DP(s)

โปรแกรมเชิงพลวัตสำหรับการหาเส้นทางที่สั้นที่สุดจากต้นทาง

นายวศิน เที่ยงดี 6710301028
นายวรุตม์ เที่ยงดี 6710301029
นางสาวณัฐธอัญ ศิรินันทศรณ์ 6710301035
นางสาวฟ้ารุ่ง กิจสวัสดิ์ 6710301053

Introduction to Dynamic Programming (DP)

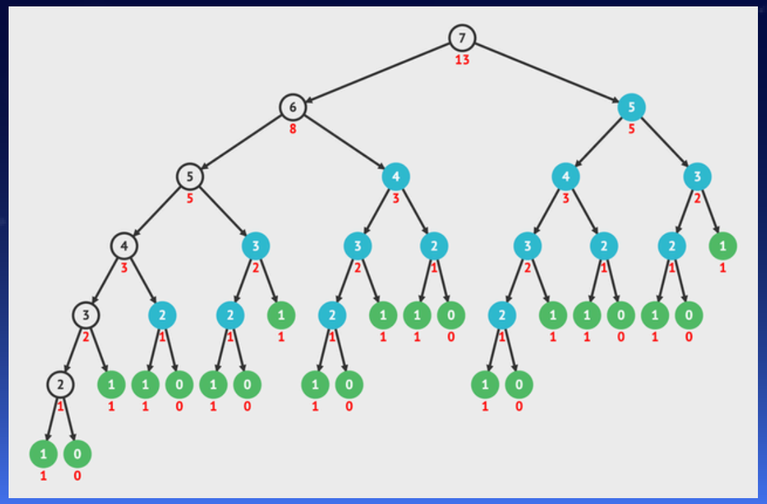
Dynamic Programming (DP) คือ เทคนิคในการแก้ปัญหาโดยการแบ่งปัญหาใหญ่ออกเป็นปัญหาย่อย (subproblems) และแก้ปัญหาย่อยเหล่านั้นเพียงครั้งเดียว แล้วนำผลลัพธ์กลับมาใช้ซ้ำ (reuse) เพื่อเพิ่มประสิทธิภาพ

หลักการสำคัญ:

1. **Optimal Substructure**: ผลลัพธ์ของปัญหาใหญ่สามารถสร้างได้จากผลลัพธ์ของปัญหาย่อย
2. **Overlapping Subproblems**: ปัญหาย่อยซ้ำกันหลายครั้ง จึงสามารถเก็บผลลัพธ์ไว้เพื่อใช้ซ้ำได้ (Memoization หรือ Tabulation)

ตัวอย่างปัญหาที่แก้ได้ด้วย DP:

- Fibonacci Sequence
- Knapsack Problem
- Longest Common Subsequence

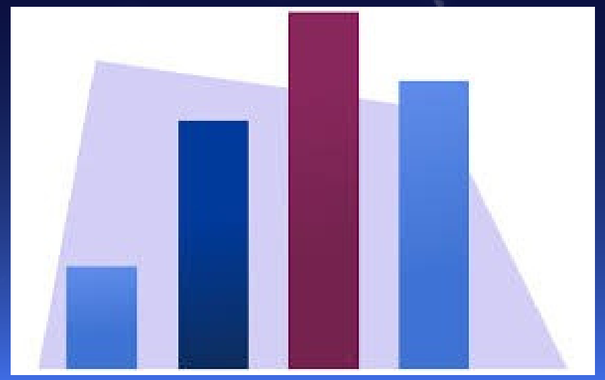


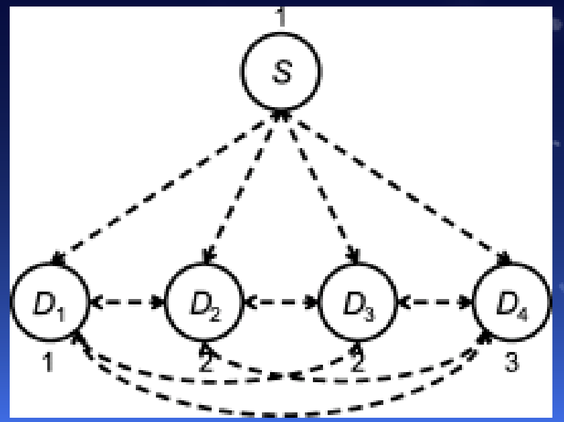
Single-Source Shortest Paths (SSSP)

ปัญหา SSSP คือการหาค่าระยะทางที่สั้นที่สุดจากโหนดต้นทาง (source) ไปยังโหนดทั้งหมดในกราฟ

ประเภทของกราฟที่ใช้:
- กราฟมีน้ำหนัก
- กราฟอาจมีน้ำหนักติดลบ

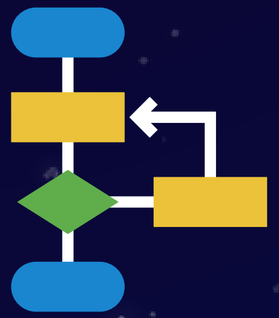
วิธีแก้ปัญหาแบบดั้งเดิม:
- **Dijkstra's Algorithm**: ใช้สำหรับกราฟที่ไม่มีน้ำหนักติดลบ
- **Bellman-Ford Algorithm**: ใช้ได้กับกราฟที่มีน้ำหนักติดลบ





Dynamic Programming Approach for SSSP

Dynamic Programming เป็นอีกวิธีหนึ่งในการแก้ปัญหา SSSP โดยใช้ตารางหรืออาร์เรย์เพื่อเก็บค่าระยะทางที่สั้นที่สุด



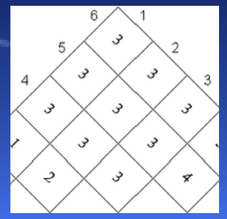
ขั้นตอนการทำงาน:

1. **Initialization**: กำหนดค่าระยะทางเริ่มต้นของโหนดต้นทาง (source) เป็น 0 และโหนดอื่น ๆ เป็น Infinity
2. **Relaxation**: ใช้การอัพเดตค่าระยะทางระหว่างโหนดซ้ำ ๆ โดยพิจารณาเส้นทางที่เหมาะสมที่สุด
3. **Termination**: ทำซ้ำจนกว่าค่าระยะทางจะไม่เปลี่ยนแปลงอีก หรือครบจำนวนรอบที่กำหนด

ตัวอย่าง Pseudocode

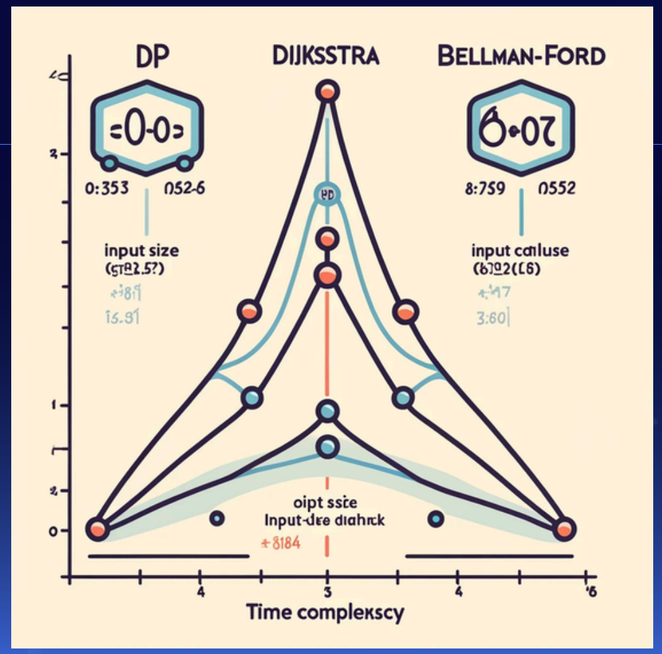
```
# n = จำนวนโหนด, edges = รายการของเส้นเชื่อม (u, v, weight)
dist = [Infinity] * n
dist[source] = 0

for i in range(n-1):
    for (u, v, weight) in edges:
        if dist[u] + weight < dist[v]:
            dist[v] = dist[u] + weight
```



Complexity Analysis

- **Time Complexity**: O(V * E)
- V = จำนวนโหนด (vertices)
- E = จำนวนเส้นเชื่อม (edges)
- **Space Complexity**: O(V) สำหรับการเก็บระยะทาง



Comparison with Other Algorithms

Algorithm	Time Complexity	Space Complexity	ใช้กับน้ำหนักติดลบได้
Dijkstra	O(V + E log V)	O(V)	ไม่ได้
Bellman-Ford	O(V * E)	O(V)	ได้
DP (Tabulation)	O(V * E)	O(V)	ได้

Applications

- การคำนวณเส้นทางในแผนที่: เช่น Google Maps ใช้หาเส้นทางที่สั้นที่สุดระหว่างเมือง
- การวิเคราะห์เครือข่าย: เช่น การหาเส้นทางที่สั้นที่สุดในเครือข่ายคอมพิวเตอร์
- ปัญหาในวิทยาการข้อมูล (Data Science): เช่น การวิเคราะห์กราฟทางสังคม

ข้อดี และ ข้อเสีย ของ DP(s)

ข้อดี

- โครงสร้างง่ายเมื่อเทียบกับอัลกอริทึมอื่น
- สามารถใช้งานได้กับกราฟที่มีน้ำหนักติดลบ

ข้อเสีย

- ประสิทธิภาพต่ำกว่าวิธีที่ปรับปรุงแล้ว เช่น Dijkstra (กรณีไม่มีน้ำหนักติดลบ)
- ใช้พื้นที่หน่วยความจำมากในกราฟขนาดใหญ่

Thank You