Spannbäume, Graphdurchsuchung & kürzeste Abstände

Datenstrukturen & Algorithmen – Prüfungsvorbereitung

1️⃣ Minimaler Spannbaum (MST)

Ein **minimaler Spannbaum (Minimum Spanning Tree, MST)** ist ein Teilgraph eines ungerichteten, gewichteten Graphen, der:

- alle Knoten verbindet
- keine Zyklen enthält
- die **Summe aller Kantengewichte minimiert**

Eigenschaften:

- genau **V − 1 Kanten** bei V Knoten
- nur für **ungerichtete Graphen**
- minimiert **Gesamtgewicht**, nicht einzelne Wege

Wichtige Algorithmen:

- **Prim**
- **Kruskal**

❗ **Prüfungsrelevant** Ein MST ist **kein Algorithmus für kürzeste Wege** zwischen zwei Knoten.

2️⃣ Kürzeste Abstände berechnen

Kürzeste Abstände werden **nicht** mit einem minimalen Spannbaum berechnet.

Richtige Algorithmen:

- **Ungewichteter Graph → Breitensuche (BFS)**
- **Gewichteter Graph (keine negativen Gewichte) → Dijkstra**
- **Negative Gewichte** → Bellman-Ford (theoretisch)

Merksatz:

> Ein MST minimiert die Gesamtkosten des Netzes, nicht die Distanz zwischen zwei Knoten.

3️⃣ Tiefensuche (DFS)

Idee

Die Tiefensuche geht von einem Startknoten aus **so tief wie möglich**, bevor sie zurückgeht.

Eigenschaften:

- nutzt **Stack / Rekursion**
- garantiert **keinen kürzesten Weg**
- geeignet für Zusammenhangs- und Zyklenerkennung

DFS – Algorithmus in Worten

1. Startknoten besuchen
2. Als besucht markieren
3. Rekursiv Nachbarn besuchen
4. Zurückgehen, wenn keine Nachbarn mehr existieren

DFS – Python-Code

```
def dfs(graph, start, visited=None):
    if visited is None:
        visited = set()

    visited.add(start)
    for neighbor in graph[start]:
        if neighbor not in visited:
            dfs(graph, neighbor, visited)

    return visited
```

4️⃣ Breitensuche (BFS)

Idee

Die Breitensuche durchsucht den Graphen **schichtweise**.

Eigenschaften:

- nutzt eine **Queue (FIFO)**
- liefert den **kürzesten Weg in ungewichteten Graphen**

- sehr prüfungsrelevant

BFS – Algorithmus in Worten

1. Startknoten in Queue legen
2. Knoten entnehmen
3. Unbesuchte Nachbarn zur Queue hinzufügen

BFS – Python-Code

```python
from collections import deque

def bfs(graph, start):
    visited = set()
    queue = deque([start])
    visited.add(start)

    while queue:
        node = queue.popleft()
        for neighbor in graph[node]:
            if neighbor not in visited:
                visited.add(neighbor)
                queue.append(neighbor)

    return visited
```

5️⃣ Algorithmus-Auswahl

Problem	Algorithmus
Traversierung	DFS / BFS
Kürzester Weg (ungewichtet)	BFS
Kürzester Weg (gewichtet)	Dijkstra
Minimale Gesamtkosten	Prim / Kruskal

6️⃣ Prüfungsfalle

❌ MST liefert **keine** kürzesten Wege ✅ MST minimiert nur die **Gesamtkosten**

Merksatz:

> Ein minimaler Spannbaum verbindet alle Knoten günstig, aber sagt nichts über die Distanz zweier Knoten aus.