and CD which disproves the statement.

(b) Zeigen Sie folgende Aussage: Jeder Minimale Spannbaum ist ein Minimaler Bottleneck Spannbaum.
Hinweis: Verwenden Sie die Kreisregel (Wenn das Gewicht einer Kante e in einem Kreis C grösser ist als das jeder anderen Kante von C, dann kann e zu keinem minimalen Spannbaum gehören).

Show that the following statement holds: Every minimal spanning tree is a minimum bottleneck spanning tree.
Hint: Use the cycle property (If the weight of an edge e of a cycle C is larger than the weights of all other edges of C, then e cannot belong to a minimum spanning tree).

/6P

We prove this statement by contradiction using a cut-and-paste argument. Consider a minimum spanning tree T with largest weight edge t and a minimum bottleneck spanning tree B with largest weight edge b. If T is not a MBST then $w(t) > w(b)$. If we add t to B then we get a cycle C with strictly largest weight edge t. Thus t cannot be in any MST => T is not a MST.

(c) Erklären Sie kurz einen Algorithmus zum Finden eines Minimum-Bottleneck Spannbaums für einen (ungerichteten) Graphen (V, E). Welche asymtptotische Laufzeit hat der Algorithmus in Abhängigkeit von Anzahl Knoten $|V|$ und Kanten $|E|$?

Shortly explain an algorithm to find a minimum bottleneck spanning tree for an (undirected) graph (V, E). What is the asymptotic runtime of this algorithm as a function of the number of nodes and edges?

/4P

Because every MST is a MBST, we can use Kruskal's or Prim's algorithm.
Kruskal's algorithm adds edges sorted by weight, edges that do not form a cycle are accepted and edges of a cycle are rejected. In order to identify cycles, a Union Find data structure can be used.
The runtime of Kruskals algorithm is $O(E \log E) = O(E \log V)$.
Alternatively Prim's algorithm can be employed with a runtime of $O(E \log V)$ or (when using Fibonacci Heaps) $O(E + V \log V)$.

Aufgabe 6: Parallele Programmierung (17P)

/4P (a) Hannes hat sein sequentielles Programm parallelisiert und macht folgende Laufzeitmessungen. Was ist der sequentielle Anteil seines Programmes, wenn man das Amdahlsche Gesetz für den vorliegenden Fall als gültig ansieht?

Hannes has parallelized his sequential program and makes the following runtime measurements. What is the sequential part of his program if Amdahl's law is assumed to hold for this case?

Anzahl Prozessoren *number processors*	Problemgrösse *problem size*	Laufzeit *running time*
$p = 1$	$n = 1000$	$t = 500$ ms
$p = 7$	$n = 1000$	$t = 200$ ms

Observed Speedup:

$$S = 500/200 = 2.5$$

Speedup according to Amdahl

$$S_A = \frac{1}{\lambda + (1-\lambda)/p} = 2.5$$

with $p = 7$ we have

$$\begin{aligned} 2.5 &= \frac{1}{\lambda + (1-\lambda)/7} \\ \lambda + (1-\lambda)/7 &= 4/10 \\ 6\lambda &= 28/10 - 1 \\ \lambda &= 3/10 = 0.3. \end{aligned}$$

/3P (b) Beschreiben Sie kurz Amdahls und Gustavsons Gesetz und erklären Sie den scheinbaren Widerspruch ganz kurz.

Shortly describe Amdahls and Gustavsons Law and explain the apparent contradiction very shortly.

Both laws are models of the speedup for parallelisation. Amdahls law assumes a relative sequential portion while Gustavson assumes a fixed absolute sequential part. It depends on the problem at hand, which of the models are more appropriate.

Im nun folgenden Teil der Aufgabe werden jeweils zwei Threads ausgeführt, einer mit Funktion `A` und einer mit `B`. Die Funktion `print` gibt jeweils einen Buchstaben aus (Thread-sicher). Das gesamte Programm sieht etwa so aus:

The following parts of the task are each executed by two threads, one of them executes `A` and the other `B`. The function `print` outputs a character (threads-safe). The overal program looks like this:

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

void A(){
    ... // executed as thread 1
}

void B(){
    ... // executed as thread 2
}

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

Geben Sie zu folgenden Funktionen `A` und `B` jeweils alle theoretisch möglichen Ausgaben an (allenfalls getrennt durch Semikolon). Wenn eine leere Ausgabe (“”) möglich ist, geben Sie das auch an.
Bestimmen Sie, ob das zugehörige Programm terminiert. Wenn das Programm nicht terminiert, geben Sie die Ausgabe so weit an, wie sie auftreten kann.

Provide all (theoretically) possible outputs (separated by semicolons) to the following functions `A` and `B`. Also provide an empty output (“”) if that can occur. Specify if the corresponding program would terminate. If the program does not terminate, provide all possible outputs as far as they can occur.

(c)

```
void A(){
    print('A');
    print('B');
}
```

```
void B(){
    print('X');
    print('Y');
}
```

/2P

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

ABXY ; AXBY ; AXYB ; XABY ; XAYB; XYAB

Das Programm terminiert/*the program terminates*

√ immer/*always* ◯ nie/*never* ◯ weder immer noch nie/*neither always nor never*

/2P (d)

```
std::mutex m; // global variable used in both threads
```

```
void A(){
    m.lock();
    print('A');
    print('B');
    m.unlock();
}
```

```
void B(){
    m.lock();
    print('X');
    print('Y');
    m.unlock();
}
```

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

```
ABXY ; XYAB
```

Das Programm terminiert/*the program terminates*

√ immer/*always* ◯ nie/*never* ◯ weder immer noch nie/*neither always nor never*

/2P (e)

```
std::mutex m; // global variables
std::mutex m2; // used in both threads
```

```
void A(){
    m.lock();
    m2.lock();
    print('A');
    print('B');
    m2.unlock();
    m.unlock();
}
```

```
void B(){
    m2.lock();
    m.lock();
    print('X');
    print('Y');
    m.unlock();
    m2.unlock();
}
```

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

```
ABXY ; XYAB ; ''
```

Das Programm terminiert/*the program terminates*

◯ immer/*always* ◯ nie/*never* √ weder immer noch nie/*neither always nor never*

(f)

/2P

```
std::mutex m; // global variables used by both threads
int count = 0; // global counter

void barrier(){ // used by both threads
    m.lock();
    count ++;
    while (count < 2);
    m.unlock();
}
```

```
void A(){
    print('A');
    barrier();
    print('B');
}
```

```
void B(){
    print('X');
    barrier();
    print('Y');
}
```

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

AX; XA

Das Programm terminiert/*the program terminates*

◯ immer/*always* √ nie/*never* ◯ weder immer noch nie/*neither always nor never*

(g)

/2P

```
std::mutex m; // global variables used by both threads
int count = 0; // global counter

void barrier(){ // used by both threads
    count++;
    m.lock();
    while (count < 2);
    m.unlock();
}
```

```
void A(){
    print('A');
    barrier();
    print('B');
}
```

```
void B(){
    print('X');
    barrier();
    print('Y');
}
```

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

AXBY; XABY; AXYB; XAYB; AX; XA [mainly count++ interleaving, but also data race possible]

Das Programm terminiert/*the program terminates*

◯ immer/*always* ◯ nie/*never* √ weder immer noch nie/*neither always nor never*

Aufgabe 7: Parallele Programmierung (9P)

Ein Semaphor S ist ein Datentyp, welcher den Zugriff zu einer gemeinsamen Ressource steuert. Er speichert einen ganzzahligen Wert v, welcher zu Beginn angibt, wie viele Threads gleichzeitig Zugriff auf die Ressource haben.
Der Semaphor bietet zwei Operationen an: P (niederländisch 'proberen') zum Zugriff und V (niederländisch 'vrijgeven') zum Freigeben der Ressource.
Ruft ein Thread $s.P$ auf, so muss er warten, solange der Wert von $s.v$ kleiner gleich null ist. Sobald ein Thread weiterfahren darf, dekrementiert er den Wert von $s.v$. Ruft ein Thread $s.V$ auf, so wird der Wert von $s.v$ inkrementiert. Ein allenfalls wartender Thread darf dann weiterfahren.
Semantik im Pseudocode (Anweisungsblock in eckigen Klammern wir atomar ausgeführt, also zu einer Zeit nur von einem Thread):

A semaphore S is a data type to control access to a common resource. It stores an integer value v that initially specifies the number of threads that can simultaneously access the resource.
The semaphore offers two operations: P (Dutch 'proberen') for accessing and V (Dutch 'vrijgeven') for releasing the resource.
When a thread calls $s.P$ then it must wait while the value of $s.v$ is smaller or equal to zero. As soon as a thread can continue running, it decrements the value of $s.v$. When a thread calls $s.V$ then the value of $s.v$ is incremented. A waiting thread (if any) can then continue executing.
Semantics in pseudo-code (the statement block in square brackets is executed atomically, i.e. only by one thread at a time):

```
P(){
  while(true){
    [ if (v>0) { v-- ; exit } ]
  }
}
```

```
V(){
  [ v++ ; ]
}
```

/3P (a) Angenommen der Datentyp `Semaphore` der rechten Seite sei bereits korrekt implementiert. Vervollständigen Sie folgenden Code so, dass `lock` und `unlock` von `MyMutex` funktionieren wie das lock und unlock einer Mutex.

Assume the data type `Semaphore` on the right hand side is correctly implemented. Complement the following code such that `lock` and `unlock` work like the lock and unlock of a mutex.

```
class MyMutex{
    Semaphore s;
public:
    MyMutex(): s(1) {};

    void lock(){ s.P(); }

    void unlock(){ s.V(); }
};
```

(b) Vervollständigen Sie nun nachfolgenden Code so, dass obige Semantik des Semaphors effizient implementiert wird.

Complement the following code such that the semantics of the semaphore from above are implemented efficiently.

/6P

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


class Semaphore {

  std::mutex m;
  using guard = std::unique_lock<std::mutex>;
  std::condition_variable cond;
  int num;

  public:
    Semaphore(int n):num(n){}

    void P(){
        guard g(m);
        cond.wait(g, [&]{return num > 0;});
        --num;
    }

    void V(){
        guard g(m);
        ++num;
        cond.notify_one();
    }

};
```