



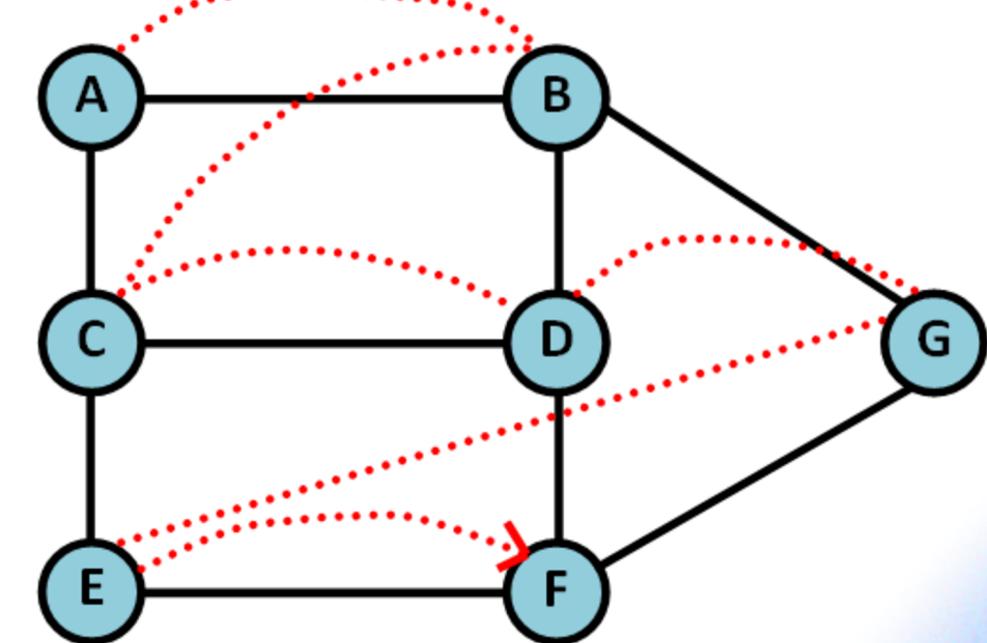
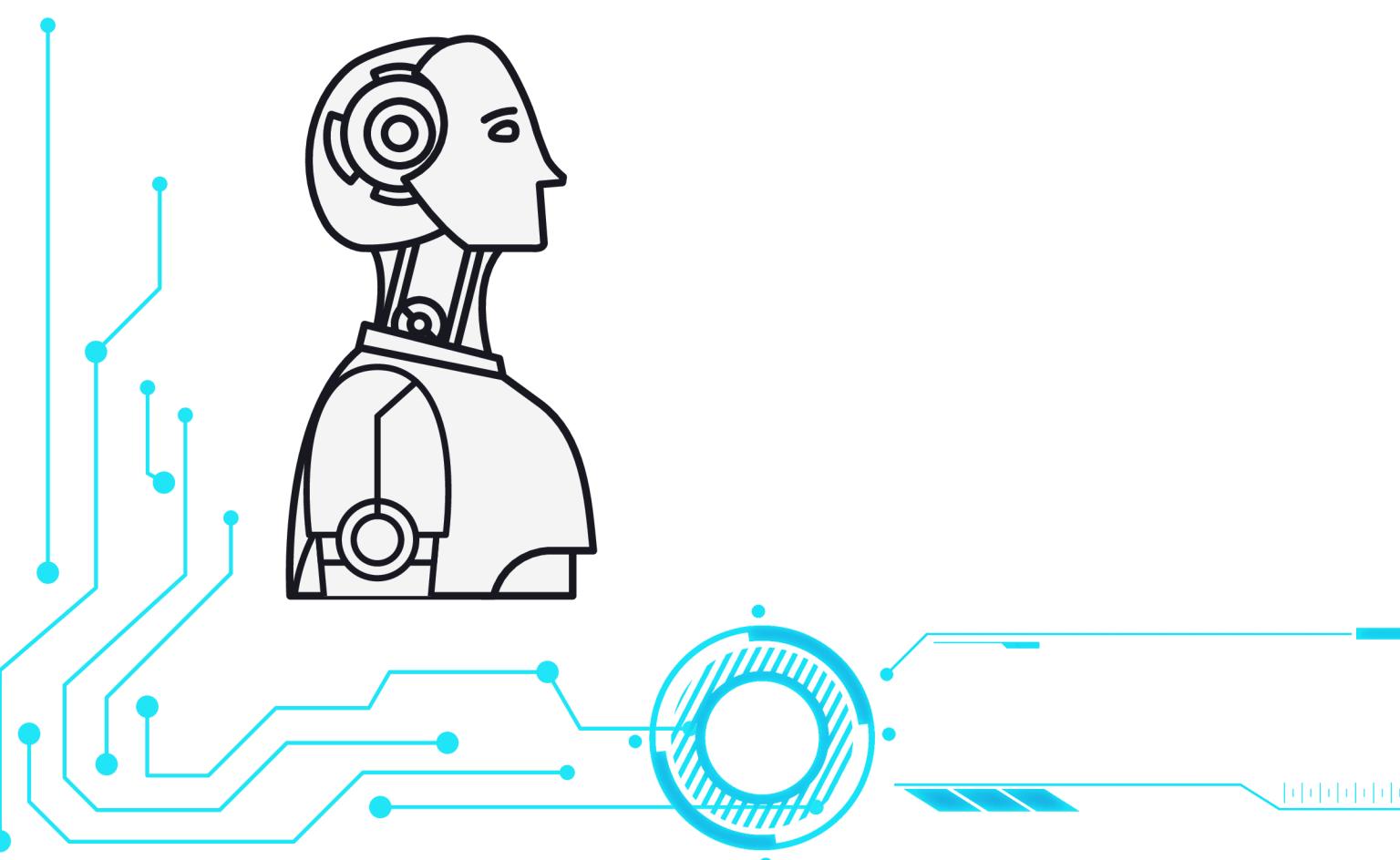
Single-Source Shortest Paths (SSSP) ແລະ BFS(s)



มาทำความรู้จักกับ

Single-Source Shortest Paths (SSSP)

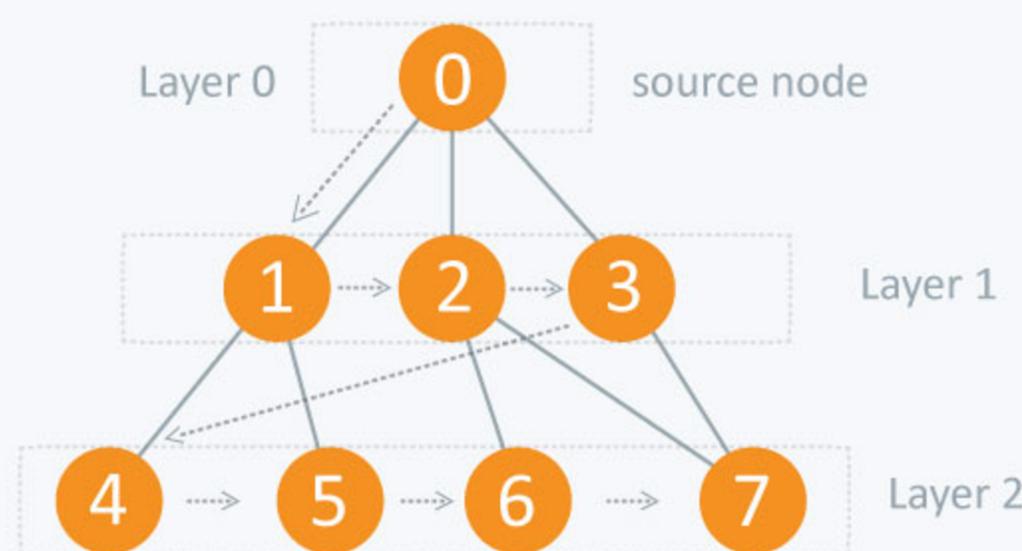
ปัญหาคลาสสิกในทฤษฎีกราฟ ที่มีเป้าหมายในการค้นหาเส้นทางที่สั้นที่สุด จากจุดเริ่มต้น (source) ไปยังจุดอื่นๆ ทั้งหมดในกราฟ หนึ่งในอัลกอริทึมที่นิยมใช้ในการแก้ปัญหานี้คือ Breadth-First Search (BFS) ที่เริ่มจากจุดต้นทาง และคืบหาไปยังจุดที่อยู่ใกล้ที่สุดก่อนทีละชั้น



เวลาการทำงาน BFS

- เวลาการทำงาน: $O(n + E)$
- n = จำนวนโหนด
- E = จำนวนเส้นเชื่อม

Breadth-First Search (BFS)

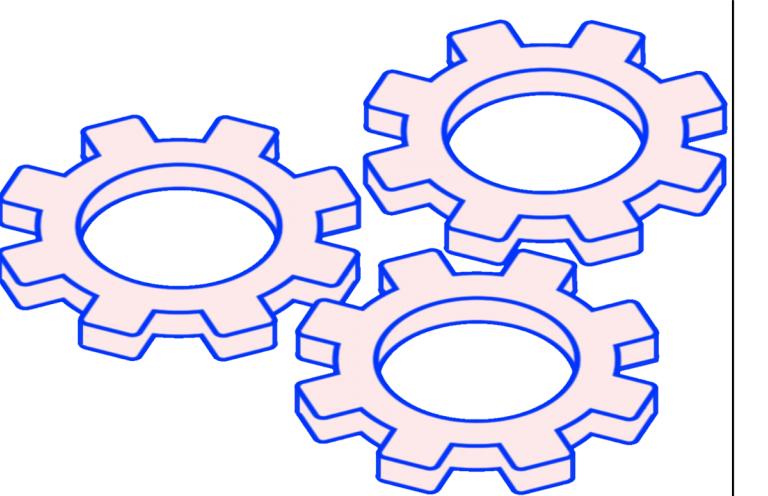


เวลาการทำงาน BFS

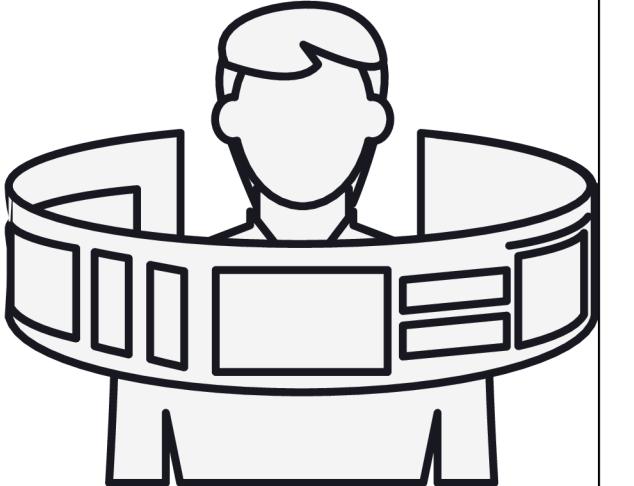
- เวลาการทำงาน: $O(n + E)$
- n = จำนวนโหนด
- E = จำนวนเส้นเชื่อม

- เริ่มจากจุดต้นทาง (s) และกำเครื่องหมายว่าได้เยี่ยมชมแล้ว
- เพิ่มจุดต้นทางลงในคิว
- นำจุดแรกออกจากคิว และสำรวจจุดที่อยู่ติดกันกึ้งหมดที่ยังไม่ได้เยี่ยมชม
- กำเครื่องหมายจุดที่อยู่ติดกันว่าได้เยี่ยมชมแล้ว และเพิ่มลงในคิว
- กำช้ำขั้นตอน 3 และ 4 จบกว่าคิวจะว่าง

BFS รับประกันว่าถ้าเจอโหนดเป้าหมายก่อน จะได้ระยะทางที่สั้นที่สุดเสมอ

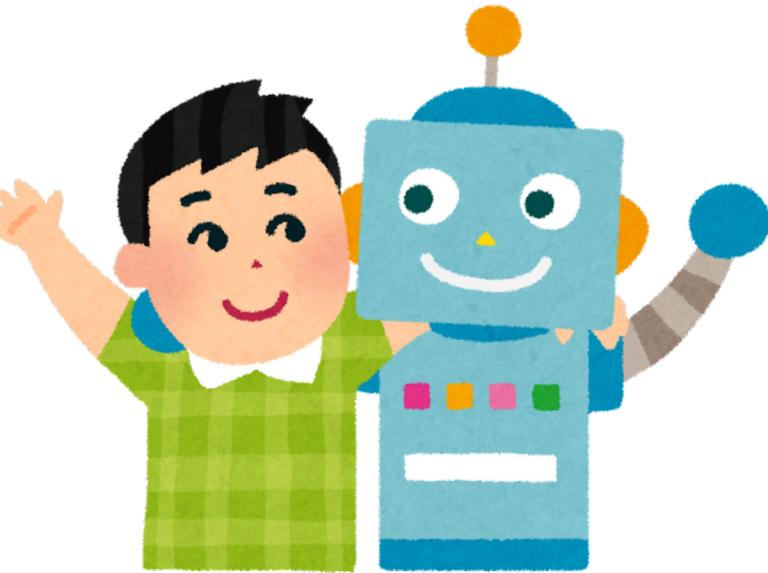


ตัวอย่างการนำไปใช้



เช่น เกมแนว Puzzle, Board Game

การหาเส้นทางที่สั้นที่สุดในเกม: เช่น การหาเส้นทางที่สั้นที่สุดเพื่อให้ตัวละครไปถึงจุดหมาย
การหาจุดที่อยู่ใกล้กันที่สุดในเครือข่าย: เช่น การหาเซิร์ฟเวอร์ที่อยู่ใกล้ผู้ใช้ที่สุด
การวิเคราะห์เครือข่ายสังคม: เช่น การหาเพื่อนที่อยู่ใกล้กันที่สุดในเครือข่าย



ตัวอย่าง Code

- ในการทำงาน ใช้ Queue เพื่อจัดลำดับการสำรวจโหนด โดยที่พับครึ่งแรกจะถูกเพิ่มเข้าไปในคิวพร้อมกำหนดระยะทาง
- ผลลัพธ์ของ โค้ดจะแสดงระยะทางที่สั้นที่สุดจากโหนดเริ่มต้นไปยังโหนดอื่น ๆ

```
from collections import deque

def bfs_shortest_path(graph, start):
    # สร้าง dictionary สำหรับเก็บระยะทางจากจุดเริ่มต้นไปยังโหนดต่าง ๆ
    distances = {node: float('inf') for node in graph}
    distances[start] = 0

    # สร้าง queue สำหรับ BFS และเพิ่มจุดเริ่มต้นเข้าไป
    queue = deque([start])

    while queue:
        current = queue.popleft()

        # ตรวจสอบเพื่อนบ้านของโหนดปัจจุบัน
        for neighbor in graph[current]:
            if distances[neighbor] == float('inf'): # ยังไม่เคยเยี่ยมชม
                distances[neighbor] = distances[current] + 1
                queue.append(neighbor)

    return distances
```

สรุป

BFS เป็นอัลกอริทึมที่มีประสิทธิภาพและง่ายต่อการใช้งานในการแก้ปัญหา SSSP ในกรณีที่กราฟไม่มีน้ำหนักหรือมีน้ำหนักเท่ากัน

```
# ตัวอย่างกราฟ (แบบไม่มีน้ำหนัก)
graph = {
    'A': ['B', 'C'],
    'B': ['A', 'D', 'E'],
    'C': ['A', 'F'],
    'D': ['B'],
    'E': ['B', 'F'],
    'F': ['C', 'E']
}

# เรียกใช้งาน BFS
start_node = 'A'
shortest_paths = bfs_shortest_path(graph, start_node)

# แสดงผล
print(f"Shortest distances from node {start_node}:")
for node, distance in shortest_paths.items():
    print(f"{node}: {distance}")
```