Name : GULLEMOT Kentin

Lab group : DIA 3

Homework 1 : AI Algorithms

Exercise 1 : Reading

Question : *In the paper, Turing discusses several potential objections to his test for intelligence. Which of these objections still hold some significance today? Are his refutations valid?*

Answer :

The following objections are those which still hold some significance today :

1) Theological Objection : This argument suggests that thinking is a function of the human soul, which machines cannot possess. While less commonly debated in mainstream discussions today, it still underpins some philosophical or religious perspectives on artificial intelligence.

2) The Argument from Consciousness : This objection claims that machines cannot truly be said to "think" unless they experience emotions or self-awareness. Today, this is still a significant challenge in discussions about AI, particularly around the idea of "strong AI" (machines having true consciousness). Turing countered this by suggesting that behavior, not subjective experience, should be the benchmark.

3) The Mathematical Objection : Gödel's incompleteness theorem, which suggests that there are truths machines cannot prove, is still debated in the context of AI limits. However, Turing's counterargument that this limitation applies to humans as well remains a strong rebuttal, especially in light of AI's achievements in problem-solving and computation.

Question : *Can you think of new objections that have arisen from developments since the paper was written?*

Answer :

Since Turing's time, new challenges have arisen, particularly related to ethics, bias, and control. Concerns about AI safety, decision-making in high-stakes areas (such as warfare or healthcare), and the ethical implications of superintelligent AI were not anticipated by Turing but have become central in modern AI discussions.

We can specially see this in our school with some new type of course named "Ethique de l'ingénieur".

Question : *Additionally, Turing predicts that by the year 2000, a computer would have a 30% chance of passing a five-minute Turing Test with an unskilled interrogator. What do you think a computer's chances would be today?*

Answer :

Today's AI would likely have a much higher chance, especially given advances in natural language processing (NLP) with the boom of LLMs like ChatGPT, Gemini, Mistral, etc. However, the complexity of truly passing a robust Turing Test still remains debated, as AI can mimic human conversation convincingly but lacks genuine understanding or consciousness.

Exercise 2 : Search Problem

A) Formulate this puzzle as a search problem. What are the states, actions, initial state, and goal condition?

Initial state :

6	9	8
7	1	3
2	5	4

A state is defined by the position of the numbers on the grid. **Each states is a matric 3x3 with all the numbers in a specific configuration.**

The actions correspond to moving the number 9 (the “sliding number”) in one of the four directions : **up, down, left, or right** *(within the limits of what the grid allows us to do)*.

The goal is to rearrange the numbers so that the sum of each row, each column, and both diagonals equals 15.

B) Determine whether the state space is represented as a graph or a tree.

The state space is best represented as a **graph** because there are multiple possible sequences of moves that could lead to the same state. Thus, different paths in the search may lead to the same configuration, which would not be the case in a tree.

C) How large is the state space?

The state space of the Sliding Magic Square puzzle consists of 181,440 unique states.

Although there are 9! = 362,880 possible permutations of the nine numbers in the 3x3 grid, not all permutations are reachable due to parity constraints inherent in sliding puzzles. Legal moves where the number 9 swaps places with an adjacent number do not change the parity of the configuration. This means that from any given initial state, you can only reach configurations that have the same parity. As a result, only half of all possible permutations are accessible through legal moves.

Therefore, the state space includes 362.080 / 2 = 181,440 possible states, representing all configurations with the same parity as the initial state.

D) What is the maximum branching factor for this problem? Provide justification.

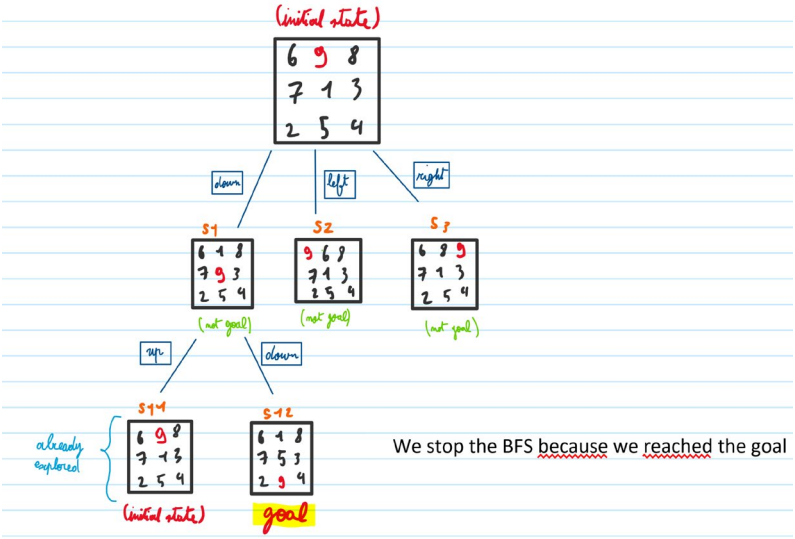
The maximum branching factor corresponds to the maximum number of possible moves for the number 9. The branching factor depends on the position of the sliding number 9, because he can't go out of the gird so the different actions are limited by the size of the grid.

In this case, for a matrix 3x3, the only way that's 9 is capable to move in in any direction (up, down, right, left) is to be in the **center of the grid**. **In this case the maximum branching factor is 4.**

In other positions, the branching factor will be lower (3 for an edge, 2 for a corner).

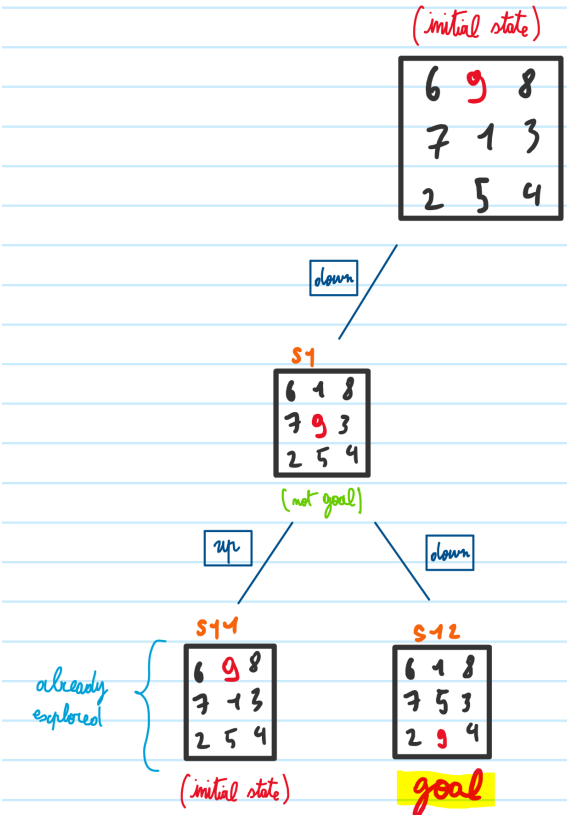
E) Draw a portion of the search graph resulting from Breadth-First Search (BFS) algorithm. Label the nodes in the order in which they are expanded.

We must pay close attention to the possible movements to make and those that we cannot make depending on the position of the number 9 (example: in the first line we cannot make an "up")



Expande node	Node list
	Initial State
Initial State not goal	{S1,S2,S3}
S1 not goal	{S2,S3,S11,S12,S13,S14}
S2 not goal	{S3, S11,S12,S13,S14,...}
S3 not goal	{S11,S12,S13,S14,...}
S11 already explored	{S12,S13,S14,...}
S12 goal	{S13,S14,...}

F) Draw a portion of the graph search generated by the Depth-First Search (DFS) algorithm and label the states in the order they are expanded.



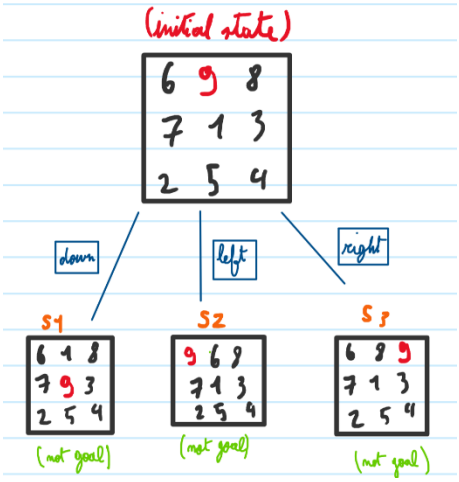
We stop the DFS because we reached the goal

Expande node	Node list
	Initial State
Initial State not goal	{S1,S2,S3}
S1 not goal	{S11,S12,S13,S14,S2,S3}
S11 already explored	{S12,S13,S14,S2,S3}
S12 goal	{S121,S122,S123,S13,S14,S2,S3}

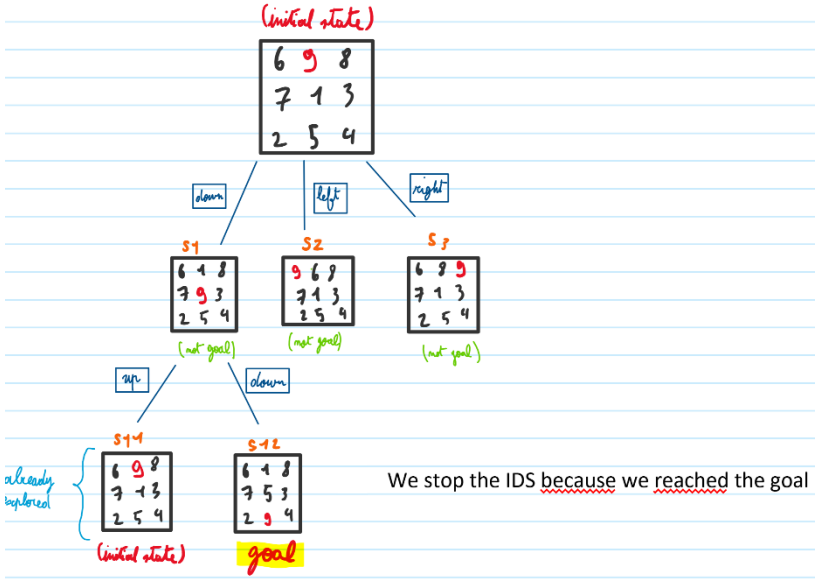
G) Draw a portion of the graph search generated by the Iterative Deepening Search (IDS) algorithm and label the order in which each state is expanded.

The principle of IDS is to do DFS with depth limit starting at l = 1 up to l = d where d is the iteration where we obtained the solution.

l = 1 :



l = 2 :



	Expande node	Node list
		Initial State
DFS at l = 1	Initial State not goal	{S1,S2,S3}
	S1 not goal	{S2,S3}
	S2 not goal	{S3}
	S3 not goal	{ }
DFS at l = 2	Initial State not goal	{S1,S2,S3}
	S1 not goal	{S11,S12,S13,S14,S2,S3}
	S11 already explored	{S12,S13,S14,S2,S3}
	S12 goal	{S121,S122,S123,S13,S14,S2,S3}

H) What are the advantages and disadvantages of Breadth-First Search and Iterative Deepening Search for this problem? Would Depth-First Search with no limit be a better or worse approach? Why?

BFS:

- **Advantages:**
  - **Optimality**: Since each move of the sliding number (9) has the same cost, BFS guarantees finding the solution with the **fewest moves**. This is ideal for puzzles like this where we seek the shortest sequence of moves.
  - **Completeness**: BFS will explore all possible states systematically, so it is guaranteed to find a solution if one exists.
- **Disadvantages:**
  - **High memory usage**: BFS requires a lot of memory to store all the states at each level. Given that the state space can be large (up to 40,320 possible configurations), the memory required grows quickly.
  - **Execution time**: If the solution is far from the initial state (requiring many moves), BFS can take longer because it must explore all shallow levels before moving deeper.

IDS:

- **Advantages:**
  - **Low memory usage**: Unlike BFS, IDS only needs to store the current path at each depth level, making it much more memory-efficient, especially when the number of states grows exponentially.
  - **Optimality and completeness**: Like BFS, IDS is **complete** and **optimal** (in terms of the fewest moves), making it a reliable choice if the solution's depth is unknown.
- **Disadvantages:**
  - **Redundancy**: IDS re-explores the same states multiple times at each new depth limit. In this puzzle, if the solution is deep, it will explore the initial configurations repeatedly.
  - **Slightly longer execution**: Due to this redundancy, IDS can take longer than BFS, though its lower memory usage often compensates for the time overhead.

DFS without a limit could potentially lead to a worse approach as it might go too deep unnecessarily. DFS could get stuck in long, unpromising paths and doesn't guarantee finding the shortest solution, making it less ideal for this puzzle compared to BFS or IDS. In this puzzle, it's possible to revisit a previous state by following certain move sequences. For instance, moving the "9" to the left and then back to the right can return you to the initial state. **Without tracking visited states, DFS could end up repeating the same sequence of moves over and over, creating an infinite loop.**

I) Propose a non-trivial admissible heuristic for solving this problem

A good heuristic for this puzzle could be the number of rows, columns, and diagonals that do not yet sum to 15. This gives a rough estimate of how far the puzzle is from completion.

This heuristic will never overestimate the actual number of moves required to solve the puzzle.

Exercise 3 : Programming the "8-Puzzle Problem"

All the answers are on the .ipynb file