🟦 Dijkstra-Algorithmus

1️⃣ Grundidee

Der **Dijkstra-Algorithmus** berechnet die **kürzesten Wege** von einem Startknoten zu allen anderen Knoten in einem **gewichteten Graphen mit nicht-negativen Kantengewichten**.

- Greedy-Algorithmus
- Baut die kürzesten Distanzen schrittweise auf
- Verwendet typischerweise eine **Prioritätswarteschlange (Min-Heap)**

2️⃣ Voraussetzungen

- **Gewichteter Graph**
- **Keine negativen Kantengewichte**
- Startknoten ist bekannt

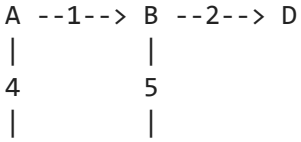
3️⃣ Laufzeiten & Eigenschaften

Eigenschaft	Wert
Laufzeit (Heap)	O((V + E) log V)
Laufzeit (Matrix, also ohne Heap)	$O(V^2)$
Speicherbedarf	O(V)
Kürzester Weg	ja
Negative Kanten	❌ nicht erlaubt

Hinweis: Bei negativen Gewichten liefert Dijkstra **falsche Resultate**.

4️⃣ Schritt-für-Schritt-Beispiel

Graph (gerichtet, gewichtet):



```
v        v
C --1--> E
```

Startknoten: **A**

Initialisierung

```
Distanz(A) = 0
Distanz(B,C,D,E) = ∞
```

Ablauf (vereinfacht)

1. A → aktualisiere B=1, C=4
2. B → aktualisiere D=3, E=6
3. D → keine Verbesserung
4. C → aktualisiere E=5

Endresultat:

```
A: 0
B: 1
C: 4
D: 3
E: 5
```

Dijkstra-Algorithmus – Algorithmus in Worten (Grundidee)

Der Dijkstra-Algorithmus ist ein Kürzester-Wege-Algorithmus für Graphen.
Er berechnet die kürzesten Wege von einem Startknoten zu allen anderen Knoten.
Der Algorithmus funktioniert nur mit nicht-negativen Kantengewichten.
Dabei wird schrittweise der aktuell kürzeste bekannte Weg festgelegt.\

Dijkstra – Algorithmus in Worten (Ablauf)

Der Algorithmus beginnt beim Startknoten.
Die Distanz zum Startknoten wird auf null gesetzt.
Alle anderen Knoten erhalten zunächst eine unendliche Distanz.
Anschließend wird der Knoten mit der aktuell kleinsten bekannten Distanz ausgewählt.
Dieser Knoten gilt nun als endgültig besucht.
Von diesem Knoten aus werden alle Nachbarknoten betrachtet.
Für jeden Nachbarn wird geprüft,
ob der Weg über den aktuellen Knoten kürzer ist als der bisher bekannte Weg.
Ist dies der Fall, wird die Distanz des Nachbarn aktualisiert.
Dieser Vorgang wird wiederholt,
bis alle Knoten besucht wurden oder keine kürzeren Wege mehr existieren.\

5️⃣ Besonderheiten / Prüfungsrelevante Hinweise

- Dijkstra ist ein **Greedy-Algorithmus**
- Ein Knoten wird „fest“ markiert, sobald seine kürzeste Distanz bekannt ist
- Sehr häufige Prüfungsfrage:
  - *Warum funktionieren negative Kanten nicht?*

6️⃣ Vor- und Nachteile

Vorteile

- effizient für viele Graphen
- liefert exakte kürzeste Wege
- weit verbreitet (Routing, Netzwerke)

Nachteile

- keine negativen Kantengewichte
- komplexer als BFS

🧠 Merksatz für die Prüfung

Dijkstra berechnet kürzeste Wege in gewichteten Graphen ohne negative Kanten mithilfe eines Greedy-Ansatzes.

7️⃣ Python-Implementierung

In [1]:
```python
import heapq

def dijkstra(graph, start):
    # graph: dict {node: [(neighbor, weight), ...]}
    distances = {node: float("inf") for node in graph}
    distances[start] = 0

    pq = [(0, start)]  # (distanz, knoten)

    while pq:
        current_dist, current_node = heapq.heappop(pq)

        if current_dist > distances[current_node]:
            continue

        for neighbor, weight in graph[current_node]:
            distance = current_dist + weight
```

```
            if distance < distances[neighbor]:
                distances[neighbor] = distance
                heapq.heappush(pq, (distance, neighbor))

    return distances


# Beispiel
graph = {
    "A": [("B", 1), ("C", 4)],
    "B": [("D", 2), ("E", 5)],
    "C": [("E", 1)],
    "D": [],
    "E": []
}

print(dijkstra(graph, "A"))
```

```
{'A': 0, 'B': 1, 'C': 4, 'D': 3, 'E': 5}
```