

1st Exam “Algorithms and Data Structures”

February 16, 2023

Name: *GPT-4, 19.3.2023*

Matriculation Number:

I hereby declare that I will only use the one-sided handwritten A4 cheat sheet written by me. I feel physically able to take the exam. If I am not registered for the exam, my participation is subject to change. I will not take any copies of the exam out of the room (neither printed nor handwritten).

Signature:

Score achieved:

1	2	3	4	5	6	7	8	9	10	Σ
<i>3</i>	<i>4</i>	<i>3,5</i>	<i>0</i>	<i>3</i>	<i>0</i>	<i>2,5</i>	<i>3,5</i>	<i>3,5</i>	<i>1</i>	<i>24</i>

Important Information:

- Please place your ID card and student ID card on the table.
- Turn off and pack away your cell phones completely.
- **You only need to answer 9 of the 10 questions.** If you answer all 10 questions, we will only consider the best 9 of them in the final sum.
Therefore, you should focus on exactly 9 questions, and only attempt the remaining question if you have time.
- Check that your exam paper includes all 15 pages.
- Write your name on each sheet.
- If you need more paper, we will provide it to you. Do not use your own paper.
- Clearly indicate on new sheets which task you are working on.
- Hand in your cheat sheet along with the exam.
- Do not write with a pencil.
- The total processing time is 120 minutes.

Good luck!

NAME: _____

Aufgabe 1: Landau Notation and Growth of Functions (2.5 + 0.5 + 1 Punkte)

- a) Decide for each of the following function pairs f and g , in which asymptotic relationship they stand to each other. Cross out EVERY box that represents a true statement. You receive 0.5 points for every completely correctly filled row in the table.

f	g	$f \in o(g)$	$f \in \mathcal{O}(g)$	$f \in \Theta(g)$	$f \in \Omega(g)$	$f \in \omega(g)$
$\sin(n)$	n	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/> ✓
$n^2 + \cos(n)$	$n^2 + \frac{2}{n}$	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/> ✓
$(\sqrt{5})^n$	4^n	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/> ✗
$\log(n^n)$	$n^{2/3}$	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/> ✗
$n \bmod 20$	n^2	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/> ✓

- b) Which of the following statements implies that $f(n) \in \Omega(g(n))$ holds? Cross exactly one box.

☐ $f(n) \leq 4g(n)$ for all $n \geq 1$

☒ $f(n) \geq 4g(n)$ for all $n \geq 136$

☐ $\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = 0$

☐ None of the above statements.

- c) Prove that $\sqrt{n} \in o(n^{2/3})$ holds.

To prove that $\sqrt{n} \in o(n^{2/3})$, we need to show that $\lim_{n \rightarrow \infty} \frac{\sqrt{n}}{n^{2/3}} = 0$.

Using the properties of exponents, we can rewrite the expression inside the limit as:

$$\lim_{n \rightarrow \infty} \frac{n^{1/2}}{n^{2/3}} = \lim_{n \rightarrow \infty} n^{\frac{1}{2} - \frac{2}{3}}$$

Now, subtract the exponents:

$$\frac{1}{2} - \frac{2}{3} = \frac{3 - 4}{6} = -\frac{1}{6}$$

So the expression becomes:

$$\lim_{n \rightarrow \infty} n^{-1/6}$$

As n approaches infinity, $n^{-1/6}$ approaches 0. Therefore, the limit is 0, and we have shown that $\sqrt{n} \in o(n^{2/3})$.

Aufgabe 2: Multiple Choice - Diverse Topics (Score per sub-task: 0.5 Punkte)

For all questions, exactly one answer choice is correct and therefore only one box is to be filled in. If you fill in more than one box for a question, your answer for that question will be considered incorrect. For each correctly answered question, you will receive 0.5 points, for an incorrectly answered question 0 points.

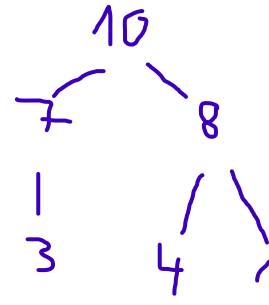
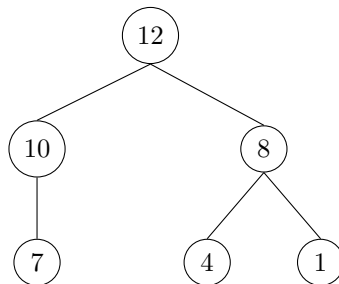
- a) If we could find the pivot element in the QuickSort algorithm in $O(1)$ time, then we would get the best guarantee on the running time with
☐ the arithmetic mean ☒ the median
- b) The connected components of an undirected graph are uniquely determined
☒ Correct ☐ Incorrect
- c) The principle of “Memoization” in dynamic programming means
☐ we store the dynamic-programming recurrence in a static array so that it may be applied at each level of recursion.
☒ we save the solutions of smaller subproblems if we need them again.
☐ the dynamic-programming algorithm can remember the current location in the recursion tree.
☐ we will remember the recursions occurring in dynamic programming for future exams.
- d) Which of the following properties is *not* a desired property of a hash function $h(x)$?
☐ $h(x)$ should be computable in $O(1)$ time.
☐ The values of $h(x)$ should be within the size of the hash table.
☐ For different x , $h(x)$ should also assume different values.
☒ If x_1 to x_n are the elements to be inserted into the hash table, then $h(x_1)$ to $h(x_n)$ should be evenly distributed over the natural numbers.
- e) In the Relax-operation of the Floyd-Warshall algorithm, if we replace the minimum with the maximum, then the algorithm will not find the shortest, but the longest paths between pairs of vertices.
☐ Correct ☒ Incorrect
- f) If you increase every edge weight by the same positive constant, then the solution to the single-source shortest-path problem will not change.
☐ Correct ☒ Incorrect
- g) A connected, unweighted Graph $G = (V, E)$ has a unique minimum spanning tree.
☐ Correct ☒ Incorrect
- h) If $T(n) = 2T(n/3) + \sqrt{n}$ holds, then we also have
☐ $T(n) = \Theta(\sqrt{n})$
☐ $T(n) = \Theta(n^2)$
☒ $T(n) = \Theta(n^{\log_3(2)})$
☐ $T(n) = \Theta(n^{\log(3)})$

4

NAME: _____

Aufgabe 3: Data structures (1 + 1 + 0.5 + 1.5 Punkte)

- a) Perform the DecreaseKey(0, 3) operation on the following Max-Heap (in this operation, the key of the root element is set to 3). Draw the resulting Max-Heap.



1

- b) Give an instance of a knapsack problem with 4 different items and weights where the greedy algorithm does not find the optimal solution. The greedy algorithm selects in each step the item with the best value-weight ratio that still fits in the knapsack.

Consider a knapsack with a capacity of 50 units and the following 4 items:

Item A: weight = 40 units, value = 60

Item B: weight = 25 units, value = 45

Item C: weight = 25 units, value = 45

Item D: weight = 10 units, value = 20

Using the greedy algorithm, we first compute the value-weight ratios:

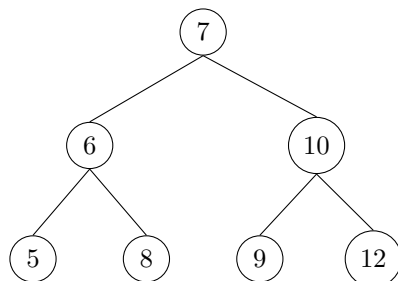
Item A: $60/40 = 1.5$ Item B: $45/25 = 1.8$ Item C: $45/25 = 1.8$ Item D: $20/10 = 2.0$

The greedy algorithm selects items in the order D, B (or C), as they have the highest value-weight ratios. The total value is $20 + 45 = 65$.

However, the optimal solution is to select items A and C (or A and B), which results in a total value of $60 + 45 = 105$. In this instance, the greedy algorithm does not find the optimal solution.

0,5

- c) Is the following tree a binary search tree?



☐ Yes

☒ No

✓

- d) Let T be a binary search tree with root node x . It is assumed that access to the children of any node can be obtained with $x.left$ and $x.right$. It holds that $x.left == NIL$ or $x.right == NIL$ if no left or right subtree exists. Provide the pseudo-code for an algorithm that outputs the nodes of the binary search tree in sorted order from largest to smallest using *PRINT*.

function PRINTBSTREVERSESORTED(Root x)

```
\If{$x \neq NIL$}
\State \Call{PrintBSTReverseSorted}{$x.right$}
\State $PRINT(x)$
\State \Call{PrintBSTReverseSorted}{$x.left$}
\EndIf
```

✓

end function

3,5

NAME: _____

Aufgabe 4: Hashing and MST (1 + 3 Punkte)

- a) Consider hashing with linear probing and the hash function $h(x) = (7 \cdot x + 2) \bmod 11$. The corresponding hash table consists of an array of length 11. Insert the elements 2, 3, 7, 15 and 1 in this order into the hash table.

	3				2	7			15	1
--	---	--	--	--	---	---	--	--	----	---

F

- b) Consider the following algorithm. The algorithm receives as input a connected, undirected, weighted graph $G = (V, E, c)$ with edge weight function $c : E \rightarrow \mathbb{R}_{\geq 0}$.

function ALTERNATIVEMST

$E_T = \emptyset$

Sei s beliebiger Knoten in G

for $v \in V \setminus \{s\}$ **do**

Let $e = (v, w)$ be the edge of v with minimal weight $c(e)$ that does not belong to E_T

$E_T \leftarrow E_T \cup \{e\}$

end for

return (V, E_T)

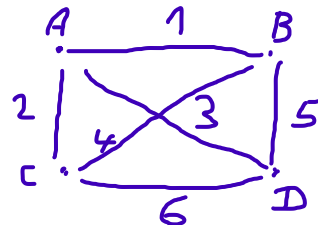
end function

Is AlternativeMST correct, that is, does AlternativeMST actually calculate a Minimum Spanning Tree of G ? Prove or disprove that AlternativeMST calculates a MST.

The given AlternativeMST algorithm does not always calculate a Minimum Spanning Tree (MST) of G . Let's disprove it using a counterexample.

Consider a connected, undirected, weighted graph G with vertices $V = \{A, B, C, D\}$ and edges $E = \{(A, B), (A, C), (A, D), (B, C), (B, D), (C, D)\}$ with weights given by the edge weight function c :

$c(A, B) = 1$
 $c(A, C) = 2$
 $c(A, D) = 3$
 $c(B, C) = 4$
 $c(B, D) = 5$
 $c(C, D) = 6$



Now, let's apply the AlternativeMST algorithm:

Set $s = A$ (an arbitrary vertex).

For vertex B, the minimal weight edge is (A, B) with weight 1. Add this edge to E_T .

For vertex C, the minimal weight edge is (A, C) with weight 2. Add this edge to E_T .

For vertex D, the minimal weight edge is (A, D) with weight 3. Add this edge to E_T .

The resulting tree $T = (V, E_T)$ has edges $E_T = \{(A, B), (A, C), (A, D)\}$ and total weight $1 + 2 + 3 = 6$.

F

However, the actual MST of G is:

MST = $\{(A, B), (A, C), (B, D)\}$ with total weight $1 + 2 + 5 = 8$.

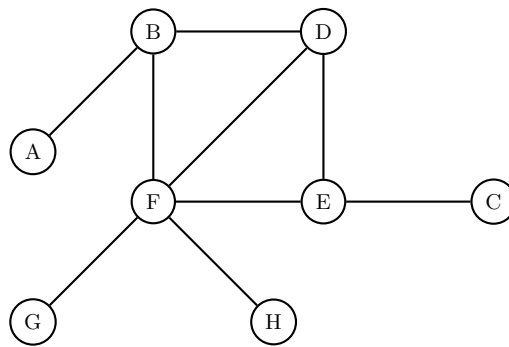
Since the resulting tree from the AlternativeMST algorithm does not match the actual MST, we can conclude that the AlternativeMST algorithm does not always calculate a Minimum Spanning Tree of G .

the provided counter example is not actually a counter example

0

NAME: _____

Aufgabe 5: DFS (1 + 1.5 + 0.5 + 0.5 + 0.5 Punkte)



- a) Perform the DFS algorithm on the above graph, starting at node *A*. Assume that the DFS algorithm visits nodes with smaller letters first when there is a choice. Give the order in which the various nodes are visited.

A B D E C F G H 1

- b) Now give the discovery time and the finishing time of the different nodes. Start with the discovery time $t = 1$ for node *A*.

Knoten v	Discovery-Zeit(v)	Finishing-Zeit(v)
A	1	16
B	2	15
C	6	7
D	3	14
E	4	13
F	8	12
G	9	10
H	11	12

0,5

- c) An undirected and unweighted graph is given as an adjacency matrix. Then the running time of the DFS algorithm is $O(|V| + |E|)$.

☐ Correct ☒ Incorrect

✓

- d) Let $G = (V, E)$ be a directed graph. Which of the following algorithmic problems can be solved with DFS? Put exactly one cross.

☐ Finding a source node.

☐ Finding a sink node.

☒ Both problems.

☐ Neither of the two problems.

✓

- e) Let $G = (V, E)$ be a directed, unweighted graph. A topological sorting exists only if the graph is acyclic.

☒ Correct

☐ Incorrect

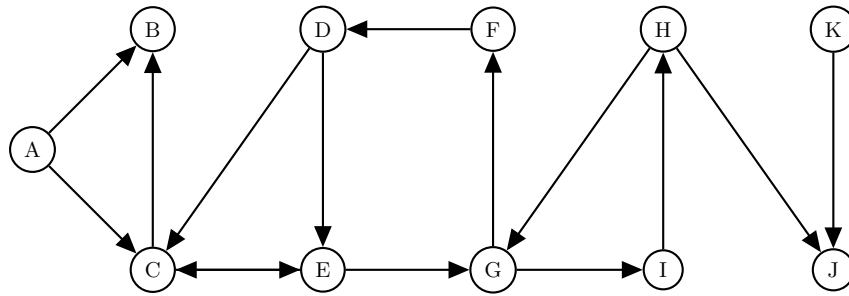
✓

3

NAME: _____

Aufgabe 6: Connected Components (0.5 + 1 + 1 + 1 + 0.5 Punkte)

Consider the following directed Graph.



a) How many strongly connected components does this graph have?

8 f

b) Draw the corresponding component graph.

A -- B -- CE -- D -- F -- GHI

J -- H ----- GHI

K

response not well-defined

c) Number the different nodes of the above component graph according to the alphabet, i.e., A, B, C, etc. Then specify a topological sort of the component graph.

wrong / not well-defined

d) Which nodes of the above component graph are source nodes, which are sink nodes?

Source nodes:

wrong / not well-defined

Sink nodes:

e) In a directed, acyclic graph, the component graph corresponds to the original graph.

☐ Correct

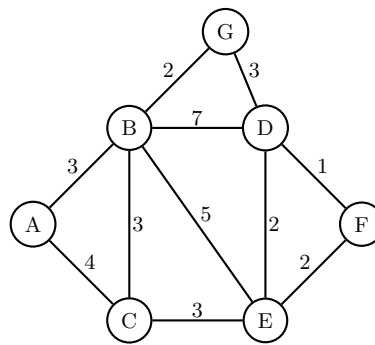
☒ Incorrect

f

0

NAME: _____

Aufgabe 7: Shortest Paths (1+1+0.5+0.5+1 Punkte)



- a) Perform the Dijkstra algorithm on the above graph to find the shortest path from A to all other nodes. Give the order in which the Dijkstra algorithm visits the nodes.

A, B, C, G, E, D, F ✓

- b) Now give the length of the shortest path from A to all other nodes.

B	C	D	E	F	G
3	4	8	7	9	5

✓

- c) If a graph does not contain a negative weight cycle, then the result of the Dijkstra algorithm is correct.

☒ Correct

☐ Incorrect

f

- d) For graphs with $\Theta(|V|^2)$ many edges, the Dijkstra algorithm is best implemented using a

☐ Array.

☒ Heap.

f

- e) Let $G = (V, E)$ be a directed graph. Now assume we define a weight function on the nodes, i.e., $c : V \rightarrow \mathbb{R}_{\geq 0}$. Does it still make sense to speak of the shortest path problem with a weight function on the nodes? If Yes, then give a formal definition of the problem. If No, then explain why the problem makes no sense.

Yes, it still makes sense to speak of the shortest path problem with a weight function on the nodes. To transform the problem into a more conventional shortest path problem with edge weights, you can associate the weight of a node with its outgoing edges.

Given a directed graph $G=(V,E)$ with a weight function on the nodes $c: V \rightarrow \mathbb{R}_0$, the objective is to find the shortest path between two nodes, s (source) and t (target), such that the total weight of the path is minimized.

Formally, the problem can be defined as:

Input: A directed graph $G=(V,E)$, a weight function on the nodes $c: V \rightarrow \mathbb{R}_0$, a source node $s \in V$, and a target node $t \in V$.

Output: A path P from s to t in G such that the total weight of the nodes in P is minimized.

0,5

To solve this problem, you can modify the graph by assigning the weight of each node v to its outgoing edges (v, u) for all $u \in V$. Let the modified edge weight function be $c': E \rightarrow \mathbb{R}_0$, defined as $c'(v, u) = c(v)$ for all $(v, u) \in E$. Now, you can apply any shortest path algorithm, such as Dijkstra's or Bellman-Ford, to find the shortest path with respect to the modified edge weights c' .

2,5

NAME:

Aufgabe 8: Algorithms (0.5+0.5+1+0.5+0.5+1 Punkte)

- a) What is the Best-Case running time of the following algorithm in relation to the length n of the input array A ?

```
function ARRAYFUNCTION(Array A)
  for  $i=1, \dots, A.length$  do
     $A[i] = i$ 
  end for
   $A = \text{QuickSortWithLastElementAsPivot}(A)$ 
  return A
end function
```

Best-Case: $\Theta(n \lg n)$

f

- b) The following program calculates a hash function. Which hash function is calculated by the program?

```
function HASHFUNCTION(Integer  $n$ )
   $n = n + 2$ 
   $n = 7 * n$ 
  while  $n \geq 153$  do
     $n = n - 153$ 
  end while
  return  $n$ 
end function
```

Hashfunktion:

$7 * (n + 2) \% 153$

✓

- c) What are the Best-Case and Worst-Case running time of the following algorithm in relation to the length n of the input array A ?

```
function ARRAYFUNCTION(Array A)
  for  $j = 2$  to  $A.length$  do
     $key = A[j]$ 
     $i = j - 1$ 
    while  $i > 0$  and  $A[i] > key$  do
       $A[i + 1] = A[i]$ 
       $i = i - 1$ 
    end while
     $A[i + 1] = key$ 
  end for
  return A
end function
```

Best-Case: $\Theta(n)$

Worst-Case: $\Theta(n^2)$

✓

- d) What is a Worst-Case input sequence of length n for the algorithm in c)?

$n, n-1, \dots, 1$

✓

(Continuation of Exercise)

- e) What is the Worst-Case running time of the following algorithm in relation to the length n of the input array A ?

```

function MYALGORITHM(Array  $A$ )
  if  $A.length \leq 100$  then
    return  $A[1]$ 
  end if
  Result =  $4 \cdot \text{MyAlgorithm}(A[1 \dots A.length-1])$ 
  return Result
end function

```

Worst-Case: $\Theta(n)$ ✓

Hint: $A[1 \dots A.length-1]$ is the Array A without the last element. This operation takes time $O(1)$.

- f) The diameter of an undirected and unweighted graph corresponds to the largest distance between two nodes in a graph. More specifically, the diameter is

$$\text{diam}(G) = \max_{(u,v) \in V \times V} \{\text{dist}(u,v)\},$$

where $\text{dist}(u,v)$ corresponds to the length of the shortest path between u and v . Provide an algorithm that calculates the diameter of a graph $G = (V, E)$. Describe your algorithm in pseudo-code.

To calculate the diameter of an undirected, unweighted graph G , we can use a Breadth-First Search (BFS) traversal from each node and find the largest distance encountered. Here's a short pseudo-code for the algorithm:

```

sql
Copy code
Function Diameter( $G$ ):
  max_distance = 0

  For each vertex  $u$  in  $V$ :
    dist = BFS_Distances( $G$ ,  $u$ )
    longest_path = max(dist.values())
    max_distance = max(max_distance, longest_path)

  Return max_distance

Function BFS_Distances( $G$ , start_vertex):
  Create a queue  $Q$ 
  Initialize distance dictionary dist with start_vertex: 0
  Enqueue start_vertex into  $Q$ 

  While  $Q$  is not empty:
    current_vertex = Dequeue( $Q$ )

    For each neighbor  $v$  of current_vertex:
      If  $v$  is not in dist:
        Enqueue  $v$  into  $Q$ 
        dist[ $v$ ] = dist[current_vertex] + 1

  Return dist

```

In this algorithm, the Diameter function calculates the diameter of the graph G by iterating through each vertex and computing the BFS_Distances. The largest distance found across all vertices is the diameter of the graph.

3,5

Aufgabe 9: Matrix Travel (2+2 Punkte)

Given an $M \times N$ matrix A of positive numbers, we can imagine this matrix as a game board of size $M \times N$. The entry at (i, j) corresponds to the costs that must be paid to enter field (i, j) of the game board. We now want to move a game piece as cost-effectively as possible from the upper left field $(1, 1)$ to the lower right field (N, M) . In each step, the game piece may only move one field down or one field to the right. So it can go from field (i, j) to field $(i + 1, j)$ or to field $(i, j + 1)$. We are interested in the costs of the cheapest path from $(1, 1)$ to (N, M) . In the following example, the cheapest path costs 33.

$$A = \begin{pmatrix} 4 & 7 & 8 & 4 & 6 \\ 6 & 7 & 3 & 9 & 2 \\ 3 & 8 & 1 & 2 & 4 \\ 7 & 1 & 7 & 3 & 7 \\ 2 & 9 & 8 & 9 & 3 \end{pmatrix}, \quad \text{A cheapest path } A = \begin{pmatrix} 1 & 7 & 8 & 4 & 6 \\ 6 & 7 & 3 & 9 & 2 \\ 3 & 8 & 1 & 2 & 4 \\ 7 & 1 & 7 & 3 & 7 \\ 2 & 9 & 8 & 9 & 3 \end{pmatrix}$$

- a) This problem can be solved with dynamic programming. Set up an appropriate recursive equation.

Let's define a function $\text{minCost}(i, j)$ that represents the minimum cost to reach cell (i, j) from cell $(1, 1)$. Then, the recursive equation can be written as:

$$\text{minCost}(i, j) = A[i][j] + \min(\text{minCost}(i-1, j), \text{minCost}(i, j-1))$$

Here, $A[i][j]$ represents the cost of entering cell (i, j) . The base cases for this recursive equation are:

$$\begin{aligned} \text{minCost}(1, 1) &= A[1][1] \\ \text{minCost}(i, 1) &= A[i][1] + \text{minCost}(i-1, 1) \text{ for all } i > 1 \\ \text{minCost}(1, j) &= A[1][j] + \text{minCost}(1, j-1) \text{ for all } j > 1 \end{aligned}$$

2

The minimum cost to reach the lower right cell (N, N) will be the value of $\text{minCost}(N, N)$.

- b) Complete the following pseudo-code to solve the problem in runtime $O(MN)$.

```
function MATRIXTRAVEL(Cost matrix A)
  T = MxN matrix filled with zeros
  for i = 1 to N do
    for j = 1 to M do
```

```
      \State $T[1,1] = A[1,1]$
      \For{$i = 2$ \textbf{to} $M$}
        \State $T[i,1] = A[i,1] + T[i-1,1]$
      \EndFor
      \For{$j = 2$ \textbf{to} $N$}
        \State $T[1,j] = A[1,j] + T[1,j-1]$
      \EndFor
      \For{$i = 2$ \textbf{to} $M$}
        \For{$j = 2$ \textbf{to} $N$}
          \State $T[i,j] = A[i,j] + \min(T[i-1,j], T[i,j-1])$
        \EndFor
      \EndFor
```

1,5

```
    end for
  end for
  return T[N,M]
end function
```

3,5

Aufgabe 10: Prove or Disprove (4 Punkte)

- a) Let $G = (V, E)$ be a connected, undirected and weighted graph. A cut $(S, V \setminus S)$ is a partition of the nodes (that is, we divide the nodes into two groups, not necessarily of equal size). A cut edge is an edge (u, v) with one node in S and one node in $V \setminus S$, that is $u \in S$ and $v \in V \setminus S$. Prove or disprove the following statement:

Assume that G has a uniquely determined minimum spanning tree. Then, for each cut $(S, V \setminus S)$ of G , there exists a uniquely determined cut edge with minimum weight.

We will prove the statement:

Given that G has a uniquely determined minimum spanning tree (MST), for each cut $(S, V \setminus S)$ of G , there exists a uniquely determined cut edge with minimum weight.

Proof:

Let T be the unique minimum spanning tree of G .

Consider an arbitrary cut $(S, V \setminus S)$ of G . According to the Cut Property of MSTs, there must exist an edge (u, v) with the minimum weight among all the cut edges, such that $u \in S$ and $v \in V \setminus S$, and this edge is included in the MST T .

Now, we need to show that this edge is unique for the given cut.

Assume, for contradiction, that there exists another edge (u', v') with the same minimum weight for the cut $(S, V \setminus S)$, where $u' \in S$ and $v' \in V \setminus S$, and $(u', v') \neq (u, v)$. Since T is an MST, it must already contain a path between u' and v' (otherwise, adding (u', v') to T would yield a smaller weight spanning tree, which contradicts the uniqueness of T).

If we add the edge (u', v') to T , we create a cycle that contains both (u, v) and (u', v') , as there is now a path between u and v and a path between u' and v' in T . Since (u, v) and (u', v') have the same weight, we can replace (u, v) with (u', v') in T , and the total weight of T remains the same, meaning that we have found another MST.

However, this contradicts our initial assumption that G has a uniquely determined minimum spanning tree. Thus, our assumption that there is another edge (u', v') with the same minimum weight for the cut $(S, V \setminus S)$ must be false. Therefore, for each cut $(S, V \setminus S)$ of G , there exists a uniquely determined cut edge with minimum weight.

1