Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Prüfung **(Lösung)**

Informatik II (D-BAUG)

Felix Friedrich, Hermann Lehner, Departement Informatik

ETH Zürich, 12.8.2019.

Name, Vorname:	. . . . .
Legi-Nummer:	. . . . .

Ich bestätige mit meiner Unterschrift, dass ich diese Prüfung unter regulären Bedingungen ablegen konnte und dass ich die allgemeinen Richtlinien gelesen und verstanden habe.

I confirm with my signature that I was able to take this exam under regular conditions and that I have read and understood the general guidelines.

Unterschrift:

Allgemeine Richtlinien:

1. Dauer der Prüfung: 60 Minuten.
2. Erlaubte Unterlagen: Wörterbuch (für von Ihnen gesprochene Sprachen). 4 A4 Seiten handgeschrieben oder $\geq$ 11pt Schriftgrösse.
3. Benützen Sie einen Kugelschreiber (blau oder schwarz) und keinen Bleistift. Bitte schreiben Sie leserlich. Nur lesbare Resultate werden bewertet.
4. Lösungen sind direkt auf das Aufgabenblatt in die dafür vorgesehenen Boxen zu schreiben (und direkt darunter, falls mehr Platz benötigt wird). Ungültige Lösungen sind deutlich durchzustreichen! Korrekturen bei Multiple-Choice Aufgaben bitte unmissverständlich anbringen!
5. Es gibt keine Negativpunkte für falsche Antworten.
6. Störungen durch irgendjemanden oder irgendetwas melden Sie bitte sofort der Aufsichtsperson.
7. Wir sammeln die Prüfung zum Schluss ein. Wichtig: Stellen Sie unbedingt selbst sicher, dass Ihre Prüfung von einem Assistenten eingezogen wird. Stecken Sie keine Prüfung ein und lassen Sie Ihre Prüfung nicht einfach am Platz liegen. Dasselbe gilt, wenn Sie früher abgeben wollen: Bitte melden Sie sich lautlos, und wir holen die Prüfung ab. Vorzeitige Abgaben sind nur bis 15 Minuten vor Prüfungsende möglich.
8. Wenn Sie zur Toilette müssen, melden Sie dies einer Aufsichtsperson durch Handzeichen.
9. Wir beantworten keine inhaltlichen Fragen während der Prüfung. Kommentare zur Aufgabe schreiben Sie bitte auf das Aufgabenblatt.

General guidelines:

1. *Exam duration: 60 minutes.*
2. *Permitted examination aids: dictionary (for languages spoken by yourself). 4 A4 pages hand written or $\geq$ 11pt font size.*
3. *Use a pen (black or blue), not a pencil. Please write legibly. We will only consider solutions that we can read.*
4. *Solutions must be written directly onto the exam sheets in the provided boxes (and directly below, if more space is needed). Invalid solutions need to be crossed out clearly. Provide corrections to answers of multiple choice questions without any ambiguity!*
5. *There are no negative points for wrong answers.*
6. *If you feel disturbed by anyone or anything, let the supervisor of the exam know immediately.*
7. *We collect the exams at the end. Important: You must ensure that your exam has been collected by an assistant. Do not take any exam with you and do not leave your exam behind on your desk. The same applies when you want to finish early: Please contact us silently and we will collect the exam. Handing in your exam ahead of time is only possible until 15 minutes before the exam ends.*
8. *If you need to go to the toilet, raise your hand and wait for a supervisor.*
9. *We will not answer any content-related questions during the exam. Please write comments referring to the tasks on the exam sheets.*

Question:	1	2	3	4	5	Total
Points:	21	10	7	10	12	60
Score:						

Generelle Anmerkung / *General Remark*

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

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

Aufgabe 1: Verschiedenes (21P)

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.

/2P (a) Tragen Sie in folgendem Array die Zahlen 1, 3, 11 und 15 ein, so dass das Array einen Min-Heap darstellt.

Enter into the following array the numbers 1, 3, 11 and 15 such that the array constitutes a Min-Heap.

1	5	2	13	8	6	3	14	15	9	12	11	10	7	4
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

/3P (b) Geben Sie für jedes der folgenden Probleme auf einem **unsortierten Array** der Länge n einen Algorithmus an, welcher das jeweilige Problem **im schlechtesten Fall** asymptotisch möglichst effizient löst. Charakterisieren Sie Ihren Algorithmus mit wenigen Worten oder geben Sie einen Namen eines bekannten Algorithmus an.

For each of the following given problems on an **unsorted array** *of length n state an algorithm that solves the problem* **in the worst case** *in the asymptotically most efficient way. Characterize your algorithm using a few words or give a name of a well known algorithm.*

Sortieren / Sort

Algorithmus *Algorithm*	Asymptotische Laufzeit *Asymptotic Running Time*
`Mergesort / Heapsort (NOT: Quicksort)`	$O(n \log n)$

Berechnung des Mittelwertes / Computation of the mean (average)

Algorithmus *Algorithm*	Asymptotische Laufzeit *Asymptotic Running Time*
`Sum up and divide`	$O(n)$

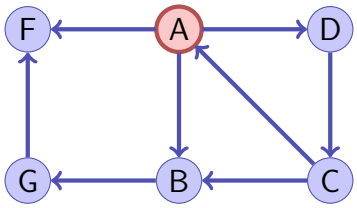
Finde kleinstes Element / Find smallest element

Algorithmus *Algorithm*	Asymptotische Laufzeit *Asymptotic Running Time*
`Elementwise comparison`	$O(n)$

(c) Der folgende Graph wird, ausgehend vom Knoten A mit Breitensuche und mit Tiefensuche besucht. Wenn es mehrere Möglichkeiten für eine Besuchsreihenfolge der Nachbarn gibt, wird immer die alphabetische Ordnung genommen. Geben Sie die Knoten in der Besuchsreihenfolge an.

The following graph is visited with a breadth-first search and a depth-first search algorithm starting at node A. If there are several possibilities for a visiting order of the neighbours, always the alphabetical order is chosen. Provide the nodes in the order they are visited.

/2P



Besuchte Knoten Tiefensuche/
Visited nodes Depth First Search:

A B G F D C

Besuchte Knoten Breitensuche/
Visited nodes Breadth First Search:

A B D F G C

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

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

/2P

Hashfunktion / *hash function* : $h(k) = k \bmod 13$

Schlüssel / *keys* : 0, 13, 1, 12, 25, 10, 23

0	1	2	3	4	5	6	7	8	9	10	11	12
0	13	1	25							10	23	12

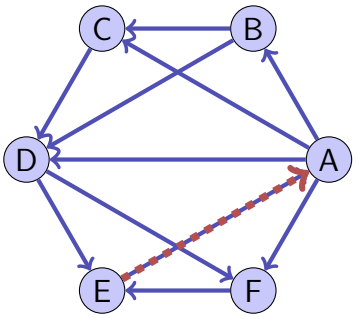
Wie viele Kollisionen treten bei der nachfolgenden (erfolglosen) Suche nach dem Schlüssel 26 auf? (Gezählt werden nur Kollisionen mit den belegten Plätzen)

How many collisions occur when key 26 is searched now (unsuccessfully)? (We only count collisions with the occupied spaces)

4

/2P (e) Streichen Sie in folgendem Graphen eine kleinstmögliche Menge von Kanten, so dass der verbleibende Graph eine topologische Sortierung hat.

In the following graph, cross out the smallest possible set of edges such that the remaining graph can be topologically sorted.



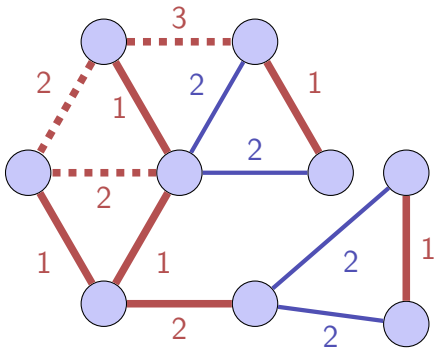
Geben Sie die dann mögliche topologische Sortierung an.

Provide the now possible topological sorting.

A B C D F E

/4P (f) **Markieren** Sie in folgendem ungerichteten Graphen die Kanten, welche zu **jedem** möglichen minimalen Spannbaum gehören. **Streichen** Sie dann alle Kanten, welche zu **keinem** möglichen minimalen Spannbaum gehören. Tipp: verwenden Sie den Algorithmus aus der Vorlesung.

In the following undirected graph **mark** *the edges that belong to* **every** *possible minimal spanning tree. Then* **cross out** *all edges that belong to* **no** *minimal spanning tree. Hint: use the algorithm from class.*



Wie viele verschiedene minimale Spannbäume gibt es zu dem gegebenen Graphen?

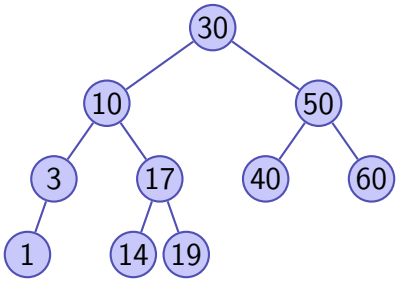
How many different Minimum Spanning Trees are there for the given graph?

4

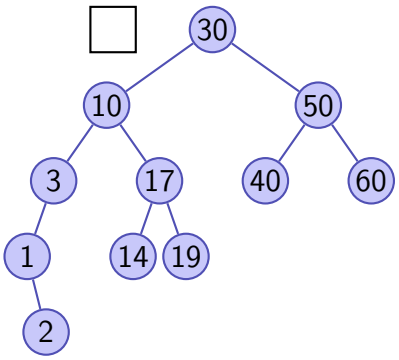
(g) Fügen Sie in folgendem AVL Baum den Schlüssel 2 ein und rebalancieren Sie. Wie sieht der AVL Baum nach dem in der Vorlesung gezeigten Algorithmus aus? Kreuzen Sie die richtige Antwort an.

In the following AVL tree, insert key 2 and rebalance. What does the AVL tree look like according to the algorithms that has been shown in class? Mark the correct answer.

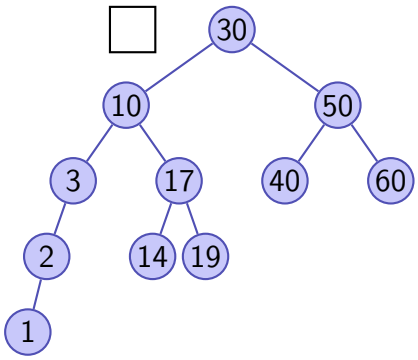
/2P



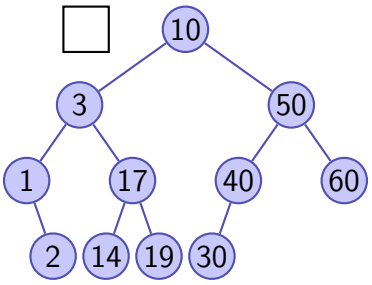
☐



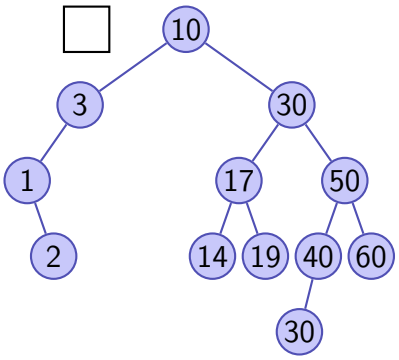
☐



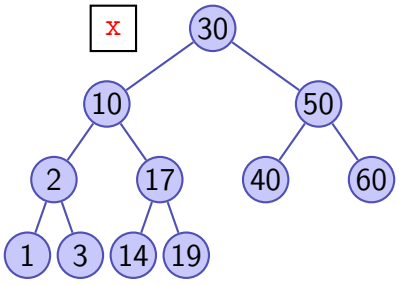
☐



☐



☒



/4P (h) Zu den folgenden Datenstrukturen der Java-API geben Sie jeweils asymptotische Laufzeiten der gegebenen Operationen an, welche Sie bei einer vernünftigen Implementation typischerweise erwarten würden. Die Datenstruktur enthalte n Elementen.

To the following data structures of the Java API, provide the asymptotic running time that you would expect for the given operations assuming a reasonable implementation. The data structure is supposed to contain n elements.

- **Array List**
  - Wahlfreier Zugriff / *Random Access* $\Theta(1)$
  - Anhängen / *Append* $\Theta(1)$ (amortisiert / *amortized*)
  - Suche kleinstes / *Search smallest* $\Theta(n)$
- **Linked List**
  - Wahlfreier Zugriff / *Random Access* $\Theta(1)$
  - Anhängen / *Append* $\Theta(1)$
  - Suche / *Search* $\Theta(n)$
- **TreeSet**
  - Einfügen / *Insert* $\Theta(\log n)$
  - Suchen / *Search* $\Theta(\log n)$
- **HashSet**
  - Einfügen / *Insert* $\Theta(1)$ (erwartet / *expected*)
  - Suchen / *Search* $\Theta(1)$ (erwartet / *expected*)

Aufgabe 2: Asymptotik (10P)

(a) Wählen Sie für die Funktionen unten aus der Liste der in der weissen Box angegebenen Funktionen jeweils diejenige aus, so dass die Gleichheit gilt.

For each case below, choose one of the provided functions from the white boxed list such that the equality holds.

/3P

Beispiel/*Example*: $\Theta(n+5) = \Theta(\boxed{2n})$

$1,$	$n \log \sqrt{n},$	$\log^2 n,$	$2n,$	$n \log \frac{1}{n},$	$n^2,$	$n^4,$	$n^2 \log n,$	$n \log^2 n,$	$n!,$	n^n

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

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

$$\Theta\left(\frac{20n^3 + 50n^2 + n}{50n^2}\right) = \Theta(\boxed{n})$$

In den folgenden Aufgabenteilen wird jeweils angenommen, dass die Funktion g mit $g(n)$ aufgerufen wird. Geben Sie jeweils die asymptotische Anzahl von Aufrufen der Funktion $f()$ in Abhängigkeit von $n \in \mathbb{N}$ mit Θ-Notation möglichst knapp an. Die Funktion f ruft sich nicht selbst auf. Sie müssen Ihre Antworten nicht begründen.

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

/2P (b)

```
void g(int n){
    for (int i = 0; i < n; ++i){
        for (int j = 0 ; j < n; ++j){
            for (int k = 0; k < n; ++k){
                f();
            }
        }
    }
}
```

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

$\Theta(n^3)$

/2P (c)

```
void g(int n){
    for (int i = 0; i<100;++i)
        f();
        for (int j = i; j<n; ++j){
            f();
        }
    }
}
```

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

$\Theta(n)$

/3P (d)

```
void g(int n){
    if (n>0){
        for (int i = 0; i<4; ++i){
            g(n-1);
        }
    } else {
        f();
    }
}
```

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

$\Theta(4^n)$

Aufgabe 3: Python (7P)

(a) Geben Sie die Ausgabe des folgenden Codes in nachfolgender Box an.

Specify the output of the following code in the box below.

/3P

```
nodes = {
  'A' : { 'Color' : 'red', 'Weight': 2 },
  'B' : { 'Color' : 'red', 'Weight': 8 },
  'C' : { 'Color' : 'black', 'Weight': 12},
  'D' : { 'Color': 'black', 'Weight': 3 },
  'E' : { 'Color': 'black', 'Weight': 8 }
}

special = {
  name: record['Color']
  for name, record in nodes.items()
  if record['Weight']>3
}

for name,color in special.items():
  print(name,"->",color)
```

```
B -> red
C -> black
E -> black
```

(b) Geben Sie die Ausgaben der Anweisungen in den Boxen im Programmtext unten an.

Fill the output of the statements into the boxes in the source code below.

/4P

```
num = [i for i in range(1,10)];
sq = [i*i for i in num];
l = list(zip(sq, num));
d = dict(zip(sq, num));
```

```
print(l[4]);
```

```
(25,5)
```

```
print(d[4]);
```

```
2
```

Aufgabe 4: Rekursion / Dynamische Programmierung (10P)

Sie besteigen einen Turm mit vielen ($n \in \mathbb{N}$) Stufen. Abhängig von der Länge Ihrer Beine und Ihrer Fitness können Sie bis zu $k > 0$ Stufen auf einmal nehmen. Auf dem Weg nach oben fragen Sie sich, wie viele Möglichkeiten es gibt, nach oben zu kommen.

You climb a tower with many ($n \in \mathbb{N}$) stairs. Depending on the length of your legs and your fitness you can take up to $k > 0$ steps at once. On the way to the top you ask yourself, in how many ways you can climb the tower.

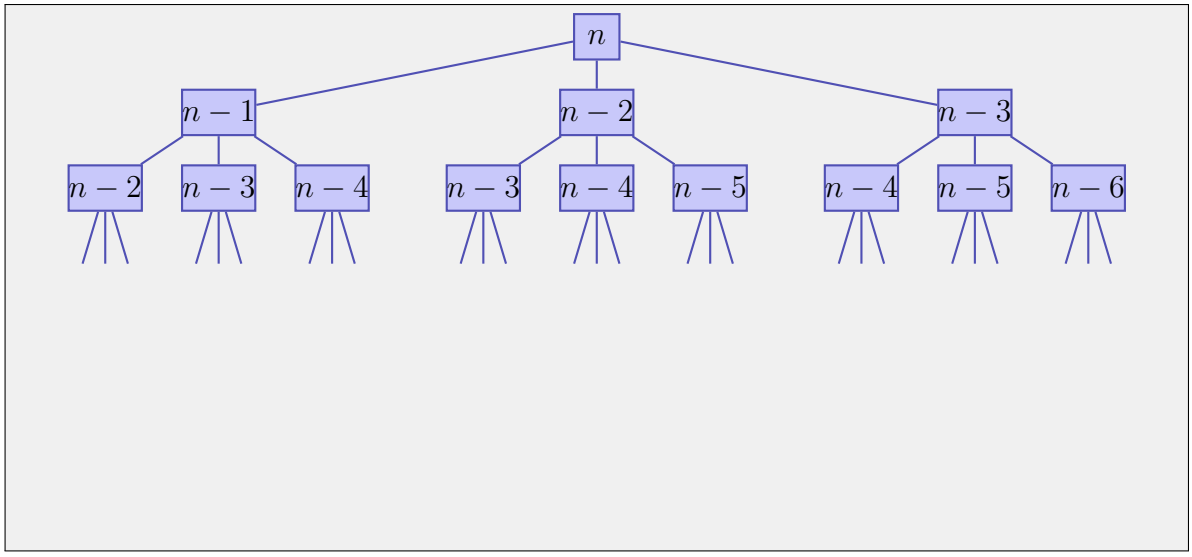
/2P (a) Aufgabe: Vervollständigen Sie die folgende rekursive Funktion so, dass sie die Anzahl der Möglichkeiten zurückgibt (also die Anzahl möglicher Kombinationen von Zahlen aus $\{1, .., n\}$ mit Summe n).

Task: complement the following recursive function such that it returns the number of possibilities. (i.e. the number of combinations of numbers from $\{1, ..., k\}$ summing up to n)

```
public static int solve(int n, int k){
    if (n == 0){
        return 1;
    }
    int combinations = 0;
    for (int i = 1; i <= Math.min(k,n); ++i){
        combinations += solve(n-i, k);
    }
    return combinations;
}
```

/2P (b) Sie stellen fest, dass Ihre Funktion für grosse n sehr lange benötigt. Skizzieren Sie den Rekursionsbaum (einen Teil davon) für $k = 3$.

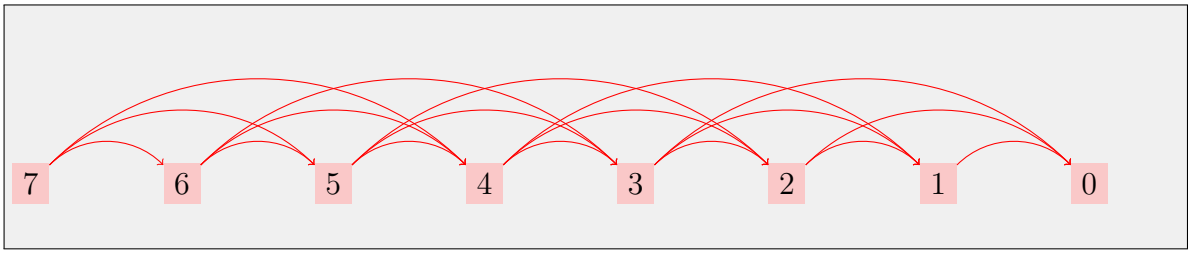
You realize that your function takes a lot of time for large n. Sketch the recursion tree (a part of it) for $k = 3$.



(c) Überlegen Sie sich (z.B. anhand Ihres Rekursionsbaumes) die Abhängigkeiten Ihres Problems. Zeichnen Sie in folgender kleinen Skizze (für $k = 3$, $n = 7$), die Abhängigkeiten ein. Sie zeigen damit dass es keine zirkulären Abhängigkeiten gibt.

Now think (e.g. with the help of your recursion tree) about the dependencies of your problem. Draw into the following small sketch (for $k = 3$, $n = 7$) the dependencies. Doing so, you show that there are no circular dependencies.

/2P



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

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

/2P

```
public static int solve(int n, int k){
    int[] solution = new int[n+1];
    solution[0] = 1;

    for (int i = 1; i <= n; ++i){
        for (int j = 1; j <= Math.min(k,i); ++j){
            solution[i] += solution[i-j];
        }
    }
    return solution[n];
}
```

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

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

/2P

$O(n \cdot k)$

Aufgabe 5: Algorithmen (12P)

/4P (a) Ihr Chef weiss, dass Sie auf gute Ideen kommen, wenn es um die Lösung algorithmischer Probleme geht. Nun betraut er Sie damit, die Erneuerung eines riesigen **Wasserleitungsnetzes** zu koordinieren. Sie haben die undankbare Aufgabe, die Kosten so klein wie möglich zu halten. Das Wasserleitungsnetz besteht aus N **Stationen**, welche mit M **Leitungen** untereinander verbunden sind. Da Sie wissen, dass das Netzwerk an vielen Stellen redundant ausgelegt ist, haben Sie folgende Idee. Sie erneuern nur die absolut notwendigen Leitungen: Jede **Station muss mit jeder anderen Station (möglicherweise indirekt) über erneuerte Leitungen verbunden** bleiben. Sie kennen die Erneuerungskosten jeder einzelne Leitung. Wie **berechnen Sie die minimalen Kosten für das gesamte Vorhaben möglichst effizient**? Beschreiben Sie ganz kurz die verwendete Datenstruktur und benennen Sie den verwendeten Algorithmus.

Your boss knows that you always have good ideas when it comes to solving an algorithmic problem. Now he puts you in charge of renewing a huge **water pipe network**. *You are assigned the invidious task of keeping costs as small as possible. The water pipe network consists of* N **stations** *that are mutually connected with* M **water pipes**. *Because you know that the network is pretty redundant, you have the following idea. You only renew the absolutely necessary: every station* **needs to be (potentially indirectly) connected with each other station via some renewed pipe** . *For each pipe you know the costs for its renewal. How do you compute the* **minimal costs for the whole project as efficient as possible**? *Briefly describe the used data structure and name the used algorithm.*

Das Netzwerk ist ein ungerichteter Graph bestehend aus Knoten (Stationen) und Kanten (Leitungen). Jede Kante ist mit den Erneuerungskosten gewichtet. Sie benötigen einen minimalen zusammenhängenden Graph. Sie berechnen diesen mit dem Kruskal-Algorithmus.

/2P (b) Was ist die asymptotische Laufzeit Ihres Algorithmus in Abhängigkeit der Anzahl Stationen N und Leitungen M?

What is the asymptotic runtime of your algorithm as a function of number stations N *and connections* M*?*

Laufzeit des Kruskal-Algorithmus: $O(M \log N)$

(c) Sie legen Ihrem Chef die Zahlen vor. Er ist nicht begeistert. „Das ist ja viel zu teuer", schimpft er. „Aber ich habe die minimalen Kosten berechnet", versichern Sie ihm. Er kommt auf eine neue Idee: „eigentlich ist **nur die (indirekte) Verbindung zwischen Station** 42 **und Station** 479 **wirklich kritisch**", sagt er nachdenklich. Er möchte von Ihnen wissen, ob Sie garantieren können, dass **beim Ausfall einer einzigen Leitung zumindest die Verbindung zwischen den genannten Stationen noch besteht**. Er weist Sie darauf hin, dass das Wasser in jedem Rohr in beiden Richtungen fliessen kann. Was tun Sie? Beschreiben Sie sehr kurz die verwendete Datenstruktur und bennennen Sie die verwendeten Algorithmen.

You confront your boss with the numbers. He is not so happy. "This is far too expensive", he moans. "But I have definitely computed the minimal costs", you reassure him. He has a new idea: "actually, **only the (indirect) connection between station** 42 **and station** 479 **is really critical***", he says. He wants to know from you, if you can guarantee that* **if a single pipe breaks down at least the connection between the two named stations is still working**. *He additionally points out that the water can flow in both directions in each pipe. What do you do? Briefly describe the data structure employed and name the used algorithms.*

/4P

Modelliere das Wassernetzwerk nun als gerichteten Graphen. Dazu verdopple jede Station als Paar von Ein- und Ausgangsknoten. Und verdopple jede ungerichtete Kante nun als Paar von gerichteten Kanten von Aus- zu Eingangsknoten. Jede Kante bekommt Kapazität 1 zugeordnet. Dann berechne mit dem Ford-Fulkerson Algorithmus den maximalen Fluss von Station 42 zu Station 479. Sie löschen dafür Eingangskanten nach der Quelle 42 und Ausgangskanten von der Senke 479. Hat der Fluss einen Wert $>= 2$, so ist nach dem MaxFlow-MinCut Theorem sichergestellt, dass jede Leitung mindestens 2-fach redundant vorhanden ist.

(d) Was ist die asymptotische Laufzeit Ihres Algorithmus in Abhängigkeit der Anzahl Stationen N, wenn von jeder Station aus maximal 256 Leitungen abgehen können?

What is the asymptotic runtime of your algorithm as a function of the number of stations N *if each station can only be connected to maximally* 256 *pipes?*

/2P

Fluss begrenzt durch 256. Ford-Fulkerson Algorithmus Laufzeit: $O(|fmax| \cdot M) = O(M) = O(N)$.