Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Prüfung **(Lösung)**

# Informatik II (D-BAUG)

Felix Friedrich, Hermann Lehner, Departement Informatik

ETH Zürich, 13.08.2018.

| | |
|---|---|
| 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: 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: | 18 | 10 | 8 | 11 | 13 | 60 |
| 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 comprehensible.*

# Aufgabe 1: Verschiedenes (18P)

1) In dieser Aufgabe sollen nur Ergebnisse angegeben werden. Begründungen sind nicht notwendig.
2) Als Ordnung verwenden wir für Buchstaben die alphabetische Reihenfolge, für Zahlen die aufsteigende Anordnung gemäss ihrer Grösse.

*In this task only results have to be provided. Explanations are not required.*

*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*

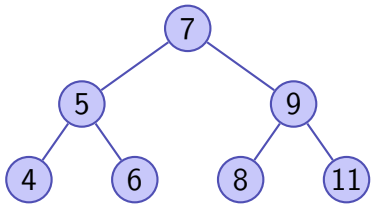
◯ im schlechtesten Fall eine Laufzeit von $O(n \log n)$/
*a worst case running time of* $O(n \log n)$

◯ 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.*

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

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

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

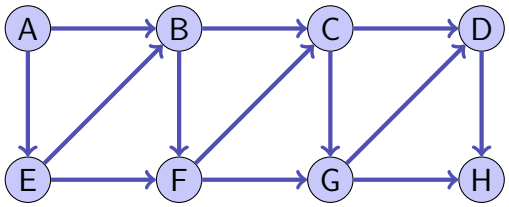
◯ Hauptreihenfolge/*preorder* ◯ Nebenreihenfolge/*postorder*
◯ Symmetrische Reihenfolge/*inorder* √ Keine davon/*none of them*

(d) Der folgende Array soll einen MinHeap speichern. Markieren Sie die beiden Elemente (ein Eltern-Kind Paar), welche aktuell die Heap-Eigenschaft verletzen. *The following array should store a MinHeap. Mark the two elements (a child-parent pair) which currently violate the heap property.* /4P

| 2 | 4 | 5 | 9 | **7** | 8 | 10 | 12 | 11 | **6** | 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.*

| $c$ | $v(c)$ | $c$ | $v(c)$ | $c$ | $v(c)$ | $c$ | $v(c)$ |
|---|---|---|---|---|---|---|---|
| - | 0 | 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 folgende Namen (Schlüssel) in der gegebenen Reihenfolge 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 'java' eingefügt.

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

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

| **eth** | inf | int | **zoo** | | **the** | java |
|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 |

(g) Geben Sie die Ausgabe der folgenden Anweisung in der Box unten an.

*Provide the output of the following statement in the box below.*

/3P

```
Arrays.asList(10, 3, 8, 9, 2, 5)
    .stream()
    .filter(i -> {return i % 2 == 1;})
    .map(i -> i/2)
    .sorted()
    .forEach(System.out::print);
```

124

## Aufgabe 2: Asymptotik (10P)

(a) Finden Sie aus der Liste der in der weissen Box angebenen Funktionen jeweils diejenige, so dass die Gleichheit gilt.

*Find from the white boxed list of provided functions for each case below the function such that the equality holds.*

/3P

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

$$\boxed{1, \quad \log n, \quad \log^2 n, \quad n, \quad n\log n, \quad n^2, \quad n^2\log n, \quad n\log^2 n, \quad n!, \quad n^n}$$

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

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

$\Theta(|\sin(n)| \cdot \log^2 n) = \Theta(\boxed{\log^2 n})$

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 $\Theta$-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 $\Theta$ notation as tight as possible. The function $f$ does not call itself. You do not have to justify your answers.*

/2P (b)

```
void g(int n){
    if (n<100){
        f();
    }
}
```

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

$\Theta(1)$

/2P (c)

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

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

$\Theta(\log^2 n)$

/3P (d)

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

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

$\Theta(2^n)$

# Aufgabe 3: Code Lesen & Schreiben (8P)

/8P

Vervollständigen Sie die folgende Methode `BFS(int s)`, welche auf einem Graphen ausgehend von einem bezeichneten Knoten `s` eine Breitensuche (BFS) durchführt.
Dabei soll ein Array `int[] predecessor` zurückgegeben werden, welcher:

- für jeden Knoten `x` den BFS-Vorgänger in `predecessor[x]` speichert,
- oder, falls `x` keinen Vorgänger hat, `predecessor[x]` auf `x` setzt.

*Complete the following method template `BFS(int s)`, which implements a Breadth-first search (BFS) starting from a designated node `s` of a graph.*
*The method should return an array `int[] predecessor` which:*

- *stores for each node `x` its BFS parent in `predecessor[x]`,*
- *or, if `x` has no such parent, sets `predecessor[x]` to `x`.*

```
class Graph {
    private int V; // Number of vertices
    private ArrayList<LinkedList<Integer>> adj; // Adjacency List
    // BFS traversal from a given source s
    public int[] BFS(int s) {
        // Initialization
        boolean[] visited = new boolean[V]; Arrays.fill(visited, false);
        int[] predecessor = new int[V];
        for (int i = 0; i < V; i++) {
            predecessor[i] = i;
        }
        // Queue preparation
        LinkedList<Integer> queue = new LinkedList<Integer>();
        visited[s] = true;
        queue.add(s);
        // BFS implementation
        while (!queue.isEmpty()) {
            int u = queue.poll();
            for (int v : adj.get(u)) {
                if (!visited[v]) {
                    visited[v] = true;
                    queue.add(v);
                    predecessor[v] = u;
                }
            }
        }
        return predecessor;
    }
    ... // Constructor
}
```

# Aufgabe 4: Datenbanken (11P)

Angenommen eine Datenbank zur Verwaltung von archäologischen Fundstücken enthält folgende Tabellen zu Gegenständen sowie ihren Fund- und Ausstellungsorten.

*Consider a data base for managing archeological finds which contains the following tables concerning the items and their finding- as well as exhibition places.*

***Item***

| id | name | age |
|---|---|---|
| 1 | Münze | 2300 |
| 2 | Amphore | 2700 |
| 3 | Säule | 2500 |
| 4 | Vase | 1830 |
| 5 | Schwert | 1280 |
| 6 | Statue | 2000 |
| 7 | Ring | 1800 |
| 8 | Gummistiefel | 35 |

***Place***

| id | name |
|---|---|
| 1 | Knossos |
| 2 | Athen |
| 3 | Delphi |
| 4 | Festos |
| 5 | Pompeji |
| 6 | Zurich |
| 7 | Berlin |
| 8 | Amsterdam |

***foundAt***

| item_id | place_id |
|---|---|
| 1 | 2 |
| 2 | 1 |
| 3 | 1 |
| 4 | 5 |
| 5 | 5 |
| 6 | 2 |
| 7 | 5 |
| 8 | 2 |

***displayedAt***

| item_id | place_id | originality |
|---|---|---|
| 1 | 1 | original |
| 1 | 2 | kopie |
| 1 | 6 | kopie |
| 2 | 1 | original |
| 2 | 6 | kopie |
| 3 | 3 | original |
| 4 | 8 | kopie |
| 5 | 5 | original |
| 6 | 3 | original |

/6P (a) Schreiben Sie zu den folgenden Anfragen jeweils in Ihren Worten, was gesucht ist.

*To the following queries write down in your own words what is looked for.*

```
SELECT name, age
FROM Item
WHERE age > 2000
```

Name und Alter von Gegenständen, welche älter als 2000 Jahre alt sind.

```
SELECT i.name
FROM Item i, Place p, foundAt f
WHERE i.id = f.item_id AND f.place_id = p.id AND p.name = "Knossos"
```

Name von Gegenständen, welche in Knossos gefunden wurden.

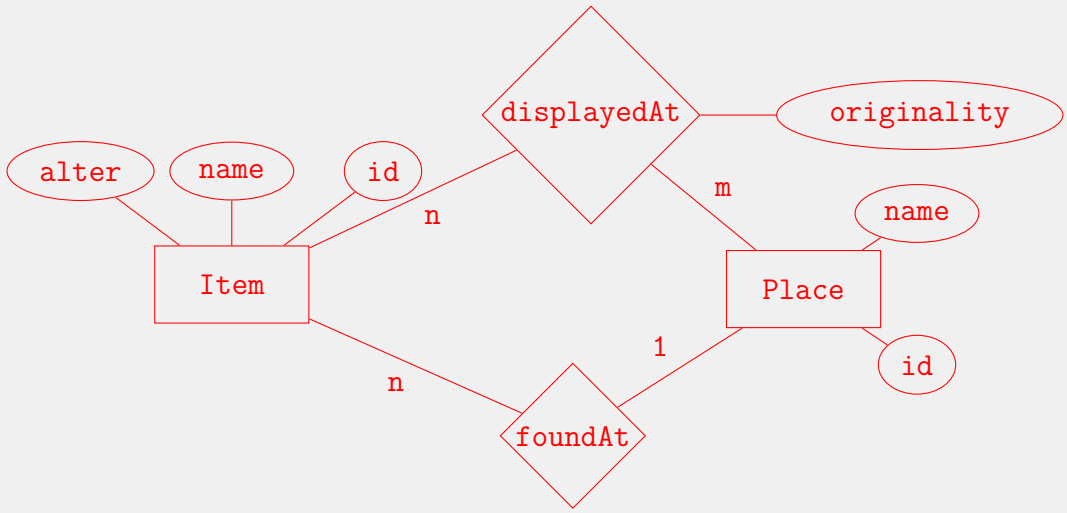
```
SELECT DISTINCT name
FROM Place
  INNER JOIN displayedAt ON displayedAt.place_id = Place.id
  INNER JOIN foundAt ON foundAt.place_id = displayedAt.place_id
WHERE displayedAt.item_id = foundAt.item_id
  AND displayedAt.originality = "original"
```

Name von Orten, an denen dort gefundene Gegenstände im Original ausgestellt wurden.

(b) Zeichnen Sie ein Entity-Relationship-Diagramm zum obigen relationalen Modell. Geben Sie die Funktionalitäten (Kardinalitäten) an.

*Draw an Entity-Relationship-diagram to the relational model from above. Provide the cardinalities.*

/3P




(c) Geben Sie im folgenden an, ob sie die Tabellen hätten weiter zusammenfassen können. Begründen Sie Ihre Entscheidung im Detail.

*Provide in the following if the tables could have been merged any further. Give a detailed explanation of your decision.*

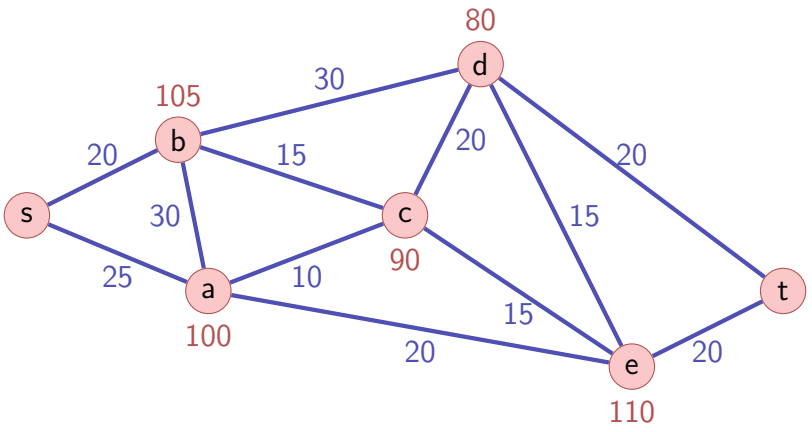
/2P

Die Tabelle Fundort hätte als 1:n Tabelle mit der Gegenstand-Tabelle vereint werden können: id / name / fundort

# Aufgabe 5: Kürzeste Wege (13P)

Der folgende ungerichtete Graph $(V, E)$ stellt eine Karte dar, bestehend aus Startpunkt $s$, Zielpunkt $t$ und Orten $a, b, \ldots, e$. Die Zahlen (Gewichte) an den Kanten bezeichnen Reisekosten zwischen den Orten. Die Zahlen an den Knoten bezeichnen allfällige Übernachtungskosten. Am Start- und Zielort kann und muss nicht übernachtet werden.

*The following undirected graph $(V, E)$ represents a map consisting of start point $s$, end point $t$ and locations $a, b, \ldots, e$. The numbers (weights) at the edges represent travelling costs between locations. The numbers at the nodes represent accomodation costs for potential overnight stays. At start point and end point, accomodation is neither possible nor necessary.*




/2P (a) Wir gehen vorerst davon aus, dass jede Strecke von $s$ nach $t$ ohne Übernachtung gefahren werden kann, dass also nur Kosten entlang der Kanten anfallen. Welchen Algorithmus schlagen Sie zur möglichst effizienten Bestimmung eines günstigsten Pfades vor?

*For this part we assume that every path from $s$ to $t$ can be travelled without staying overnight, i.e. that costs only incur at the edges. Which algorithm do you propose to compute a cheapest possible path as efficiently as possible?*

Dijkstra Algorithm

/2P (b) Was ist die asymptotische Laufzeit des Algorithmus in Abhängigkeit von der Anzahl Knoten $|V|$ und Anzahl Kanten $|E|$?

*What is the asymptotic runtime of your algorithm as a function of the number of nodes $|V|$ and edges $|E|$?*

$O((|E| + |V|) \log |V|) = O(|E| \log |V|)$

(c) Wir gehen nun davon aus, dass auf jedem möglichen Weg von $s$ nach $t$ genau einmal übernachtet werden muss, dass also zusätzlich zu den Kosten entlang der Kanten die Übernachtungskosten bei genau einem besuchten Ort anfallen.
Beispiel: Benützen des Wegs $s$—$a$—$e$—$t$ mit Übernachtung bei $e$ kostet 175.
Beschreiben Sie einen möglichst effizienten Algorithmus zur Bestimmung des günstigsten Weges mit genau einer Übernachtung. Der Algorithmus soll den günstigsten Weg und den Übernachtungsort ausgeben.

*Now we assume that on any path from $s$ to $t$ we have to stay overnight at exactly one location. Thus we assume that in addition to the travelling costs at the edges there are accomodation costs at one of the visited locations.*
*Example: Taking path $s$—$a$—$e$—$t$ with accomodation at $e$ costs 175.*
*Describe an efficient algorithm to determine a cheapest possible path with exactly one overnight stay. The algorithm should output the cheapest path and the chosen accomodation location.*

/7P

We use Dijkstras algorithm twice, once from $s$ and once from $t$ in order to determine both the shortest path from $s$ and from $t$ to each node. Then we add for each node the path costs from $s$ and from $t$ plus the accomodation costs of the hotel and choose the hotel $x$ with smallest costs. We determine the best path by traversing back both from $x$ to $s$ and to $t$.

Alternate Solution: We make a copy $G'$ of the graph $G$, and connect each node $v \in V \setminus \{s,t\}$ with its copy $v'$ by an edge with weight corresponding to the overnight stay at hotel $v$. Then we run Dijkstra from $s$ to $t'$. We determine the best path by traversing back from $t'$ to $s$; the overnight stay corresponds to the first encountered original (i.e. not a copy) node $x$.

(d) Was ist die Laufzeit des Algorithmus von (c) mit kurzer Begründung?

*What is the running time of the algorithm of (c) with short explanation?*

/2P

Twice Dijkstra yields $O((|E|+|V|)\log|V|) = O(|E|\log|V|)$ Determination of overal cost for each location in additionally $O((|V|)$ steps, yielding $O(|E|\log|V|)$.

Alternate Solution: Copying the graph takes time $O(|V| + |E|)$, running Dijkstra in the new graph $G \cup G'$ which has size $2|E| + 3|V|$ takes time $O((|E|+|V|)\log|V|) = O(|E|\log|V|)$.