ARTIFICIAL INTELLIGENCE – Midterm Summary

Intelligence หรือ ความอัจฉริยะ หมายถึง ความสามารถในการคิด ทำความเข้าใจ และเรียนรู้สิ่งต่างๆ ได้ หรือคือไม่ได้ทำจากสัญชาตญาณ (Instinct) ดังนั้น Artificial Intelligence จึงหมายถึง เครื่องจักรหรือซอฟท์แวร์ที่มีความอัจฉริยะซึ่งถูกสร้างโดยมนุษย์ สำหรับยุคของ AI นั้น จะแบ่งเป็นยุคหลักๆ ทั้งหมด 7 ยุค คือ

Era	Year	Idea
The birth of AI	1943 – 1956	- Turing Test - Artificial Intelligence (Machine Intelligence + ANN + Automata Theory) - Chess Game Programming
The rise of AI	1956 – late 1960s	- LISP (High-Level Programming Language) - General Problem Solver
The disillusionment of AI	late 1960s – early 1970s	- General methods for broad classes of problems
Discovery of Expert System	early 1970s – mid 1980s	- DENDRAL - MYCIN (Medical Expert System) - EMYCIN - PROSPECTOR - PROLOG
Evolutionary Computation	early 1970s – NOW	- Genetic Algorithm/Programming - Evolutionary Strategies
The rebirth of neural network	mid 1980s – NOW	(CPU was powerful enough.)
Computing with words	late 1980s – NOW	- Fuzzy Theory

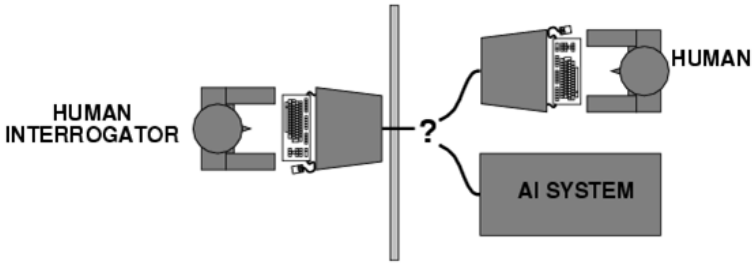
เมื่อเทียบระหว่าง Human Intelligence และ Artificial Intelligence แล้วนั้น จะพบว่าจุดเด่นของ AI ก็คือ Memory, Knowledge Transferring (Fast) and Representation (Docs), No biased ส่วนจุดที่ AI ไม่สามารถทำได้คือ Creativity, Instinct Deduction, Experience (ต้องมีข้อมูลหรือสารสนเทศเข้าไป) โดยเมื่อนำ AI กับ Human Intelligence มาเปรียบเทียบกัน จะสามารถแบ่ง AI ได้ 4 แบบ คือ

Thinking Humanly (Think like human)	Learning Algorithm
Acting Humanly (Act like human)	Robotics
Thinking Rationally (Logical Deduction)	Expert System
Acting Rationally (Logical Action)	Agents

และเมื่อเปรียบเทียบระหว่างโปรแกรมทั่วไป และโปรแกรมที่ Implement AI เข้าไป จะพบว่าโปรแกรมที่มี AI ไม่จำเป็นที่ต้องรับ Input ที่ครบถ้วนเหมือนโปรแกรมทั่วไป และไม่ต้อง Implement Algorithm เฉพาะทางเข้าไป เพราะจะมีส่วน Learning Algorithm เข้าไปช่วยในการตัดสินใจ

Turing Test

คิดค้นโดย Alan Turing ด้วยแนวคิดที่ต้องการแยก Machine กับ Human ออกจากกัน ซึ่งวิธีการดังกล่าวเรียกว่า Turing Imitation คือการให้คน 2 คน ซึ่งเป็นหญิงและชายแยกกัน แล้วเข้าใช้งานโปรแกรมพูดคุยกัน โดยในช่วงแรกจะพูดคุยกันแบบปกติ แล้วผู้ชายจะถูกแทนที่ด้วย Machine ให้คุยโต้ตอบกับผู้หญิงต่อไป หาจับไม่ได้ แปลว่า Machine ดังกล่าวมี Intelligence จริง



โดยจะมีรางวัลให้กับ Chat Bot ที่ทำได้ใกล้เคียงกับมนุษย์ได้มากที่สุดเรียกว่า Loebner Prize ซึ่งในปี 2017 มี Chat Bot ที่ได้รับรางวัลคือ Mitsuku

Chinese Room

คิดค้นโดย John Searle ด้วยแนวคิดที่ว่าเมื่อ Machine ผ่าน Turing Test แล้วนั้น ก็ยังคงไม่เพียงพอต่อการตัดสินได้ว่ามี Intelligence ดั้งนั้นจึงคิดการทดสอบ (จำลอง) ขึ้นมา โดยจะใช้วิธีว่ามีคนอยู่ 2 คน คนหนึ่งไม่รู้ภาษาจีน อยู่ในห้องที่มีหนังสือจะมีวิธีการตอบสัญลักษณ์ภาษาจีนต่างๆ อยู่ แล้วจะให้อีกคนที่รู้ภาษาจีนส่งสัญลักษณ์ภาษาจีนเข้ามาให้คนที่อยู่ข้างในตอบ แต่อย่างไรก็ดี การทดลองนี้ Searle ได้สรุปว่า ไม่มี AI ที่สามารถเข้าใจภาษาจีนได้จริงๆ

PROBLEM SOLVING & STATE SPACE

หลักการแก้ไขปัญหา (Problem Solving) นั้น ประกอบไปด้วยสถานะ (State) และการเปลี่ยนสถานะ (Transition) โดยจะต้องมีการระบุ Initial State หรือสถานะเริ่มต้น และ Goal State หรือสถานะเป้าหมาย ซึ่งสิ่งที่ AI จะทำได้นั้น คือการเปลี่ยนสถานะ ซึ่งจะต้องมีการกำหนดนิยามของ Transition ก่อน เรียกว่า Operator หรือว่าพูดให้ง่ายๆ ก็คือ กฎของ AI

โปรแกรม AI จะไม่คำนวณจาก Algorithm ในการแก้ไขปัญหาเฉพาะทาง แต่จะใช้ Data Structure อย่าง Graph Theory หรือ Tree เข้ามาช่วย ประกอบกับ Search Algorithm และตัวเลขเพื่อใช้ในการตัดสินใจ ดังนั้นขั้นแรกคือจะต้องแสดงปัญหาให้อยู่ในรูปของ Data Structure ให้ได้ก่อน

PRACTICE 1 จงแปลงปัญหาให้เป็น Data Structure ต่างๆ ที่เหมาะสมกับการแก้ไขปัญหาต่อไปนี้

1. ปัญหา Tic-Tac-Toe หรือ O X
2. ปัญหาเขาวงกต ขนาด 4x4 ช่อง

โดยส่วนมาก เรามักจะกำหนด State ของแต่ละปัญหาให้อยู่ใน Domain ของตัวแปรตามจำนวนมิติของปัญหา เช่น หากเป็นเข้าวงกตขนาด NxN แล้ว เราจะกำหนดให้ Domain เป็นพิกัด x และ y ซึ่ง x และ y จะมีค่าอยู่ในเซตของ {1, ..., N} หรือเขียนเป็นประโยคสัญลักษณ์ได้ว่า (x, y) **ϵ** {Z | 1 <= Z <= N} สรุปง่ายๆ คือ เรามักจะเขียน State ของปัญหาให้อยู่ในรูปของ**ระบบพิกัด**

PRACTICE 2 จงเขียน Domain Definition ของปัญหาต่อไปนี้

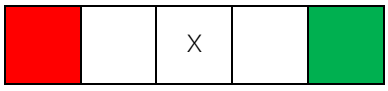
1. ปัญหา Tic-Tac-Toe หรือ O X
2. ปัญหาเขาวงกต ขนาด 4x4x3 ช่อง
3. ปัญหาการรินน้ำให้ได้ 4 ลิตร เมื่อมีถังน้ำขนาด 3 ลิตร และ 5 ลิตร

เมื่อเราสามารถสร้าง Domain Definition ได้แล้ว เราจำเป็นจะต้องศึกษากลไกการทำงานของปัญหาว่า สถานะเริ่มต้นของปัญหา (Initial State) และสถานะเป้าหมายของปัญหา (Goal State) คืออะไร

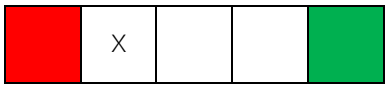
PRACTICE 3 จงเขียน Initial State และ Goal State ของปัญหาต่อไปนี้

1. ปัญหาการรินน้ำให้ได้ 4 ลิตร เมื่อมีถังน้ำขนาด 3 ลิตร และ 5 ลิตร
2. ปัญหา Tic-Tac-Toe หรือ O X

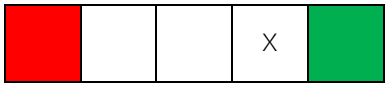
หลังจากนั้น เมื่อได้ State Space และ Initial and Goal State แล้ว สิ่งสุดท้ายคือการกำหนดกฎของปัญหาว่ามีกฎเป็นอย่างไรบ้าง เช่น ในปัญหาเขาวงกตจะสามารถเดินไปทางซ้ายหรือเดินไปทางขวาได้ แต่เราจะไม่สามารถเดินได้เมื่อเกิน Domain ของปัญหาไปแล้ว



Initial State



X = X – 1; (Move Left)

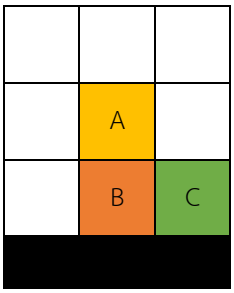


X = X + 1; (Move Right)

PRACTICE 4 จงเขียน Operator ของปัญหาต่อไปนี้ ให้ครอบคลุมกฎของปัญหา

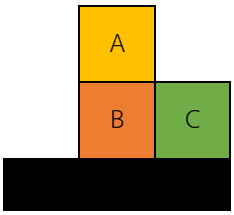
1. ปัญหาการรินน้ำให้ได้ 4 ลิตร เมื่อมีถังน้ำขนาด 3 ลิตร และ 5 ลิตร
2. ปัญหา Tic-Tac-Toe หรือ O X

แต่ในการเขียน Operator นั้น เราไม่จำเป็นที่จะต้องอ้างอิงจาก Logic ทั่วไปตามหลักคณิตศาสตร์อย่างเดียว โดยอาจจะเขียนให้อยู่ในลักษณะของตรรกะที่เข้าใจได้ง่าย โดยการมองวัตถุข้างใน State แต่ละ State ให้เป็น Object แล้วเลือกหยิบใช้ Object นั้นมาเข้าคู่กับ Operator (เป็น Verb ของ Object) เช่น ปัญหากล่อง 3 กล่องวางเรียงกันบนโต๊ะ เราจะให้กล่อง A B C อยู่เรียงกัน หากเรามองในแง่ของ Logic ในทางคณิตศาสตร์ทั่วไป อาจจะมองให้เป็น Array 2 มิติที่กล่องแต่ละกล่องเรียงกันอยู่ในนั้น



Normal approached state representation for Box Puzzle

ดังนั้นหากเรามองในแง่ว่ากล่องแต่ละใบคือ Object แล้วคิด Operator ให้คล้ายกับมนุษย์ เช่น วางกล่อง X ไปบนกล่อง Y และวางกล่อง X ไว้บนโต๊ะ เป็นต้น เราจะเรียก Operator หรือ Logic แบบนี้ว่า Predicate Logic



Predicate approached state representation for Box Puzzle

PRACTICE 5 จงเขียน Initial State, Goal State รวมถึง Operator ให้ครอบคลุมปัญหาดังต่อไปนี้ รวมถึงหาจำนวน State ทั้งหมดที่เป็นไปได้ของปัญหาดังกล่าว

1. ปัญหา Tower of Hanoi
2. ปัญหา 15-Puzzle (ตัวเลข 15 ตัวบนช่องทั้งหมด 16 ช่อง)
3. ปัญหา The Farmer, Fox, Goose and Grain

โดยในการทำ State Space นั้น จะต้องมีองค์ประกอบทั้งหมดนี้เพื่อเตรียมนำไปให้ Algorithm ทำงาน ซึ่งสำหรับ Algorithm ที่นำมาใช้งานในรูปแบบ Artificial Intelligence นั้นคือ Heuristic Algorithm หรือจะเป็น Algorithm ที่สามารถหาคำตอบได้ แต่ในบางครั้งคำตอบที่ได้อาจจะไม่ได้ดีที่สุด

ใน State Space นั้น เราสามารถเขียนรูปแบบของ Tree เพื่อแจกแจง State ต่างๆ เมื่อใช้งาน Operator ใดๆ ก็ตาม โดยจะแบ่ง State Space Search ออกเป็น 3 ประเภท คือ Blind Search, Heuristic Search และ Adversarial Search

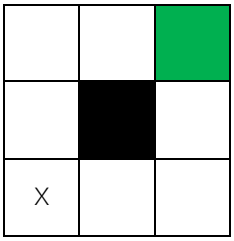
Blind Search

เป็นการค้นหาแบบที่เราจะไม่เห็น State ทั้งหมดที่เป็นไปได้ แบ่งเป็น 2 วิธี คือ Exhaustive Search และ Partial Search

Exhaustive Search คือการค้นหาแบบทั้งหมดทุกกรณีที่เป็นไปได้ ฉะนั้นยังไงก็ต้องเจอกับ Goal State ของปัญหาหาก Operator หรือองค์ประกอบต่างๆ ไม่ผิดพลาด

PRACTICE 6 ให้วาดแสดงทุกขั้นตอนของ Exhaustive Search เพื่อค้นหา Goal State ทั้งหมดที่เป็นไปได้ของปัญหา

1. กำหนดให้ Domain ของปัญหาเป็น X **ϵ** {1, 2, 3, 4} โดยจะมี Operator เป็น X = X + 1 กำหนดให้ Initial State คือ X = 1 แล้ว Goal State คือ X = 4
2. กำหนดปัญหาเข้าวงกตขนาด 3x3 โดยมี Operator เป็นขึ้น ลง ซ้าย ขวา ตามลำดับ และห้ามไปที่ช่องที่ไปไม่ได้ หรือช่องที่เคยผ่านมาแล้ว กำหนดให้จุด (1, 3) คือ Initial State และ (3, 1) คือ Goal State



อีกแบบหนึ่งคือ Partial Search โดยจะใช้วิธีค้นหาเพียงแค่บางส่วนเท่านั้น (แต่ก็มีโอกาสจะครบทุกกรณีใน Worst Case Scenario) ซึ่งเหมาะกับการหาคำตอบเพียง 1 กรณีเท่านั้น โดยจะแบ่งออกเป็น 2 วิธีย่อยๆ คือ Breadth-First Search (BFS) และ Depth-First Search (DFS)

- Breadth First Search หรือการค้นหาแบบแนวตื้น คือจะค่อยๆ ไล่ State ใน Level นั้นๆ ของ Tree ให้หมดก่อนแล้วจึงไปยัง Level ถัดไป ซึ่งการใช้ BFS นั้น จะทำให้ได้ Optimal Solution ของปัญหา (Level น้อยที่สุด)

PRACTICE 7 ให้วาดแสดงทุกขั้นตอนของ Partial Search เพื่อค้นหา Goal State ที่ดีที่สุด (Optimal) เพียง 1 State

1. ปัญหา 8-Puzzle ที่กำหนดให้มี Operator เป็นการเลื่อนช่องว่างไปทาง ซ้าย ขวา บน ล่าง ตามลำดับ โดยกำหนดให้ Initial State และ Goal State มีรูปแบบดังต่อไปนี้

2	3	
1	4	6
7	5	8

Initial State

2		4
1	6	3
7	5	8

Goal State

2. ปัญหา 15-Puzzle ที่กำหนดให้มี Operator เป็นการเลื่อนช่องว่างไปทาง ขวา บน ซ้าย ล่าง ตามลำดับ โดยกำหนดให้ Initial State และ Goal State มีรูปแบบดังต่อไปนี้

1	7	2	4
5	6	3	8
	9	10	12
13	14	11	15

Initial State

1	2		4
6	7	3	8
5	9	10	12
13	14	11	15

Goal State

- Depth First Search หรือการค้นหาแบบแนวลึก คือจะค่อยๆ ไล่ State ไปเป็นกิ่งทางลึกก่อนจนกว่าจะไม่สามารถไปต่อได้ แล้วจึง Recursive ขึ้นมาที่ Parent Node แล้วทำต่อเรื่อยๆ ซึ่งในการใช้ DFS นั้น จะทำให้ได้ Goal State ที่เร็วที่สุด แต่จะไม่รับประกันความเป็น Optimal Solution เหมือนใน BFS

PRACTICE 8 ให้วาดแสดงทุกขั้นตอนของ Partial Search เพื่อค้นหา Goal State ที่เพียง 1 State โดยใช้วิธี DFS

1. ปัญหา 8-Puzzle ที่กำหนดให้มี Operator เป็นการเลื่อนช่องว่างไปทาง ซ้าย ขวา บน ล่าง ตามลำดับ โดยกำหนดให้ Initial State และ Goal State มีรูปแบบดังต่อไปนี้

2	3	
1	4	6
7	5	8

Initial State

2		4
1	6	3
7	5	8

Goal State

2. ปัญหา 15-Puzzle ที่กำหนดให้มี Operator เป็นการเลื่อนช่องว่างไปทาง ขวา บน ซ้าย ล่าง ตามลำดับ โดยกำหนดให้ Initial State และ Goal State มีรูปแบบดังต่อไปนี้

1	7	2	4
5	6	3	8
	9	10	12
13	14	11	15

Initial State

1	2		4
6	7	3	8
5	9	10	12
13	14	11	15

Goal State

Heuristic Search

จากการใช้ Blind Search ซึ่งเป็นการค้นหาในรูปแบบที่มองไม่เห็น State ถัดไป หรือไม่มีหลักการอะไรที่ตายตัว นอกจาก Initial State ไปถึง Goal State แต่ใน Heuristic Search นั้น จะเพิ่ม Knowledge Base (องค์ความรู้) เข้าไปหนึ่ง

ส่วนของ Algorithm คือ Heuristic Function เพื่อทำให้เห็นเส้นทางที่ดีเพียงพอในแต่ละครั้งสำหรับการ Traverse ไปยัง Goal State ซึ่ง Algorithm ใน Heuristic Search แบ่งเป็น 3 แบบ คือ Hill Climbing, Best First Search และ A* Algorithm

Hill Climbing Algorithm

เป็น Algorithm ที่ใช้ Heuristic Function ในการช่วยตัดสินใจ โดยจะทำการแตกกิ่งแล้วเลือกกิ่งที่มีค่า Heuristic ที่เหมาะสมที่สุดเป็นเส้นทางต่อไป ซึ่งแบ่งออกเป็น 3 รูปแบบ คือ Simple Hill Climbing, Steepest-Ascent Hill Climbing และ Beam Algorithm

- Simple Hill Climbing จะแตกกิ่งทีละกิ่งแล้วใช้การคำนวณ Heuristic Function เทียบกับ Node ที่แตกออกมา หาก Node ที่แตกออกมามีค่าที่ดีกว่า จะไปยัง Node นั้น แต่หากไม่ จะทำการแตกกิ่งต่อไป ซึ่งใน Simple Hill Climbing จะมีปัญหาในเรื่องของ Local Maximum Problem และ Plateau Problem

PRACTICE 9 ให้วาดแสดงทุกขั้นตอนของ Simple Hill Climbing เพื่อค้นหา Goal State ที่เพียง 1 State ที่กำหนดให้ Heuristic Function $h(x) = SUM(d_x(c_i, g_i) + d_y(c_i, g_i))$; d เป็นความระยะห่างจาก State ปัจจุบัน กับ Goal State ในแกนต่างๆ

1. ปัญหา 8-Puzzle ที่กำหนดให้มี Operator เป็นการเลื่อนช่องว่างไปทาง ซ้าย ขวา บน ล่าง ตามลำดับ โดยกำหนดให้ Initial State และ Goal State มีรูปแบบดังต่อไปนี้

2	3	
1	4	6
7	5	8

Initial State

2		4
1	6	3
7	5	8

Goal State

2. ปัญหา 15-Puzzle ที่กำหนดให้มี Operator เป็นการเลื่อนช่องว่างไปทาง ขวา บน ซ้าย ล่าง ตามลำดับ โดยกำหนดให้ Initial State และ Goal State มีรูปแบบดังต่อไปนี้

1	7	2	4
5	6	3	8
	9	10	12
13	14	11	15

Initial State

1	2		4
6	7	3	8
5	9	10	12
13	14	11	15

Goal State

- Steepest-Ascent Hill Climbing จะนำมาแก้ไขปัญหา Local Maximum Problem (ข้ามกิ่งที่มีโอกาสเป็น Optimal Solution ไป) ซึ่งจะแตกกิ่งครั้งละ Level แล้วใช้การคำนวณ Heuristic Function เทียบกับ Node ที่แตกออกมาทั้งหมดใน Level นั้นๆ แทน เปรียบเสมือนการนำ Simple Hill Climbing Algorithm + Breadth-First Search (BFS)

PRACTICE 10 ให้วาดแสดงทุกขั้นตอนของ Steepest-Ascent Hill Climbing เพื่อค้นหา Goal State ที่เพียง 1 State ที่กำหนดให้ Heuristic Function h(x) = SUM($d_x(c_i, g_i) + d_y(c_i, g_i)$) ; d เป็นความระยะห่างจาก State ปัจจุบัน กับ Goal State ในแกนต่างๆ

1. ปัญหา 8-Puzzle ที่กำหนดให้มี Operator เป็นการเลื่อนช่องว่างไปทาง ซ้าย ขวา บน ล่าง ตามลำดับ โดยกำหนดให้ Initial State และ Goal State มีรูปแบบดังต่อไปนี้

2	3	
1	4	6
7	5	8

Initial State

2		4
1	6	3
7	5	8

Goal State

2. ปัญหา 15-Puzzle ที่กำหนดให้มี Operator เป็นการเลื่อนช่องว่างไปทาง ขวา บน ซ้าย ล่าง ตามลำดับ โดยกำหนดให้ Initial State และ Goal State มีรูปแบบดังต่อไปนี้

1	7	2	4
5	6	3	8
	9	10	12
13	14	11	15

Initial State

1	2		4
6	7	3	8
5	9	10	12
13	14	11	15

Goal State

- Beam Algorithm จะมาช่วยแก้ไขปัญหา Local Maximum Solution ของ Hill Climbing Algorithm ไป โดยการให้เลือกเส้นทางใน Steepest-Ascent Hill Climbing ได้มากกว่า 1 กิ่ง (วิธีที่เลือกทุกกิ่งจะพัฒนาเป็น Best-First Search Algorithm)

PRACTICE 11 ให้วาดแสดงทุกขั้นตอนของ Beam Algorithm ที่สามารถเลือกได้ 2 กิ่ง เพื่อค้นหา Goal State ที่เพียง 1 State ที่กำหนดให้ Heuristic Function h(x) = SUM($d_x(c_i, g_i) + d_y(c_i, g_i)$) ; d เป็นความระยะห่างจาก State ปัจจุบัน กับ Goal State ในแกนต่างๆ

1. ปัญหา 8-Puzzle ที่กำหนดให้มี Operator เป็นการเลื่อนช่องว่างไปทาง ซ้าย ขวา บน ล่าง ตามลำดับ โดยกำหนดให้ Initial State และ Goal State มีรูปแบบดังต่อไปนี้

2	3	
1	4	6
7	5	8

Initial State

2		4
1	6	3
7	5	8

Goal State

2. ปัญหา 15-Puzzle ที่กำหนดให้มี Operator เป็นการเลื่อนช่องว่างไปทาง ขวา บน ซ้าย ล่าง ตามลำดับ โดยกำหนดให้ Initial State และ Goal State มีรูปแบบดังต่อไปนี้

1	7	2	4
5	6	3	8
	9	10	12
13	14	11	15

Initial State

1	2		4
6	7	3	8
5	9	10	12
13	14	11	15

Goal State

Best-First Search

อย่างที่ได้กล่าวไปว่าหากใช้ Steepest-Ascent Hill Climbing Algorithm แบบไม่ตัดกิ่งออกเลย ซึ่งเวลาเทียบ Heuristic Function ก็จะเทียบรวมทั้งกราฟ จะทำให้ได้ Optimal Solution เสมอ

PRACTICE 12 ให้วาดแสดงทุกขั้นตอนของ Best-First Search ที่สามารถเลือกได้ 2 กิ่ง เพื่อค้นหา Goal State ที่ดีที่สุดเพียง 1 State ที่กำหนดให้ Heuristic Function $h(x) = SUM(d_x(c_i, g_i) + d_y(c_i, g_i))$; d เป็นความระยะห่างจาก State ปัจจุบันกับ Goal State ในแกนต่างๆ

1. ปัญหา 8-Puzzle ที่กำหนดให้มี Operator เป็นการเลื่อนช่องว่างไปทาง ซ้าย ขวา บน ล่าง ตามลำดับ โดยกำหนดให้ Initial State และ Goal State มีรูปแบบดังต่อไปนี้

2	3	
1	4	6
7	5	8

Initial State

2		4
1	6	3
7	5	8

Goal State

2. ปัญหา 15-Puzzle ที่กำหนดให้มี Operator เป็นการเลื่อนช่องว่างไปทาง ขวา บน ซ้าย ล่าง ตามลำดับ โดยกำหนดให้ Initial State และ Goal State มีรูปแบบดังต่อไปนี้

1	7	2	4
5	6	3	8
	9	10	12
13	14	11	15

Initial State

1	2		4
6	7	3	8
5	9	10	12
13	14	11	15

Goal State

A* Search

เป็น Algorithm Best-First Search ที่มีการเปลี่ยนไปใช้ Function f'(x) = g(s) + h'(s) ซึ่ง g(s) คือ Cost ระหว่าง Initial State ถึง Current State และ h'(s) คือ Cost ระหว่าง Current State ถึง Goal State ดังนั้น f'(s) จะเป็น Cost ระหว่าง Initial State ถึง Goal State (คำนวณจากผลบวก g(s) และ h'(s))

PRACTICE 13 ให้วาดแสดงทุกขั้นตอนของ A* Search เพื่อค้นหา Goal State ที่ดีที่สุดเพียง 1 State ที่กำหนดให้ Heuristic Function h'(s) = SUM($d_x(c_i, g_i) + d_y(c_i, g_i)$) ; d เป็นความระยะห่างจาก Current State กับ Goal State ในแกนต่างๆ และ g(s) = SUM($d_x(i_i, c_i) + d_y(i_i, c_i)$) ; d เป็นความระยะห่างจาก Initial State กับ Current State ในแกนต่างๆ

1. ปัญหา 8-Puzzle ที่กำหนดให้มี Operator เป็นการเลื่อนช่องว่างไปทาง ซ้าย ขวา บน ล่าง ตามลำดับ โดยกำหนดให้ Initial State และ Goal State มีรูปแบบดังต่อไปนี้

2	3	
1	4	6
7	5	8

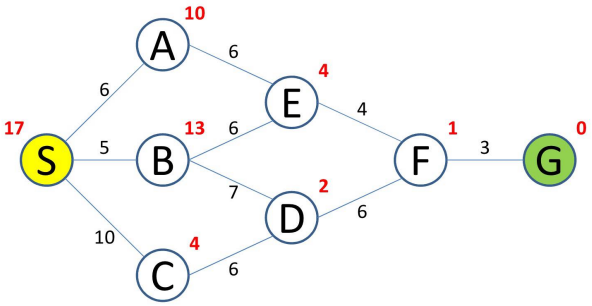
Initial State

2		4
1	6	3
7	5	8

Goal State

PRACTICE 14 ให้วาดแสดงทุกขั้นตอนของ A* Search เพื่อค้นหา Goal State ที่ดีที่สุดเพียง 1 State ที่กำหนดให้ Heuristic Function h'(s) = SUM($d(c_i, g_i)$) ; d เป็นความระยะห่างจาก Current State กับ Goal State และ g(s) = SUM($d(i_i, c_i)$) ; d เป็นความระยะห่างจาก Initial State กับ Current State

1. ปัญหาการหา Shortest Path ต่อไปนี้ จาก Node S ไปยัง Node G



Adversarial Search

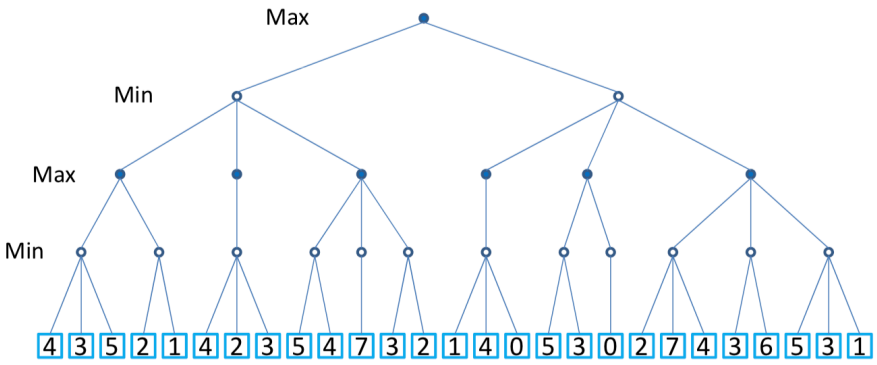
ใน Adversarial Search เหมาะกับ 2-player game หรือเกมที่มี 2 ผู้เล่น ซึ่งจะมีผลแพ้-ชนะเกิดขึ้น ทำให้เวลาเล่นจะเกิดการสลับเปลี่ยนกัน ดังนั้นการมอง State เพียงแค่ Level เดียวจะไม่เพียงพออีกต่อไป ซึ่งสำหรับวิธีการก็ยังคงใช้วิธี Heuristic แบบเดิม เพียงแต่จะใช้วิธีที่สามารถมองลงไปได้มากกว่า 1 Level ของ State Space Tree โดยจะมีวิธี Minimax Algorithm และเทคนิคช่วยตัดกิ่งคือ Alpha-Beta Pruning

Minimax Algorithm

จากเกมที่มี 2 ผู้เล่น จะทำให้เกิดการสลับ Turn การเล่นกันไปเรื่อยๆ จนทำให้ต้องมีการมอง Level ของ State Space Tree อย่างน้อย 2 Levels ดังนั้น Minimax Algorithm จึงจะใช้วิธีการมองแบบสลับ Level กล่าวคือ ใน Level แรกจะมอง Cost ที่มากที่สุด ส่วนใน Level ถัดมาก็จะมอง Cost ที่ต่ำที่สุด ซึ่ง Cost ก็เปรียบเป็นโอกาสชนะของผู้เล่นในรอบนั้นๆ มาจากสูตร Cost = (1^{st} Player Win Rate – 2^{nd} Player Win Rate) เรียกว่า Evaluation Function

ใน Level บนสุดของ Graph จะเป็นค่า Max ของ Child และ Level ถัดไปจะเป็นค่า Min ของ Child แล้วจะสลับแบบนี้ไปเรื่อยๆ จนกว่าจะสุด โดยไล่การค้นหาแบบ Depth-First Search (DFS) ทำให้ในตอนสุดท้ายเมื่อ Trace แล้ว จะได้เส้นทางที่ดีที่สุดของการเล่นเกม

PRACTICE 15 ให้วาดแสดงการ Trace ทุกขั้นตอน โดยใช้ Minimax Algorithm พร้อมทั้งบอกเส้นทางที่ดีที่สุด

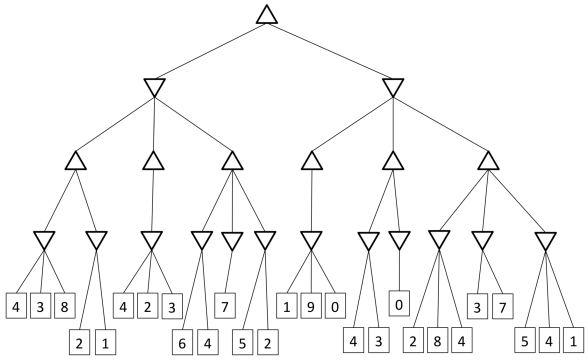


Alpha-beta Pruning (หนังสือแต่ละเล่มจะเขียนวิธีที่แตกต่างกัน)

เนื่องจากใน Minimax Algorithm จะต้องทำทุกขั้นตอนโดยละเอียด ทำให้มีค่าเหมือนการทำ Exhaustive Search จึงมีเทคนิคที่เข้ามาช่วยตัดกิ่งที่ไม่จำเป็นออก คือ Alpha-beta Pruning ซึ่งจะใช้วิธีคำนวณค่า Alpha และ Beta โดยค่า Alpha จะคำนวณที่ Max Level ส่วนค่า Beta จะคำนวณที่ Min Level ซึ่งจะตรวจสอบหาก Level ด้านบนมีการคำนวณค่า Alpha หรือ Beta เอาไว้แล้ว จะเปรียบเทียบกับค่าตนเองด้วยเงื่อนไขดังนี้

- MAX Level: Alpha > Parent Beta จะตัดกิ่งที่เหลือของ Node ดังกล่าวออก (Cutoff)
- MIN Level: Beta < Parent Alpha จะตัดกิ่งที่เหลือของ Node ดังกล่าวออก (Cutoff)

PRACTICE 16 ให้แสดงการทำ Cutoff ด้วย Alpha-beta Pruning ของ Graph ต่อไปนี้



Mitsuku

- เป็น Chatbot
- ใช้เทคโนโลยี AIML (AI + XML)
- คนทำชื่อ Steve Worswick
- เริ่มต้นเมื่อปี 2000
- Deploy ลง Pandorabots
- ปี 2017 มี Avatar 3D
- ใช้ Knowledge Based จาก AIML
- สามารถ Deduction หรือให้เหตุผลได้ รูปแบบเป็น CBR (Case-based Reasoning)
- ได้รางวัล Loebner Prize 4 ปี คือ 2013, 2016, 2017, 2018
- สร้างเพื่อมาพิชิต Turing Test

AIBO

- หุ่นยนต์สุนัข (Dog-liked Robot)
- สร้างโดย SONY ปี 1998 เป็น Prototype
- แบ่งเป็น 4 ยุค
    - 1st Gen (1999) : ERS-110, ERS-111
    - 2nd Gen: ERS-210, ERS-300 (Latte and Macaron), ERS-220, ERS-210A/220A
    - 3rd Gen: ERS-7, ERS-7M2, ERS-7M3
    - 4th Gen (2018) : ERS-1000
- มีเทคโนโลยี WAN, NLP (Natural Language Processing) เช่น Speech Recognition, Deep Learning (เช่นใช้ kNN; k-nearest neighbour หรือ SVM; support vector machine ในการแยกสี)
- เฝ้าบ้านไม่ได้ เหมาะกับการเอามาเลี้ยงดูเล่นๆ
- R-CODE เป็น Scripting Language ของ AIBO
- AIBO Vision System ใช้ Algorithm SIFT (Computer Vision Algorithm) เป็น Feature Detection
- มี Application เชื่อมต่อได้
- CMU เอา AIBO ไปใช้เรื่อง Visual Tracking

ViV

- Voice Controlled Assistance สร้างโดย Dag Kittlaus, Adam Cheyer และ Chris Brigham
- เป็น Speech Recognition Service
- ViV + Samsung
- มี NLP ที่ดีกว่ารุ่นอื่นๆ เช่น Siri
- ใช้ Dynamic Program Generation หรือเขียนโปรแกรมได้ด้วยตนเอง โดยจะทำ Word Segmentation แล้วนำมาหาความสัมพันธ์ (ซึ่งรุ่นอื่นๆ ไม่มี) จึงค่อยโต้ตอบ

AlphaGo

- Google DeepMind ปี 2014 โดย David Sliver
- แบ่งเป็น 5 Version คือ Fan (Fan Hul 5:0 / 2015), Lee (Lee Sedul 4:1 / 2016), Master (2016), Zero (2017) และ AlphaZero (2017)
- Purpose is to think/deduce like Human
- Neural Nets (Layer / Loss Function / Backpropagation) / Monte Carlo Tree Search / Reinforcement Learning (+Reward)

Google Brain

- Google in 2011 โดย Jeff Dean, Greg Corrado, Andrew Ng
- เริ่มจาก Project Google X
- 2012 – Image Classification of Cats
- วัตถุประสงค์: To make machine intelligent and improve people's lives
- 2016: Artificial-Intelligence-Devised Encryption System ใช้ Generative Adversarial Network (GAN) โดยมี Alice และ Bob คอยแลกเปลี่ยนข้อความ โดยที่ Bob จะเป็นคน Decrypt โดยจะมี Eve มาขัดขวาง ซึ่งมีไว้ใช้เพื่อสร้าง Encryption System ของ Alice ให้ดีขึ้น (มี Loss Function)

REFERENCES

---

1. https://dtai.cs.kuleuven.be/education/ai/Exercises/Session4/Solutions/solution.pdf
2. https://www.cc.gatech.edu/~riedl/classes/2014/cs3600/homeworks/minimax-soln.pdf