German University in Cairo
Department of Computer Science
Assoc. Prof. Haythem O. Ismail
Eng. Nourhan Ehab
Eng. Ammar Yasser

Introduction to Artificial Intelligence, Winter Term 2020
Project 1: Mission Impossible: Cold War [1]

Due: November 22^{nd}

1. Project Description

The year is 1969 and the cold war is at its peak. Tensions between the United States (USA) and the Soviet Union (USSR) reached unprecedented levels. In fear of ruining their economies by starting the war, each country thinks it should win the war without actually going to war. To achieve that, both the USA and the USSR are rapidly expanding their nuclear arsenals to strike fear in the enemy and force them to surrender. The USSR is secretly planning on annexing an island called *Nova* [2]. Nova is an inhabitable island that, at first glance, seems completely useless. However, the Soviet Committee for State Security (KGB) gathered new information that confirms that Nova possesses a huge supply of underground enriched uranium. Such amount of uranium, if militarized, will make the USSR's arsenal far exceed that of the USA putting an end to the war once and for all.

A double agent leaked the USSR's plan to the American Central Intelligence Agency (CIA). As the stakes could not be higher, the CIA turned to its most renowned agent and the leader of the Impossible Missions Force (IMF), Ethan Hunt. The IMF team are now tasked with their most dangerous mission yet. The mission involves them flying to Nova undiscovered, planting a device near the uranium supply that will manipulate the chemical properties of the uranium rendering it unmilitarizable [3], and then finally coming back to the US. This mission should be performed in top secrecy because if the USSR found any of the IMF agents on its ground, that would be a declaration of war.

The IMF team took a special stealthy submarine to Nova. Then, they flew in a helicopter to the uranium's location. Finally, they planted and activated the device. By now, the hard part of the mission was accomplished and all that remains is to return to the submarine to go back to the US. After activating the device, they went back to the helicopter and started flying back to the submarine. However, they did not know that the device that manipulated the uranium caused a malfunction in the helicopter as well which started going out of control and was about to crash. Every member of the IMF team jumped out of the helicopter with a parachute and they all landed in different places. Everybody, except Ethan, is badly injured and is about to die. It is now Ethan's task to bring all the injured IMF members to the submarine using a small truck that he found nearby the location he landed in.

Nova can be thought of as an $m \times n$ grid of cells where $5 \leq m, n \leq 15$. Initially, a grid cell is either free or contains one of the following: Ethan with the truck, an IMF member, or

[1] This project's theme is based, to an extent, on the Mission Impossible move franchise.
[2] Based on an actual island named Novaya Zemlya lying in the Arctic Ocean to the north of Russia.
[3] Yes you guessed it, "unmilitarizable" is not actually a word but you get what I mean.

the submarine. In this project, you will use *search* to help Ethan complete the mission by finding all the injured IMF members and bringing them back to the submarine. Ethan can move with the truck in all four directions, can carry an IMF member only if both of them are in the same cell with the truck, and can drop IMF members at the submarine only if he is in the same cell where the submarine lies. The truck can only accommodate up to c IMF members at the same time. Accordingly, Ethan might have to make multiple trips to the submarine to transport all the injured IMF members. The IMF members are bleeding and will continue to incur damage with every passing time step. A time step is the duration taken by Ethan to complete one action. With every time step, the damage of any IMF member increases by 2. Since Ethan knows how to stop the bleeding, an IMF member stops incurring damage once Ethan picks them up. If the damage of an IMF member reaches 100, they die.

Using search you should formulate a plan that Ethan can follow to complete the mission. An optimal plan is one where the deaths are at a minimum as a first condition. Given two plans with the same number of deaths, the more optimal plan is the one where the total damage incurred by the IMF members is minimal. The following search strategies will be implemented and each will be used to help Ethan:

a) Breadth-first search.

b) Depth-first search.

c) Iterative deepening search.

d) Uniform-cost search.

e) Greedy search with at least two heuristics.

f) A* search with at least two *admissible* heuristics.

Each of the aforementioned strategies should be tested and compared in terms of RAM usage, CPU utilization, and the number of search tree nodes expanded. **You must only use Java to implement this project.**

Your implementation should have two main functions `genGrid` and `solve`:

- `genGrid()` generates a random grid. The dimensions of the grid, the starting position of Ethan and the submarine, as well as the number and locations of the IMF members are to be randomly generated. You need to make sure that the dimensions of the generated grid is between 5×5 and 15×15 and the number of generated IMF members is between 5 and 10. For every IMF member, a random starting health between 1 and 99 should also be generated. The number of members the truck can carry "c" should be randomly generated as well.
- `solve(String grid, String strategy, boolean visualize)` uses search to try to formulate a winning plan:
  - *grid* is a string representing the grid to perform the search on. This string should be in the following format:
    `m,n; ex,ey; sx,sy;`
    $\texttt{x}_1$,$\texttt{y}_1$, ...,$\texttt{x}_k$,$\texttt{y}_k$;
    $\texttt{h}_1$,...,$\texttt{h}_k$;
    `c`
    where:
    * `m` and `n` represent the width and height of the grid respectively.

* `ex` and `ey` represent the x and y starting positions of Ethan.
* `sx` and `sy` represent the x and y positions of the submarine.
* $\texttt{x}_i$,$\texttt{y}_i$ represent the x and y position of IMF member i where $1 \leq i \leq k$ and k is the total number of IMF members.
* $\texttt{h}_i$ represent the health of IMF member i where $1 \leq i \leq k$ and k is the total number of IMF members.
* `c` is the maximum number of members the truck can carry at a time.

Note that the string representing the grid does not contain any spaces or new lines. It just formatted this way above to make it more readable. All x and y positions are assuming 0-indexing.

– *strategy* is a symbol indicating the search strategy to be applied:
* `BF` for breadth-first search,
* `DF` for depth-first search,
* `ID` for iterative deepening search,
* `UC` for uniform cost search,
* `GR`i for greedy search, with $i \in \{1, 2\}$ distinguishing the two heuristics, and
* `AS`i for A* search, with $i \in \{1, 2\}$ distinguishing the two heuristics.

– *visualize* is a boolean parameter which, when set to `true`, results in your program's side-effecting a visual presentation of the grid as it undergoes the different steps of the discovered solution (if one was discovered).

The function returns a `String` of the following format: `plan;deaths;healths;nodes` where

– `plan` is a string representing the operators Ethan needs to follow separated by commas. The possible operator names are: `up, down, left, right, carry` and `drop`.
– `deaths` is a number representing the number of deaths in the found goal state.
– `heaths` is a string of the format $\texttt{h}_1$,...,$\texttt{h}_k$ where $\texttt{h}_i$ is the health of IMF member i in the found goal state.
– `nodes` is the number of nodes chosen for expansion during the search.

2. Sample Input/Output:

Example Input Grid: 5,5;1,2;4,0;0,3,2,1,3,0,3,2,3,4,4,3;20,30,90,80,70,60;3

			F (20)	
		E		
	F (30)			
F (90)		F (80)		F (70)
S			F (60)	

E represents Ethan, S represents the submarine and F (H) represents an IMF member with initial health H.

Example Output:
up,right,carry,down,down,down,right,carry,down,left,carry,left,left,left,drop,
up,carry,up,right,carry,down,right,carry,down,left,left,drop;
2;24,68,100,100,84,80;1165836

3. Groups: You may work in groups of *at most* four.

4. Deliverables

a) Source code.

- You should implement a data type for a search-tree node as presented in class.
- You should implement an abstract data type for a generic search problem, as presented in class.
- You should implement the generic search procedure presented in class, taking a problem and a search strategy as inputs. *You should make sure that your implementation of search does not allow repeated states and that all your search strategies terminate in under 1 minute.*
- You should implement a `MissionImpossible` subclass of the generic search problem. This class must contain `genGrid()` and `solve` as *static* methods.
- You should implement all of the search strategies indicated above together with the required heuristics. A trivial heuristic (e.g. $h(n) = 1$) is *not* acceptable. You should make sure that your heuristic function runs in maximum *polynomial* time in the size of the state representation.
- Your program should implement the specifications indicated above.
- Part of the grade will be on how readable your code is. Use explanatory comments whenever possible

b) Project Report, including the following.

- A discussion of your implementation of the search-tree node ADT.
- A discussion of your implementation of the search problem ADT.
- A discussion of your implementation of the MissionImpossible problem.
- A description of the main functions you implemented.
- A discussion of how you implemented the various search algorithms.
- A discussion of the heuristic functions you employed and, in the case of A*, an argument for their admissibility.
- At least two running examples from your implementation.
- A comparison of the performance of the implemented search strategies on your running examples in terms of completeness, optimality, RAM usage, CPU utilization, and the number of expanded nodes. You should comment on the differences in the RAM usage, CPU utilization, and the number of expanded nodes between the implemented search strategies.
- Proper citation of any sources you might have consulted in the course of completing the project. *Under no condition*, you may use on-line code as part of your implementation.
- If your program does not run, your report should include a discussion of what you think the problem is and any suggestions you might have for solving it.

5. Important Dates

Teams. Make sure you submit your team members' details by October 30^{th} at 23:59 using the following link `https://forms.gle/Hjr5N7Sz48CHw6mc6`. Only one team member should submit this for the whole team. After this deadline, we will be posting on the MET website a team ID for each submitted team. You will be using this team ID for submission.

Source code and Report. Online submission by November 22^{nd} at 23:59. The submission details will be announced after the team submission deadline.

Brainstorming session. In tutorials.