Intelligent Agents

agent คืออะไรก็แล้วแต่ เช่น robot ที่มีชีวิตในสิ่งแวดล้อมที่กำหนด, bot

Intelligent	Autonomous
หุ่นยนต์ที่ฉลาด แต่ยังมนุษย์ต้องเข้าไปควบคุม	หุ่นยนต์ที่ปล่อยทำได้เลย ไม่ต้องมีมนุษย์ เช่น หุ่นยนต์ดูดฝุ่นเสร็จแล้วกลับไปแท่นชาร์จได้ด้วยตัวเอง

agent ไปอยู่ใน env บางอย่าง

Environment -: Discrete vs. Continuous

- Episodic
- Accessible
  - local รู้แค่ที่อยู่ตัวเองอยู่
  - global รู้ว่า L, R มีฝุ่นไหม?
  - none

ตัวอย่าง Vacuum World

มีคำสั่ง operation => LEFT, RIGHT, SUCK

ในโลก continuous หุ่นยนต์อยู่พิกัด x = 0.5 y = 2

discrete คือสถานหุ่นยนต์ L, R, S

- static vs. dynamic vs. Semi

env ไม่เปลี่ยน หุ่นหมดไฟ ยังไม่ทำอะไรเลย

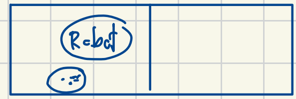
env ไม่เปลี่ยน แต่มาตรวัดเปลี่ยน เช่น เสื่อมสภาพ policy เปลี่ยน

- Deterministic

การที่หุ่นยนต์ (agent) รู้ว่าทำ action นี้แล้ว ผลลัพธ์เป็นไง

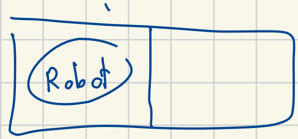
- Non-Deterministic ไม่รู้ผลลัพธ์จากการ action
  เช่น สั่งดูดฝุ่น ฝุ่นไม่หมด

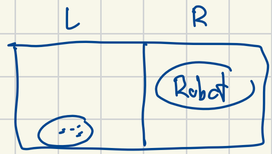
สั่ง LEFT



⇓

SUCK ฝุ่นหมด จะเป็น deterministic



Accessibility	Deterministic (สั่งแล้วทำ)	
Yes	Yes	 Single state = ⌜LEFT, SUCK⌟

Yes
อยากรู้อยู่ไหนได้
	No
	สั่งซ้าย อาจไม่ไปซ้าย
		while not in LEFT_ROOM:
		    LEFT
		while not CLEAN:
		    SUCK

Accessibility

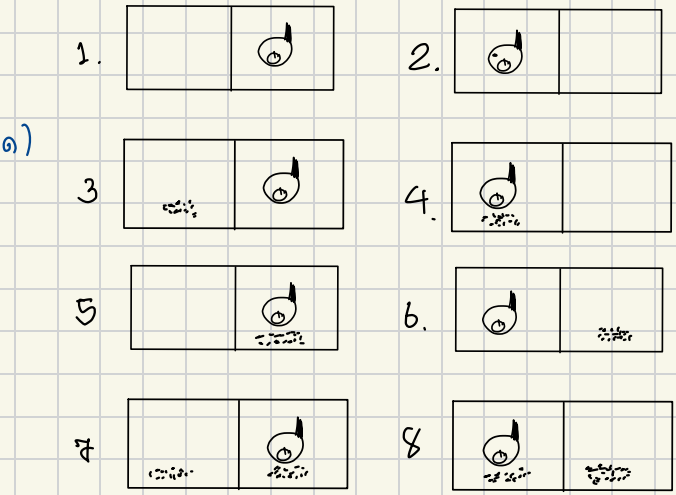
No

Deterministic

Yes

ไม่รู้ว่าอยู่ห้องไหน แต่ทำได้ ว่าไปไหน single state แล้ว

จำนวน state ที่เป็นไปได้ ถังขยะต่อดูดฝุ่น



[1,2,3,4,5,6,7,8]

↓ SUCK (ผลการดูด)

[1,2,3,6]

↓ LEFT

ไม่ว่าเป็นต่อง ดูใน policy (list)

[2,4,6]

↓ SUCK

[2]

เพราะ ③ สั่ง LEFT จะเป็น ④

No

No

ไม่รู้ว่าอยู่ไหน และก็ขยับไปไม่ได้ ต้องความจำ

Problem Solving and Search

Problem Solving

Algorithm => เมื่อปัญหาเปลี่ยน อัลกอเปลี่ยน

AI => เมื่อปัญหาเปลี่ยน อัลกอไม่เปลี่ยน ที่เปลี่ยนคือ state กับ operation

8-Puzzle ปัญหาเลื่อนแผ่นเลข

3	2	7
1	8	6
4	5	

Start State

=>

1	2	3
4	5	6
7	8	

Goal State

=> Problem definition

① state ควร define อย่างไร

State : [3, 2, 7, 1, 8, 6, 4, 5, 0] → i % 3, i // 3

[[3, 2, 7], [1, 8, 6], [4, 5, 0]]

Operation : เราคือช่องว่าง 0 พอ เลื่อน L, R, U, D

Path cost : 1 per move

Goal : [1, 2, 3, 4, 5, 6, 7, 8, 0]

Block World

State : [['A','B','C'],['D']]

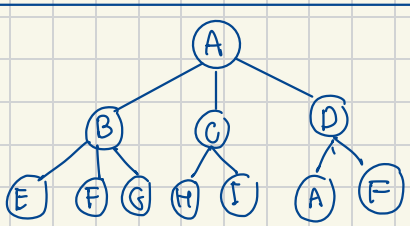
Operation : move the free box to another stack or the floor

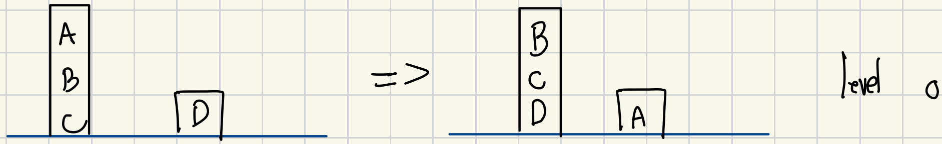
Goal : [['D','C','B','A']]

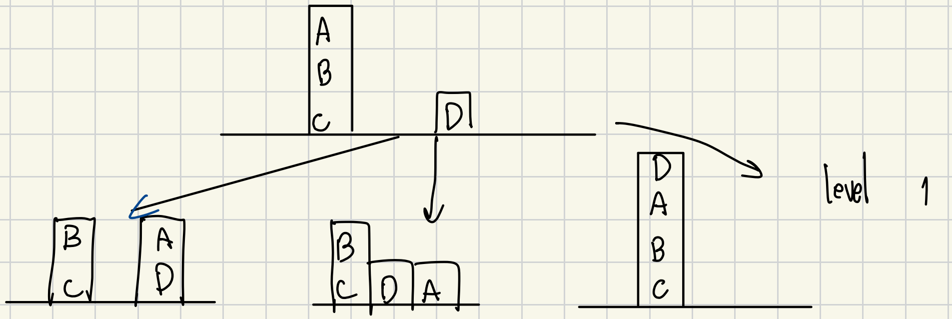
Path cost : 1 per move

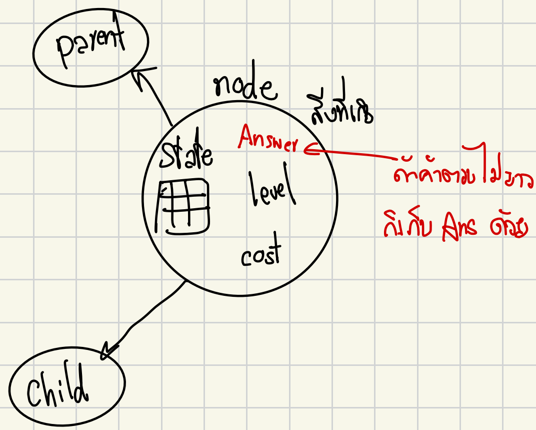
State : [['','B'],['A','C'],['B','Floor'],['','Floor']]
A B C D

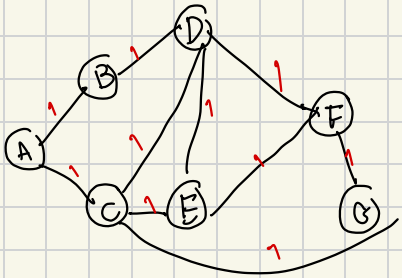
ถ้าอันแรกเป็น empty แสดงว่าต้องเอาไปวางบน empty อันนั้น หรือวางบนพื้น




A
B
C
D
=>
B
C
D
A
level 0


A
B
C
D
level 1
B
C
A
D
B
C
D
A
D
A
B
C


parent
node
สิ่งที่เก็บ
State
Answer
level
cost
ถ้าค่าตามไม่ตรง
ก็เก็บ Ans ด้วย
Child



A → G

General Search

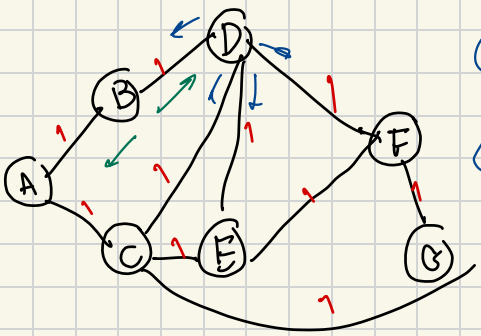
Fringe = [] เก็บ leaf node ที่ยังมองดูของต้นไม้

① เอา initial node ใส่ใน Fringe list

แต่ใส่ initial state ลงใน initial node ก่อนค่อยใส่ fringe list

```
while TRUE:
    if fringe is empty: return FAILURE ไม่เหลือ leaf node
    front ← remove_front(fringe)  fringe.pop(0)
    if is_goal(front) return solution
    insert_all(gen_successor(front), fringe)
```

(



goal = G

front ← ไม่ empty ต้องออก fringe

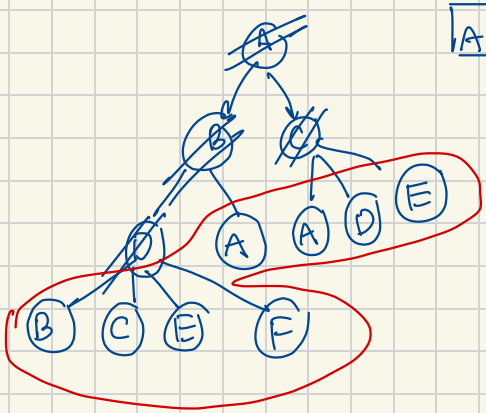
A | A

B | B | C

C | C | D | A

D | D | A | A | D | E

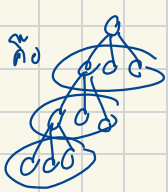
A | A | D | E | B | C | E | F



ที่ทำทั้งบนนี้เรียกว่า Breadth - First - Search (BFS)

การวิเคราะห์อัลกอริทึม

b : Branching Factor => จำนวน children ที่สามารถเกิดขึ้นสูงที่สุด b = 3 คือ



d : depth ของ Least cost solution

m : Maximum depth

3 ตัวชี้วัดประสิทธิภาพของอัลกอริทึม

① Optimality เจอคำตอบที่ดีที่สุด

② Completeness เจอคำตอบหรือไม่

③ Time Complexity วัดเป็น Big O วัดจำนวน node ที่ถูก generate => O(?)

④ Space Complexity จำนวน node สูงสุดที่ต้องเก็บ

เมื่อ b = 3

Optimality : yes

Completeness : yes

จำนวน node ที่ expand คือ node ที่อยู่บน △ ขวา

level ของ least cost solution $b^0+b^1+b^2+\dots+b^{(b^d)}$

Time complexity $O(b^d)$

Space $= O(b^{d+1}) = O(b \cdot b^d)$

8 - Puzzle

b = 4 เพราะเลื่อนได้ 4 ทิศ

1	6	8
2	4	
5	7	3

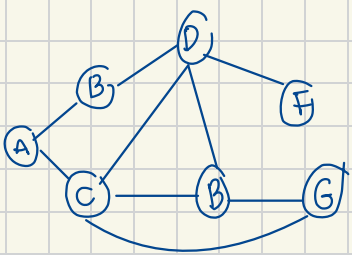
d = 40 ถ้าต้องการหาคำตอบที่ลึกสุดเลยว่า 5 ครั้ง

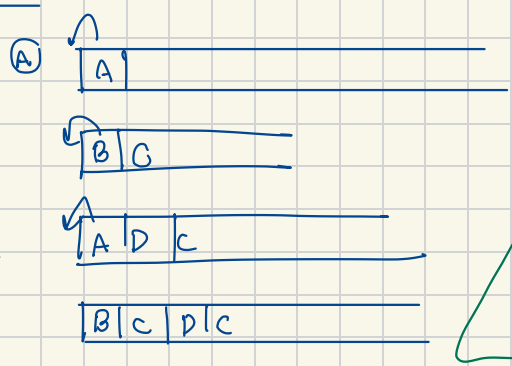
$4^{40} = 2^{80} \times 2^3 = 2^{83}$ bytes

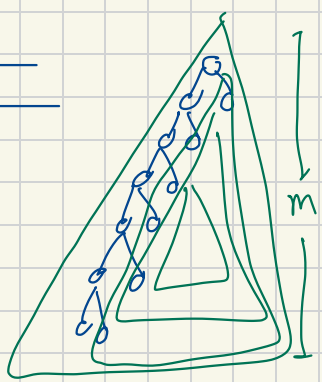
$= \frac{2^{83}}{2^{30}}$ GB

$= 2^{53}$ GB

Depth - First - Search (DFS)

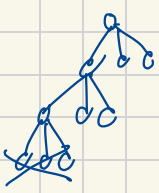






Optimality : No ถ้าคำตอบอยู่ด้านขวาสุดๆ ก็

Completeness : (Yes ถ้ามีการ check repeated state + maximum depth)



ถ้าไม่มี No

Time Complexity : $O(b^m)$

$4^{50} = \frac{2^{100}}{2^{30}}$ s $= 2^{70}$ s

$= \frac{2^{76}}{2^{26}} = 2^{44}$ ปี

Space Complexity : $O(bm)$

$\frac{2^{70}}{2^6 \text{นาที} \times 2^6 \times 2^5 \times 2^9}$