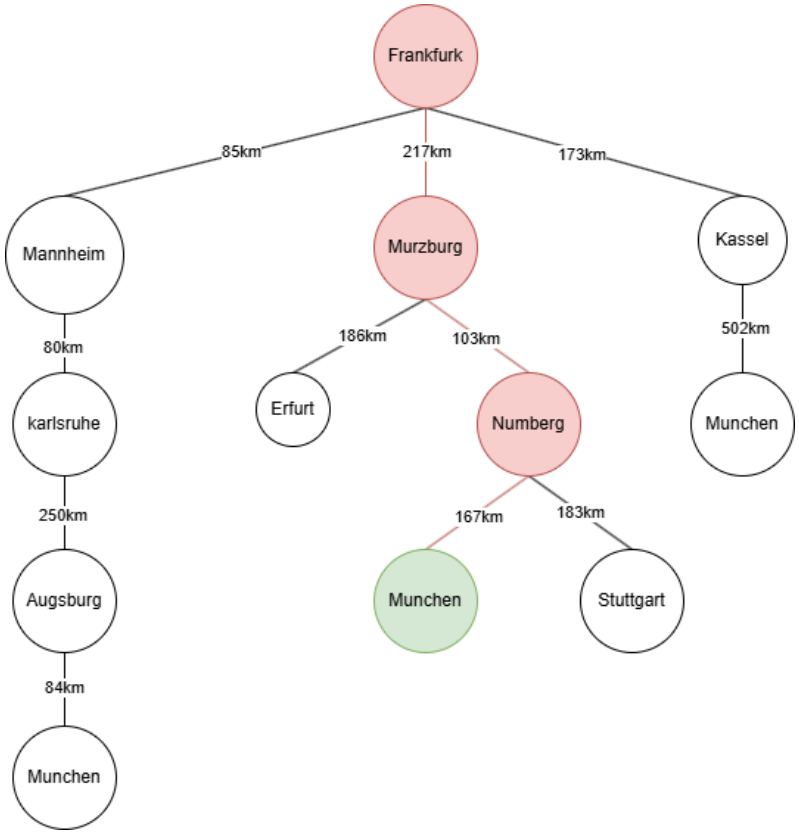
1. Construct a search tree from the state space graphs below (start state: Frankfurt, goal state: München)



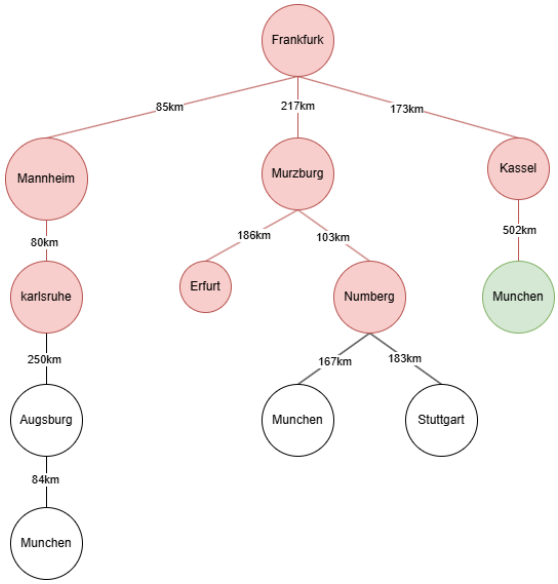
ตัวอย่าง Uniform Cost Search (UCS)

2. Find the graphical solution with these methods:

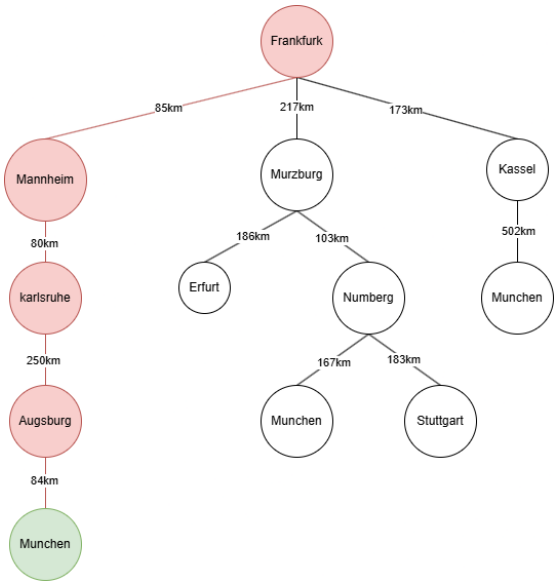
- BFS (ignore the cost – distance)

- DFS (ignore the cost – distance)

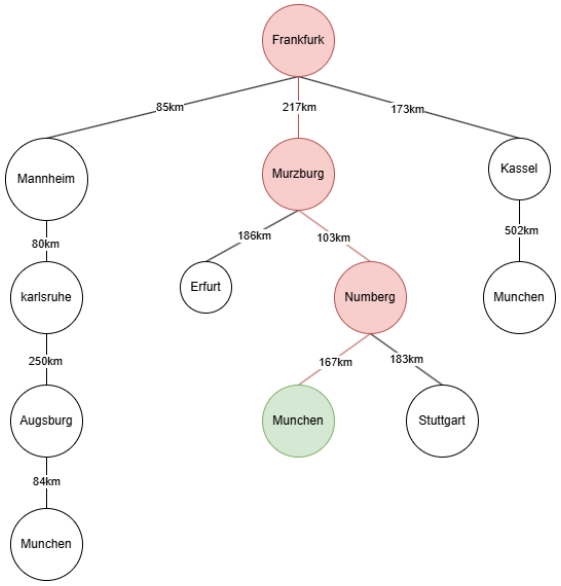
- UCS



ตัวอย่าง Breadth-First Search



ตัวอย่าง Depth-First Search (DFS)



ตัวอย่าง Uniform Cost Search (UCS)

3. Explain the advantages and disadvantages of these methods

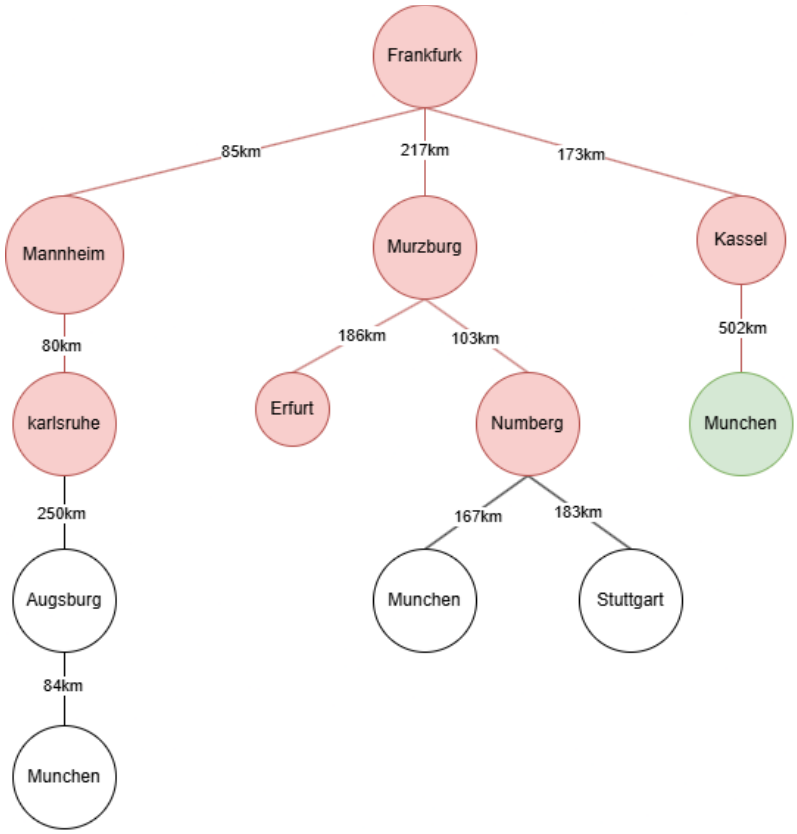
- BFS – DFS - UCS

Breadth-First Search (BFS): การค้นหาแบบกว้างก่อน

Breadth-First Search (BFS) เป็นอัลกอริทึมที่ใช้ในการสำรวจกราฟ โดยเริ่มจากจุดเริ่มต้นหนึ่งจุด แล้วค่อยๆ ขยายไปยังจุดที่ใกล้เคียงทั้งหมดในระดับเดียวกันก่อนที่จะไปยังระดับถัดไป วิธีการทำงานโดยสรุปคือ:

การทำงานของ BFS

1. เริ่มต้น: เลือกจุดเริ่มต้น (Root) และใส่ลงในคิว (Queue)
2. ขยาย: นำจุดที่อยู่ออกมาพิจารณา
3. สำรวจ: ตรวจสอบจุดที่อยู่ติดกับจุดที่นำออกมาพิจารณา หากยังไม่เคยถูกเยี่ยมชม ให้ใส่ลงในคิว
4. ทำซ้ำ: กลับไปที่ขั้นตอนที่ 2 จนกว่าคิวจะว่าง



ตัวอย่าง Breadth-First Search

ข้อดีของ BFS

- **หาเส้นทางที่สั้นที่สุด:** หากทุกๆ ขอบในกราฟมีค่าใช้จ่ายเท่ากัน BFS จะสามารถหาเส้นทางที่สั้นที่สุดจากจุดเริ่มต้นไปยังจุดอื่นๆ ได้
- **ค้นหาจุดที่อยู่ใกล้เคียงที่สุด:** BFS จะพบจุดที่อยู่ใกล้เคียงกับจุดเริ่มต้นที่สุดก่อน
- **ใช้ในการแก้ปัญหาต่างๆ:** BFS สามารถนำไปประยุกต์ใช้ในการแก้ปัญหาต่างๆ เช่น การหาส่วนประกอบที่เชื่อมต่อกัน (connected component), การตรวจสอบว่ากราฟเป็นแบบ bipartite หรือไม่, และการหาต้นไม้ครอบคลุมน้อยสุด (minimum spanning tree)

ข้อเสียของ BFS

- **ใช้หน่วยความจำมาก:** ในกรณีที่กราฟมีขนาดใหญ่ BFS อาจต้องใช้หน่วยความจำจำนวนมากในการเก็บคิว

- **อาจใช้เวลานาน:** หากกราฟมีจำนวนจุดและขอบมาก BFS อาจใช้เวลานานในการค้นหา

เมื่อไรควรใช้ BFS

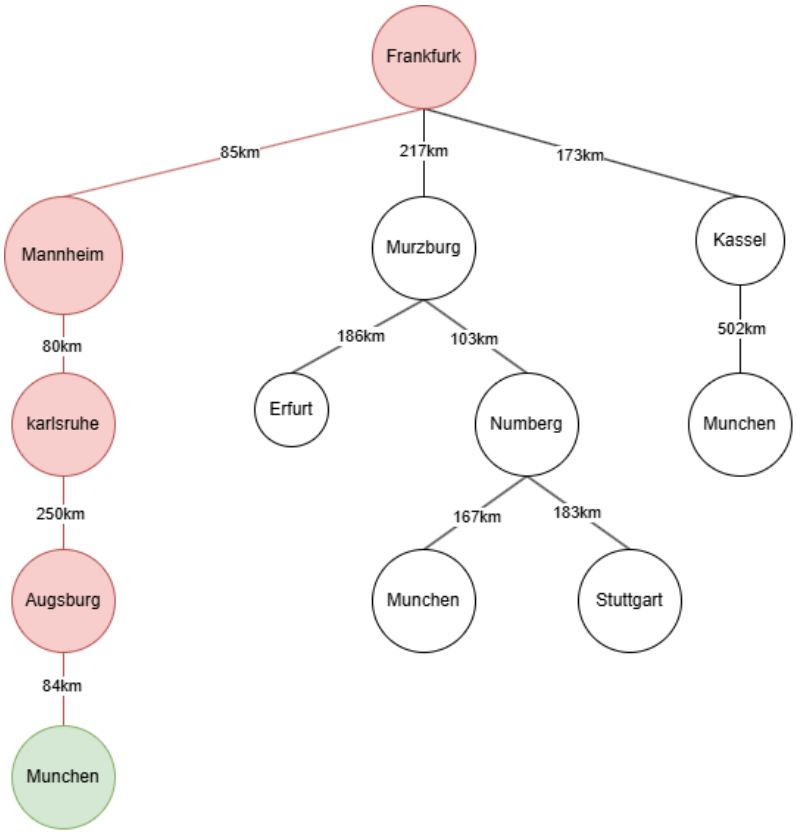
- **ต้องการหาเส้นทางที่สั้นที่สุด:** หากทุกๆ ขอบในกราฟมีค่าใช้จ่ายเท่ากัน
- **ต้องการค้นหาจุดที่อยู่ใกล้เคียงที่สุด:** เช่น การหาเพื่อนบ้านที่ใกล้ที่สุดในเครือข่ายสังคม
- **ต้องการตรวจสอบโครงสร้างของกราฟ:** เช่น การหาส่วนประกอบที่เชื่อมต่อกัน

Depth-First Search (DFS): การค้นหาแบบลึกก่อน

Depth-First Search (DFS) เป็นอีกหนึ่งอัลกอริทึมที่ใช้สำรวจกราฟ โดยมีวิธีการค้นหาที่แตกต่างจาก BFS คือ DFS จะเลือกสำรวจเส้นทางใดเส้นทางหนึ่งให้ลึกที่สุดก่อน แล้วจึงย้อนกลับมาสำรวจเส้นทางอื่นๆ

การทำงานของ DFS

1. **เริ่มต้น:** เลือกจุดเริ่มต้น (root) และใส่ลงใน stack
2. **ขยาย:** นำจุดที่อยู่ด้านบนของ stack ออกมาพิจารณา
3. **สำรวจ:** ตรวจสอบจุดที่อยู่ติดกับจุดที่นำออกมาพิจารณา หากยังไม่เคยถูกเยี่ยมชม ให้ใส่ลงใน stack
4. **ทำซ้ำ:** กลับไปที่ขั้นตอนที่ 2 จนกว่า stack จะว่าง



ตัวอย่าง Depth-First Search (DFS)

ข้อดีของ DFS

- **ใช้หน่วยความจำน้อยกว่า BFS:** เนื่องจาก DFS ต้องเก็บเพียงเส้นทางที่กำลังสำรวจอยู่ ทำให้ใช้หน่วยความจำน้อยกว่า BFS
- **หาคำตอบได้เร็ว:** หากคำตอบที่ต้องการอยู่ลึกในกราฟ DFS อาจจะพบคำตอบได้เร็วกว่า BFS
- **ใช้ในการแก้ปัญหาต่างๆ:** เช่น การตรวจสอบว่ากราฟเป็นแบบ bipartite หรือไม่, การหาจุดเชื่อม (articulation point), และการหาส่วนประกอบที่เชื่อมต่อกันอย่างเข้มแข็ง (strongly connected component)

ข้อเสียของ DFS

- **อาจไม่พบเส้นทางที่สั้นที่สุด:** DFS ไม่ได้รับประกันว่าจะพบเส้นทางที่สั้นที่สุดเสมอไป
- **อาจติดอยู่ในวงวน:** หากกราฟมีวงจร DFS อาจติดอยู่ในวงจรนั้นและไม่สามารถสำรวจกราฟทั้งหมดได้

เมื่อไรควรใช้ DFS

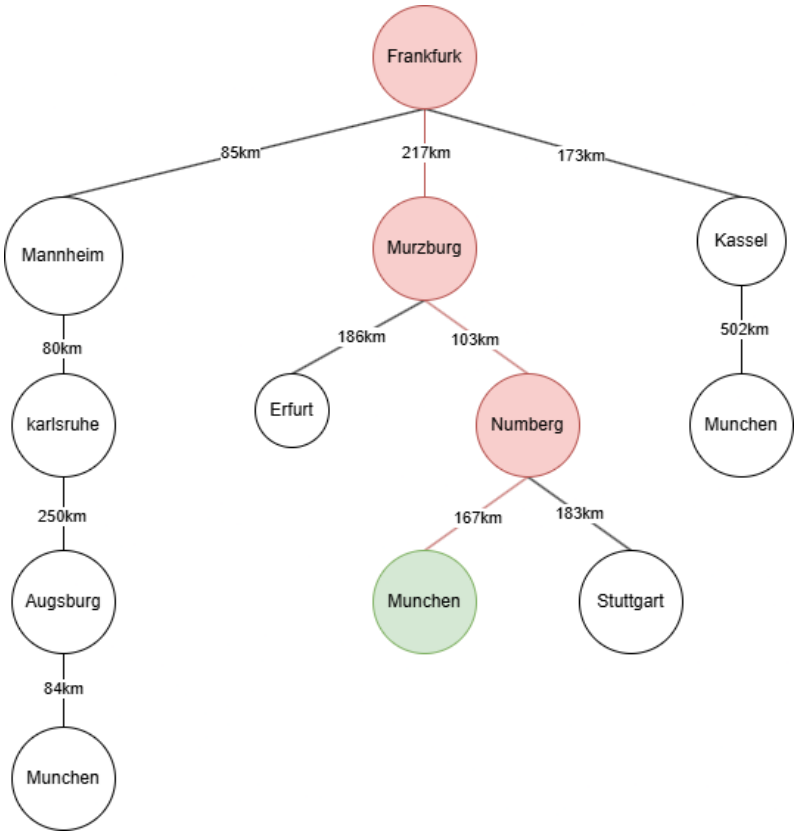
- **ต้องการหาคำตอบที่อยู่ลึกในกราฟ:** เช่น การหาจุดปลายทางในเกมที่มีหลายระดับ
- **ต้องการตรวจสอบโครงสร้างของกราฟ:** เช่น การหาส่วนประกอบที่เชื่อมต่อกันอย่างเข้มแข็ง
- **ต้องการใช้หน่วยความจำน้อย:** เมื่อกราฟมีขนาดใหญ่มาก

Uniform Cost Search (UCS): การค้นหาแบบค่าใช้จ่ายน้อยที่สุด

UCS เป็นอัลกอริทึมที่ใช้ในการค้นหาเส้นทางที่มีค่าใช้จ่ายรวมน้อยที่สุดในกราฟ โดยค่าใช้จ่ายนี้ อาจเป็นระยะทาง เวลา หรือค่าใช้จ่ายอื่นๆ ก็ได้ UCS จะสำรวจเส้นทางต่างๆ โดยพิจารณาจากค่าใช้จ่ายสะสมจากจุดเริ่มต้นไปยังจุดปัจจุบัน

วิธีการทำงานของ UCS

1. **เริ่มต้น:** เริ่มจากจุดเริ่มต้น (node) และใส่ลงในโครงสร้างข้อมูลที่เรียงลำดับตามค่าใช้จ่าย (โดยทั่วไปใช้ priority queue)
2. **ขยาย:** เลือก node ที่มีค่าใช้จ่ายสะสมน้อยที่สุดออกมาพิจารณา
3. **สำรวจ:** พิจารณา node ที่อยู่ติดกับ node ที่เลือกมา หากยังไม่เคยถูกเยี่ยมชม ให้คำนวณค่าใช้จ่ายสะสมใหม่ และใส่ลงในโครงสร้างข้อมูล
4. **ทำซ้ำ:** กลับไปที่ขั้นตอนที่ 2 จนกว่าจะถึงจุดหมายปลายทาง หรือจนกว่าโครงสร้างข้อมูลว่าง



ตัวอย่าง Uniform Cost Search (UCS)

ข้อดีของ UCS

- **การันตีเส้นทางที่ถูกที่สุด:** หากมีเส้นทางที่ไปถึงจุดหมายปลายทางได้ UCS จะสามารถหาเส้นทางที่มีค่าใช้จ่ายรวมน้อยที่สุดได้เสมอ
- **เหมาะสำหรับปัญหาที่ค่าใช้จ่ายของแต่ละขอบแตกต่างกัน:** UCS สามารถจัดการกับกราฟที่มีค่าใช้จ่ายของแต่ละขอบไม่เท่ากันได้ดี

ข้อเสียของ UCS

- **ใช้ทรัพยากรมาก:** ในกราฟขนาดใหญ่ UCS อาจต้องใช้เวลาและหน่วยความจำจำนวนมาก เนื่องจากต้องสำรวจทุกเส้นทางที่เป็นไปได้
- **ไม่เหมาะสำหรับปัญหาที่ต้องการคำตอบโดยประมาณ:** หากต้องการคำตอบที่ "ค่อนข้างถูก" แต่เร็วกว่า UCS อาจไม่ใช่ตัวเลือกที่ดีที่สุด

เมื่อไหร่ควรใช้ UCS?

- **ปัญหาการค้นหาเส้นทาง:** เช่น การหาเส้นทางที่สั้นที่สุดบนแผนที่, การหาเส้นทางที่ใช้เวลาน้อยที่สุดในการเดินทาง
- **ปัญหาการวางแผน:** เช่น การวางแผนการผลิต, การวางแผนการขนส่ง