Datenstrukturen und Algorithmen Beispielprüfung, Lösung 7/2017

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 Schrift grösse.
3. Benützen Sie einen Kugelschreiber (blau oder schw keinen Bleistift. Bitte schreiben Sie leserlich. Resultate werden bewertet.
4. Lösungen sind direkt auf das Aufgabenb vorgesehenen Boxen zu schreiben (und mehr Platz benötigt wird). Ungülti lich durchzustreichen! Korrekture gaben bitte unmissverständlich
5. Es gibt keine Negativpunk
6. Störungen durch irge Sie bitte sofort der
7. Wir sammeln d Sie unbedingt s Assistenten eingez und lassen Sie Ihre P Dasselbe gilt, wenn Sie el- den Sie sich lautlos, und Vor- zeitige Abgaben sind nur bis ngsende möglich.
8. Wenn Sie zur Toilette müssen, mel 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:

n: 120 minutes.

ation aids: dictionary (for spoken lan- hand written or $\geq$ 11pt font size

not a pencil. Please write solutions that we can

directly onto the exam boxes (and directly below, if). Invalid solutions need to be Provide corrections to answers of questions without any ambiguity!

negative points for false answers.

eel disturbed by anyone or anything, let the visor of the exam know immediately.

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.

If you need to go to the toilet, raise your hand and wait for a supervisor.

We will not answer any content-related questions during the exam. Please write comments referring to the tasks on the exam sheets.

Beispielprüfung als Richtwert für den Stil der Prüfung im August. Die Aufgaben entstammen grösstenteils alten Prüfungen. / *Example exam as a rough guideline for the style of the exam in August. The tasks originate largely from old exams*

Question:	1	2	3	4	5	6	Total
Points:	8	7	10	9	7	7	48
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. (8P)

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) Führen Sie auf dem folgenden Array zwei Iterationen des Algorithmus *Sortieren durch Einfügen* aus. Das zu sortierende Array ist durch vorherige Iterationen bereits bis zum Doppelstrich sortiert worden.

Execute on the following array two iterations of the algorithm Insertion Sort. The array has been sorted by previous iterations up to the double vertical bars.

3	7	10	15 ‖	12	6	1	5	2	13

3	7	10	12	15 ‖	6	1	5	2	13

3	6	7	10	12	15 ‖	1	5	2	13

/1P (b) Fügen Sie die unten dargestellten Schlüssel in der gegebenen Reihenfolge in die untenstehende Hashtabelle ein. Benutzen Sie Double-Hashing mit der Hashfunktion $h(k) = k \bmod 11$ und benutzen Sie $h'(k) = 1 + (k \bmod 9)$ zur Sondierung. Die Sondierung läuft immer nach links.

Enter the following keys in the given order into the hash-table displayed below. Use double hashing with the hash function $h(k) = k \bmod 11$ and use $h'(k) = 1 + (k \bmod 9)$ for probing. Probing always goes to the left.

Schlüssel / *Keys*: 9, 11, 17, 25, 31, 20

11			25	31		17		20	9	
0	1	2	3	4	5	6	7	8	9	10

(c) Das untenstehende Array enthält Elemente eines Min-Heaps in der üblichen Form gespeichert. Wie sieht das Array aus, nachdem das Minimum entfernt wurde und die Heap-Bedingung wieder hergestellt wurde?

The array below contains elements of a Min-Heap stored in the usual form. What does the array look like after the minimum has been removed and the heap-condition has been re-established?

/1P

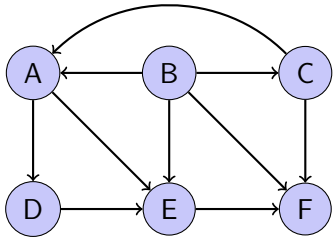
3	5	9	22	11	14	13	32	50	20

5	11	9	22	20	14	13	32	50	--

(d) Geben Sie die topologische Sortierung des untenstehenden Graphen an.

Provide the topological sorting of the graph displayed below.

/1P



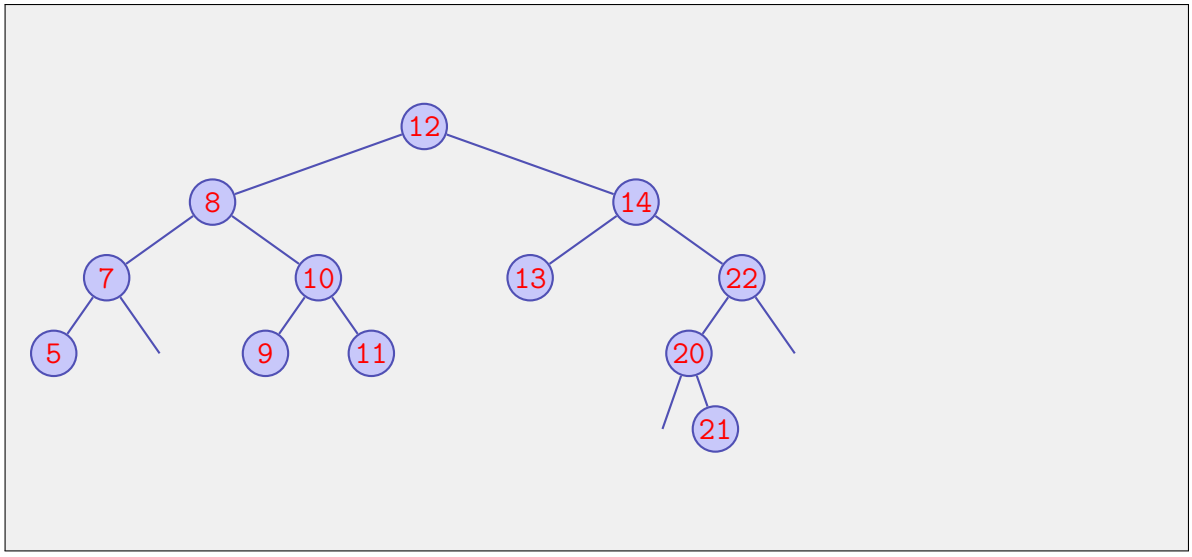
Top. Sortierung / *Top. Sorting* B , C , A , D , E , F ,

(e) Zeichnen Sie den binären Suchbaum, dessen Post-Order-Traversierung die dargestellte Folge ergibt.

Draw the binary search tree that provides the following sequence when post-order-traversing is applied.

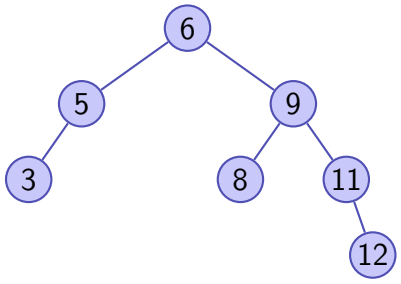
/1P

Folge / *Sequence*: 5, 7, 9 , 11, 10, 8, 13, 21, 20 , 22 , 14, 12

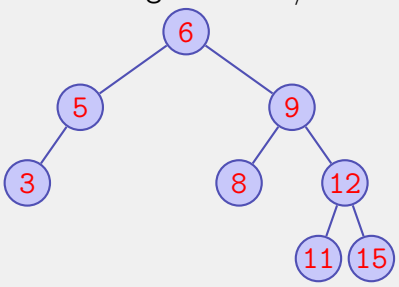


/1P (f) Fügen Sie in untenstehendem AVL-Baum zuerst den Schlüssel 15 ein und löschen Sie danach im entstandenen AVL-Baum den Schlüssel 3.

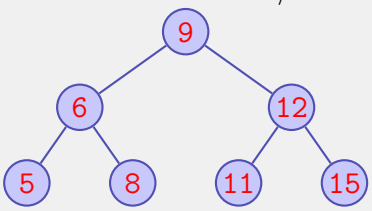
First insert into the AVL-Tree below the key 15. After insertion, delete the key 3 in from the AVL-Baum.



Nach Einfügen von 15 / *After insertion of* 15

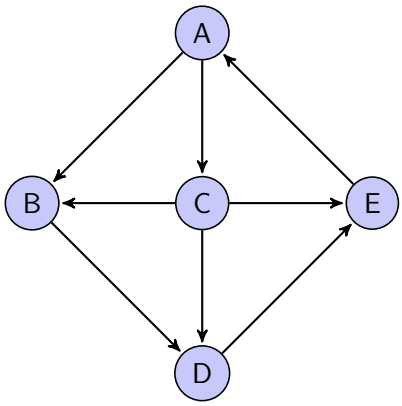


Nach Löschen von 3 / *After deletion of* 3



/1P (g) Geben Sie jeweils eine Reihenfolge an, in der die Knoten des folgenden Graphen von einer Breitensuche (BFS) bzw. von einer Tiefensuche (DFS) mit Startknoten A besucht werden, wenn die Nachbarn eines Knotens in alphabetischer Reihenfolge abgearbeitet werden.

Provide the order in which the nodes of the following graph are traversed from a breadth first search (BFS) and from a depth first search (DFS) with starting node A. Neighbours of a node are visited in alphabetic order.



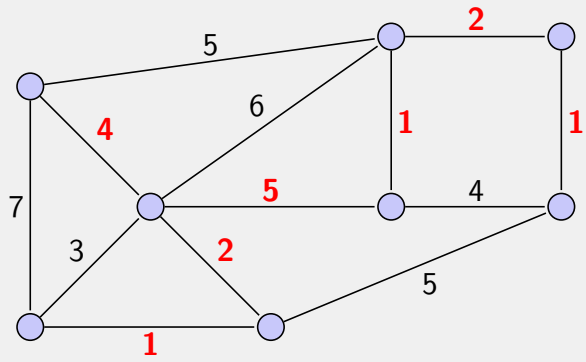
BFS: A B C D E

DFS: A B D E C

(h) Markieren Sie im untenstehenden gewichteten Graphen die Kanten eines minimalen Spannbaumes.

Mark in the weighted graph below the edges of a minimal spanning tree.

/1P



(the solution is not unique, either of the 5-edge can be chosen)

Aufgabe 2. (7P)

(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 for the following functions an order such that the following holds: If a function f is left of 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 $n^5 \in \mathcal{O}(n^7)$.

/1P

$$\frac{n}{\log n}, \quad 2^{1024}, \quad 3^{\sqrt{n}}, \quad \sum_{i=1}^{n} i/2, \quad 2^{3\log_2 n}, \quad n\sqrt{n}, \quad \log(n^{17})$$

$2^{1024}, \log(n^{17}), \frac{n}{\log n}, n\sqrt{n}, \sum_{i=1}^{n} i/2, 2^{3\log_2 n}, 3^{\sqrt{n}}$

/3P (b) Gegeben Sei die folgende Rekursionsgleichung:

Consider the following recursion:

$$T(n) = \begin{cases} 2 \cdot T(\frac{n}{2}) + 4n - 1, & n > 1 \\ 2 & n = 1 \end{cases}$$

Geben Sie eine geschlossene (nicht rekursive) Formel für $T(n)$ an und beweisen Sie diese mittels vollständiger Induktion.
Hinweise:

1. Sie können davon ausgehen, dass n eine Potenz von 2 ist.
2. Für $q \neq 1$ gilt $\sum_{i=0}^{k} q^i = \frac{q^{k+1}-1}{q-1}$.

Provide a closed (non-recursive) formula for $T(n)$ and prove it using mathematical induction.
Hints:

1. *You can assume that n is a power of 2.*
2. *For $q \neq q$ it holds that $\sum_{i=0}^{k} q^i = \frac{q^{k+1}-1}{q-1}$.*

$$\begin{aligned} T(n) &= 2T(n/2) + 4n - 1 \\ &= 2(2T(n/2^2) + 4\frac{n}{2} - 1) + 4n - 1 \\ &= 2^2 T(n/2^2) + 4n \cdot 2 - (1 + 2^1) \\ &= 2^3 T(n/2^3) + 4n \cdot 3 - (2^0 + 2^1 + 2^2) \\ &= \dots \\ &= 2^i T(n/2^i) + 4n \cdot i - \sum_{j=0}^{i-1} 2^j \\ &= 2^i T(n/2^i) + 4n \cdot i - 2^i + 1. \end{aligned}$$

with $i = \log_2 n$ we get

$$\begin{aligned} T(n) &= 2^i T(1) + 4n \cdot i - 2^i + 1. \\ &= 2 \cdot 2^i - 2^i + 4n \cdot i + 1 \\ &= n + 4n \log_2(n) + 1. \end{aligned}$$

Induktion:
Hypothese $T(n) = n + 4n \log_2(n) + 1$.

Anfang $T(1) = 1 + 1 = 2$.

Schritt $(n/2 \to n)$: $T(n) = 2T(n/2) + 4n - 1 = 2(n/2 + 4n/2 \log_2(n/2) + 1) + 4n - 1 = n + 4\log_2(n) - 4n + 2 + 4n - 1 = 4n \log_2(n) + 1$.

(c) Geben Sie die asymptotische Laufzeit des folgenden Algorithmus in Abhängigkeit von $n \in \mathbb{N}$ (so knapp wie möglich) in Θ-Notation an. Sie müssen Ihre Antwort nicht begründen.

Provide the asymptotic runtime of the following algorithm as a function of $n \in \mathbb{N}$ (as tight as possible) in Θ-Notation. You do not have to justify your answer.

/1P

```
int i = 1;
int j = 0;
while (j < n){
    j += i;
    i *= 2;
}
```

Laufzeit/*Runtime*

$\Theta(\log(n))$

(d) Geben Sie die asymptotische Laufzeit des folgenden Algorithmus in Abhängigkeit von $n \in \mathbb{N}$ (so knapp wie möglich) in Θ-Notation an. Sie müssen Ihre Antwort nicht begründen.

Provide the asymptotic runtime of the following algorithm as a function of $n \in \mathbb{N}$ (as tight as possible) in Θ-Notation. You do not have to justify your answer.

/1P

```
int r = 1;
for (int i = n; i>0;){
    if (i % 2 ==0){
        r *= r;
        i /= 2;
    }
    else
    {
        r *= 3;
        --i;
    }
}
```

Laufzeit/*Runtime*

$\Theta(\log(n))$

(e) Geben Sie die asymptotische Laufzeit des folgenden Algorithmus in Abhängigkeit von $n \in \mathbb{N}$ (so knapp wie möglich) in Θ-Notation an. Sie müssen Ihre Antwort nicht begründen.

Provide the asymptotic runtime of the following algorithm as a function of $n \in \mathbb{N}$ (as tight as possible) in Θ-Notation. You do not have to justify your answer.

/1P

```
for (int i = 1; i <= n; i *= 3){
    for (int j = n; j >= 4 * i; --j){
    }
}
```

Laufzeit/*Runtime*

$\Theta(n \log n)$

Aufgabe 3. (10P)

Ein Student möchte heute und morgen jeweils eine Portion Bratkartoffeln kochen und hat dazu n Kartoffeln $\{1, \ldots, n\}$ mit einem Gesamtgewicht von $G \in \mathbb{N}$ Gramm zur Verfügung. Die Kartoffel i wiegt $g_i \in \mathbb{N}$ Gramm. Es sollen nun alle n Kartoffeln so auf die zwei Portionen A und B verteilt werden, dass diese annähernd gleich schwer sind. Da die Portionen für unterschiedliche Tage bestimmt sind, muss eine Kartoffel entweder komplett für die Portion A oder komplett für die Portion B verwendet werden (jede Kartoffel muss entweder sofort benutzt werden oder vollständig für morgen aufbewahrt werden). Konkret suchen wir also zwei Mengen A und B von Kartoffeln mit $A \cap B = \emptyset$, $A \cup B = \{1, \ldots, n\}$, deren Gewichtsdifferenz minimal ist.

A student wants to cook fried potatoes, one portion today and one portion tomorrow. For this purpose he has n potatoes $\{1, \ldots, n\}$ at his disposal that have an overall weight of $G \in \mathbb{N}$ gram. The potato i weighs $g_i \in N$ grams. We want to partition all n potatoes into the portions A and B such that these portions provide roughly the same weight. Since the portions are planned to be cooked on different days, every potato must either be completely used for portion A, or completely used for portion B (every potato must either be used immediately, or must be completely saved for tomorrow). More concretely we want to compute two sets A and B of potatoes with $A \cap B = \emptyset$, $A \cup B = \{1, \ldots, n\}$ with a minimal weight difference.

/6P (a) Geben Sie einen Algorithmus an, der nach dem Prinzip der dynamischen Programmierung arbeitet und die kleinstmögliche Gewichtsdifferenz zweier Mengen A und B (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 weight difference of two sets A and B (as described above). Address the following aspects in your solution.
(The trivial algorithm that simply enumerates all possible solutions does not give any points.)

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

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

Tabellengrösse / *table size*: $(n+1) \times (\lfloor G/2 \rfloor + 1)$ `boolean entries`

Bedeutung Eintrag / *entry meaning*: $T[i, g] = true$ $(0 \leq i \leq n,\ 0 \leq g \leq \lfloor G/2 \rfloor)$ genau dann wenn Menge $K \subseteq \{1, \ldots, i\}$ existiert mit $\sum_{k \in K} g_k = g$.

(II)

Wie berechnet man einen Eintrag der Tabelle aus den vorher berechneten Einträgen?

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

$T[i,0] = true$ für jedes $i \in \{0,\dots,n\}$
$T[0,g] = false$ für jedes $g \in \{1,\dots,\lfloor G/2 \rfloor\}$
Für $1 \leq i \leq n$ und $g \in \{1,\dots,\lfloor G/2 \rfloor\}$:

$$T[i,g] = \begin{cases} T[i-1,g] & g < g_i \\ T[i-1,g] \vee T[i-1,g-g_i] & g \geq g_i \end{cases}$$

(III)

In welcher Reihenfolge können die Einträge berechnet werden?

In which order can the entries be computed?

$T[i,g]$ für aufsteigende Werte erst in i und für feste Werte von i aufsteigend in g.

(IV)

Wie kann der Wert einer minimalen Gewichtsdifferenz aus der Tabelle erhalten werden?

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

Ermittle das grösste g_{max} mit $T[n,g_{max}] = true$. Minimale Gewichtsdifferenz beträgt dann $G - 2g_{max}$ Gramm.

/2P (b) Beschreiben Sie detailliert, wie aus der DP-Tabelle abgelesen werden kann, welche Kartoffel an welchem Tag benutzt wird.

Describe in detail how you can recognize from the DP table which potato is used on which day.

Wenn g_{max} grösster Wert mit $T[n, g\max] = true$. Beginne mit $i = n$ und $g = g_{max}$: Prüfe, ob $T[i, g] = T[i-1, g]$. Falls ja, dann $i \leftarrow i - 1$. Sonst gilt $T[i, g] = T[i-1, g-g_i]$, i wird ausgegeben und $i \leftarrow i - 1$, $g \leftarrow g - g_i$. Wenn $i = 0$ dann fertig.

/2P (c) Geben Sie die Laufzeit des in (a) und (b) entwickelten Verfahrens an und begründen Sie Ihre Antwort. Ist die Laufzeit polynomiell?

Provide the running time of algorithm developed in (a) and in (b), and justify your answer. Is the running time polynomial?

Tabelle hat eine Grösse von $\Theta(nG)$. Jeder Eintrag wird in konstanter Zeit von vorigem Eintrag berechnet. Gesamtlaufzeit beträgt somit $\Theta(nG)$. Das ist pseudopolynomiell. Der Algorithmus as (b) terminiert nach $\Theta(n)$ Schritten. Jeder Schritt ist in konstanter Zeit ausführbar. Laufzeit beträgt also $\Theta(n)$.

Aufgabe 4. (9P)

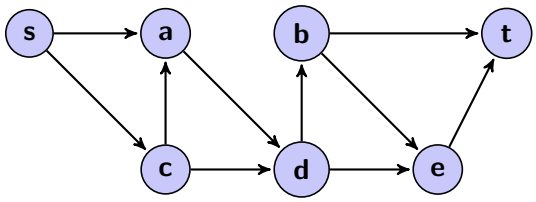
Zwei Computer sollen über ein Netz zuverlässig kommunizieren. Die Computer sind nicht unbedingt direkt miteinander verbunden, sondern schicken im Allgemeinen Pakete über verschiedene Zwischenstationen durchs Netz. Es soll sichergestellt werden, dass ein Computer dem anderen Computer auch bei einem Ausfall einer einzigen, beliebigen Verbindung im Netz Pakete schicken kann. Das Netz ist durch eine Menge V von Knoten und eine Menge E von (gerichteten) Verbindungen zwischen je zwei Knoten gegeben. Weiterhin sind zwei Knoten s und t in V gegeben, die miteinander kommunizieren sollen.

Two computers should communicate reliably over a network. The computers are not necessarily directly connected, but in general send packages over several intermediate stations through the network. We want to make sure that a computer can send packages to the other computer even if a single, arbitrary connection in the network fails. The network is given as a set V of vertices and a set E of (directed) connections between two vertices each. Moreover, two vertices s and t in V are given that should communicate with each other

(a) Wir wollen entscheiden, ob s auch dann noch Pakete an t schicken kann, wenn eine einzige, beliebige Verbindung in E unterbrochen wird. Modellieren Sie dazu das Problem als Flussproblem. Beschreiben Sie dazu die Konstruktion eines geeigneten Netzes $N_a = (V_a, E_a, c_a)$ und geben Sie an, welche Kapazitäten c_a die Kanten besitzen sollen. Wie kann aus dem Wert eines maximalen Flusses abgelesen werden, ob die Kommunikation von s nach t möglich ist, wenn eine einzige, beliebige Verbindung in E unterbrochen wird?
Beispiel: im dargestellten Netz kann der Knoten s auch dann noch Pakete an den Knoten t schicken, wenn eine einzige beliebige Kante entfernt wird.

/4P

We want to decide whether s is able to send packages to t even if a single, arbitrary connection in E is broken. For this purpose, model the above problem as a flow problem. Describe the construction of an appropriate network $N_a = (V_a, E_a, c_a)$, and describe which capacities c_a the edges should have.
How can you deduce from the value of a maximum flow if the communication from s to t is always possible if a single, arbitrary connection in E is broken?
Example: In the network depicted below, the vertex s can send packages to vertex t even if a single, arbitrary edge is removed.



Definieren Sie das Netzwerk auf der nächsten Seite, möglichst in Worten und nicht formal.

Define the network on the next page, if possible in words and not formally.

Definition des Netzwerkes (möglichst in Worten und nicht formal)

Definition of the network (if possible in words and not formally)

Knotenmenge V_a / *Node set* V_a Ein Knoten pro Computer

Kantenmenge E_a / *Edge set* E_a Eine Kante für jede gerichtete Verbindung von Computer zu Computer

Kapazitäten c_a / *Capacities* c_a Alle Kanten haben Kapazität 1

Die Kommunikation ist nach Unterbruch einer Verbindung noch möglich, wenn (mit formaler Begründung) / *The communication is possible even after a cut of one connection if ... (justify your answer formally)* maximaler Fluss von s nach t mindestens 2 beträgt. Nach dem Min-Cut/Max-Flow Theorem existiert genau dann kein $s-t$ Schnitt mit Kapazität kleiner als 2.

/5P (b) Jetzt wollen wir entscheiden, ob der Knoten s auch dann noch Pakete an den Knoten t schicken kann, wenn ein Knoten aus $V \setminus \{s,t\}$ defekt ist und nicht genutzt werden kann. Modellieren Sie dazu das Problem als Flussproblem und beschreiben Sie ein geeignetes Netz $N_b = (V_b, E_b, c_b)$. Wie kann aus dem Wert eines maximalen Flusses in diesem Netz abgelesen werden, ob die Kommunikation von s nach t möglich ist, wenn nur ein Knoten entfernt wird?

Now we want to decide whether the vertex s is able to send packages to t if a vertex in $V \setminus \{s,t\}$ is broken and cannot be used. For this purpose, model this problem as a flow problem and describe an appropriate network $N_b = (V_b, E_b, c_b)$. How can you deduce from the value of a maximum flow in this modified network if the communication from s to t is possible if only a single vertex is removed?

Beispiel: Im oben dargestellten Netz kann der Knoten s keine Pakete an den Knoten t schicken, wenn der Knoten d nicht genutzt werden kann.

Example: In the network depicted above, the vertex s cannot send a package to the vertex t if vertex d cannot be used.

Definition des Netzwerkes (möglichst in Worten und nicht formal)

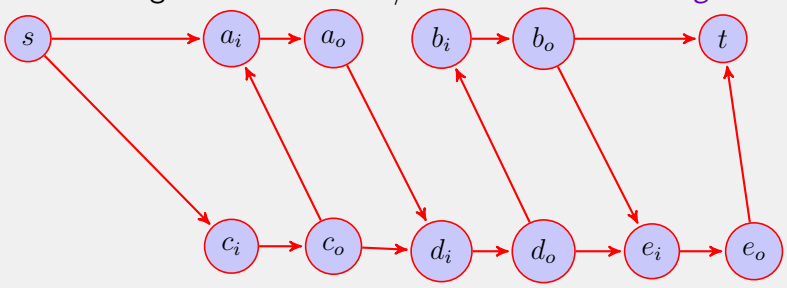
Definition of the network (if possible in words and not formally)

Knotenmenge V_a / *Node set V_a*Zwei Knoten pro Computer: ein Input Knoten, ein Output Knoten

Kantenmenge E_a / *Edge set E_a*Eine Kante für jede gerichtete Verbindung von Computer output zu Computer Input und innerhalb des Computers von Knoten Input zu Knoten Output.

Kapazitäten c_a / *Capacities c_a* Alle Kanten haben Kapazität 1

Schematische Darstellung des Netzes / *Schematic drawing of the network*



Die Kommunikation ist möglich, wenn ... / *The communication is possible if ...* der maximale Fluss grösser ist als Null.

Aufgabe 5: Parallel and Concurrent Programming (7P)

/2P (a) Ein single-threaded Programm läuft nicht schnell genug. Wir nehmen an, dass dieses Programm aus einem sequentiellen Teil (60% der Ausführungszeit) und einem parellelisierbaren Teil besteht (40% der Ausführungszeit). Sie haben zwei Optimierungsmöglichkeiten:
1. Optimierung des sequentiellen Teiles auf die Hälfte der ursprünglichen Ausführungszeit
2. Parallelisierung des parallelisierbaren Teiles (unter der Annahme, dass dieser perfekt skaliert)
Beide Ansätze sind gleich aufwändig. Wann würden Sie die zweite Option bevorzugen? Begründen Sie Ihre Antwort.

A single-threaded program you are developing is not fast enough. We assume that the program consists of a sequential part (that takes 60% of the execution time) and a parallelizable part (that takes the rest 40% of execution time). You have two options to make it faster:
1. optimize the sequential part so that its time is cut down to half
2. parallelize the parallel part (assuming that it scales perfectly)
Both approaches require the same effort. Under what conditions would you choose the second option? Justify your answer.

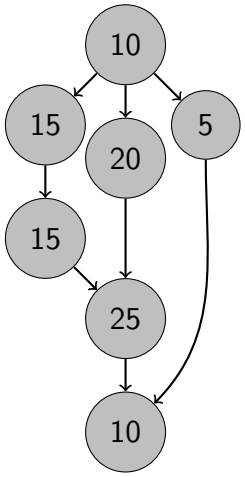
First option provides a speedup of $\frac{1}{0.7}$
Second option provide a speeup of

$$S_p = \frac{1}{0.6 + (0.4)/p}$$

Option 2 is to be preferred if $S_p > 1/0.7$, i.e. $0.6 + 0.4/p < 0.7$ i.e. when then number of processors available $p > 4$.

/1P (b) Folgende Abbildung zeigt den Task Graphen eines Algorithmus. Die Zahl an jedem Knoten bezeichnet die benötigte Ausführungszeit für den jeweiligen Berechnungsschritt.

The following figure shows the task graph for an algorithm. The number in each node denotes the execution time per task.



Wie gross ist der maximale Speedup der mit Parallelisierung erreicht werden kann, verglichen mit sequentieller Durchführung bei einer einzigen Ausführung des Algorithmus?

What is the maximum overall speedup that can be achieved by parallelism when the algorithm runs once compared to sequential execution?

```
Critical path execution time is 10 + 15 + 15 + 25 + 10 = 75. Overal
sequential execution time = 100. Speedup 100/75 = 4/3 = 1.33
```

(c) In der Vorlesung hatten wir zwei Arten Race-Conditions unterschieden. Nennen und beschreiben Sie diese zwei Formen.

In the lectures we have made a distinction between two kinds of race conditions. Name and describe the two forms.

/4P

```
 Data race: Fehlerhaftes Programmverhalten verursacht durch ungenügend
synchronisierten Zugriff zu einer gemeinsam genutzten Resource, z.B.
gleichzeitiges Lesen/Schreiben oder Schreiben/Schreiben zum gleichen
Speicherbereich. Bad interleaving: Fehlerhaftes Programmverhalten
verursacht durch eine unglückliche Ausführungsreihenfolge eines
Algorithmus mit mehreren Threads, selbst dann wenn die gemeinsam
genutzten Resourcen anderweitig gut synchronisiert sind.
```

Aufgabe 6. (7P)

Ein Zoo-Simulationsprogramm simuliert zwei Arten Tiere: Hasen (Rabbit) und Löwen (Lion) unter Verwendung von Threads. Es gibt eine einzige Wasserquelle, genannt WaterSource, welche folgendermassen implementiert werden muss:

- Tiere (Threads) unterschiedlicher Klasse dürfen nicht gleichzeitig von der Wasserquelle trinken.
- Jedes Tier (Thread), das von der Wasserquelle trinken will, darf irgendwann trinken.

Das Protokoll soll mit den Funktionen `enter` und `leave` implementiert werden, welche jeweils mit dem Typ des Tiers als Parameter aufgerufen werden. `enter` wird aufgerufen, wenn ein Tier trinken will. Trinkt bereits ein Löwe, so muss der Hase warten (und umgekehrt). Jedes Tier trinkt nur für eine beschränkte Zeit und ruft dann `leave` auf.

/4P (a) Vervollständigen Sie den Code der Klasse WaterSource auf der nächsten Seite so, dass der gegenseitige Ausschluss von Hasen und Löwen gewährleistet ist und dass immer mindestens ein Tier trinken kann.

A Zoo simulation program simulates two kinds of animals: rabbits (Rabbit) and lions (Lion). There is a single water source called WaterSource that must be implemented in the following way:

- *Animals (threads) of different type may not drink from the water source simultaneously.*
- *Every animal (thread) who needs to drink from the water source eventually succeeds.*

The protocol is implemented via the functions `enter` and `leave` that are called with the animal type as parameter. `enter` is called when an animal wants to drink. If a lion is currently drinking, the rabbit has to wait (and vice versa). Leave is called when the animal stops drinking.

Complement the code of class WaterSource on the next page such that the mutual exclusion of rabbits and lions is ensured and that always at least one animal can drink.

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

const int Lion = 0;
const int Rabbit = 1;
class Watersource{
private:
  int lions = 0;
  int rabbits = 0;
  using unique_lock = std::unique_lock<std::mutex>;
    std::mutex mtx;
    std::condition_variable cv;
public:
  // called with type == Lion or type == Rabbit
  void enter(int type){
  unique_lock lock(mtx);
  if (type == Lion){
          cv.wait(lock, [&]{return rabbits==0;});
          lions++;
        }
  else{
  cv.wait(lock, [&]{return lions==0;});
  rabbits++;
  }
  }

  // called with type == Lion or type == Rabbit
  void leave(int type){
      unique_lock lock(mtx);
  if (type == Lion)
  lions--;
  else
  rabbits--;
  cv.notify_all();
  }
}
```

/3P (b) Sie stellen fest, dass Ihr Programm ein ungünstige Konstellation zulässt: wenn ständig Löwen zur Wasserquelle kommen und sich beim Trinken abwechseln, verdursten die Hasen. Beschreiben Sie kurz, wie Sie dieses Problem lösen würden. Gehen Sie davon aus, dass das Betriebssystem grundsätzlich eine faire Zuteilung des Monitors realisiert.

You realize that your program permits a bad constellation: when constantly lions come to the water source and drink one after the other, the rabbits die of thirst. Describe shortly, how this problem can be solved. You can assume that the runtime system generally provides a fair scheduling of the monitor.

```
Vorsehen weiterer Counter: rabbits_waiting, lions_waiting. Heraufzählen
des jeweiligen Counters vor dem Ausführen der Wartebedingung.
Herabzählen nach der Wartebedingung. In der Wartebedingung bekommt
ein Löwe nur Zugriff, wenn kein Hase trinkt und wenn kein Hase wartet
oder auch kein Löwe trinkt (und umgekehrt). Fairness reduziert sich
damit auf die faire Zuteilung durch das Laufzeitsystem.

if (type == Lion){
  lions_waiting++;
  cv.wait(lock, [&]{
    return rabbits==0 && (rabbits_waiting == 0 || lions == 0)
  });
  lions_waiting--;
  lions++;
}
else { // type == Rabbit
  rabbits_waiting++;
  cv.wait(lock, [&]{
    return lions==0 && (lions_waiting == 0 || rabbits == 0)
  });
  rabbits_waiting--;
  rabbits++;
}
```