German University in Cairo
Department of Computer Science
Dr. Nourhan Ehab

Introduction to Artificial Intelligence, Winter Term 2025

Assignment 1: Minimax Battle

Due Tuesday, November 18th by 23:59

1. **Assignment Description** In this project, you will be implementing an adversarial search agent that plays a turn-based battle game. The game has two players, each player has an army of units each with a health and damage level assigned to each unit.

   - At each move, you the player can select which unit from his army to attack which unit from the other army.
   - An attack action results in the attacked unit losing an amount of health equal to the damage of the unit performing the attack.
   - Once a unit's health reaches zero, it is off the team and can no longer perform attacks or be attacked.
   - A unit's damage never changes.
   - A unit's health can never go lower than zero.
   - A better winning strategy is the one where the winner finishes with a larger total health remaining (That is the sum of all health values for all army units.)
   - The game ends when one of the players' total army health is zero. The winner is the other player.

   Given an initial description of the state, the agent should be able to search for a plan (if such a plan exists) to achieve its objective. In the following sections there are more details on the agent, actions and the implementation requirements.

2. **Implementation:** In this project, you will implement adversarial search to find a sequence of actions that solves the game with the best score. The search is to be implemented as described in the lectures. Search will be implemented in 2 ways:

   a) Using minimax only.

   b) Using minimax with alpha-beta pruning.

   You will be provided with a starter folder. You must adhere to the structure of this starter folder by not moving or deleting existing files but you can add as you wish. The starter folder hierarchy will be as follows:

```
aiproject1/
└─ src/
   ├─ battle/
   │  ├─ BattleSolver
   │  └─ Node
   └─ tests/
      ├─ battleTestsPublic
      └─ BattleGameChecker
```

- `BattleSolver` which implements the "Battle" problem.
- `Node`, which implements a generic search-tree node.
- `battleTestsPublic` includes public tests that should be run to validate your work. To run these tests, you will need junit 5.8.

You must not modify the existing attributes and function signatures in the provided starter or their access modifiers. These attributes will be used to test your work so you must use them. You may add any other attributes/functions that you need to complete the project. Please read the comments in the starter code carefully before starting.

Inside `BattleSolver` you will implement `solve` as the key function which will be the basis for testing :

- `solve(`*initialStateString*`,`*ab*`,` *visualize*`)` uses minimax search to find the optimal sequence of steps to win the game.
  - `initialStateString` a provided string that defines the parameters of the instance of the problem. It gives the initial health and damage values for each army and the starting player (The player we want to win). It is a string provided in the following format:

    $health_0A, damage_0A, health_1A, damage_1A, ...health_iA, damage_iA;$
    $health_0B, damage_0B, health_1B, damage_1B, ...health_jB, damage_jB;$
    $StartingPlayer;$

    Where
    * $health_iA$ is the health value of the ith unit in A's army.
    * $damage_iA$ is the damage that can be done of the ith unit in A's army.
    * *StartingPlayer* this is a single upper case character 'A' or 'B' that indicates which player starts first.

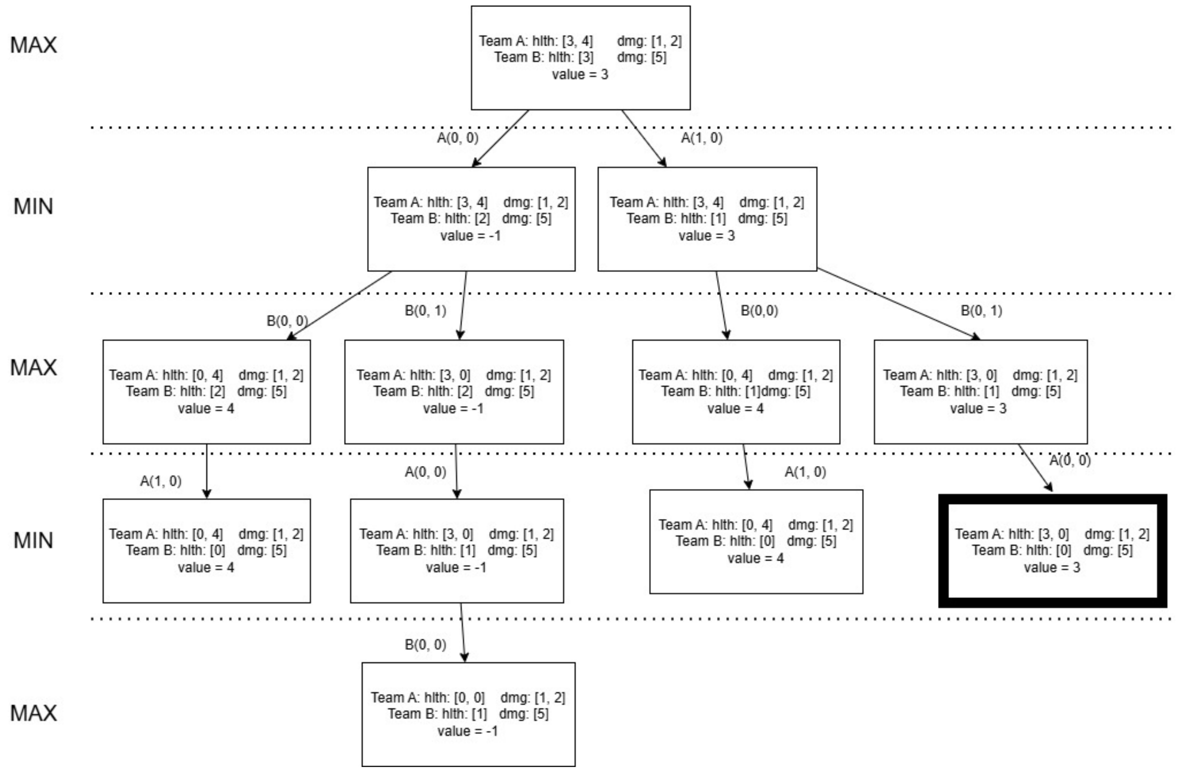
    Note that the string representing the initial state does not contain any spaces or new lines. It is just formatted this way here to make it more readable.
  - `ab` is a boolean indicating whether alpha-beta pruning should be applied or not. A `true` indicates alpha-beta pruning should be used while `false` indicates it shouldn't.
  - `visualize` is a Boolean parameter which, when set to `true`, results in your program's side-effecting a display of the state information as it undergoes the different steps of the discovered solution (if one was discovered). *A GUI is not required, printing to the console would suffice.* The main value of this part is to help you see and understand the plan the agent generates. *implementing this functionality is optional but having the parameter in the function is mandatory.*

  The function returns a `String` of 3 elements, in the following format:
  `plan;score;nodesExpanded;`
  where:
  - `plan`: the sequence of actions from start to finish of the game (if such a sequence exists) separated by commas. The actions should be represented as `Player(attacker index, target index)`
  - `score`: the score of the final state reached. This utility value of a terminal state should be the total remaining health of the starting army if they win or -1* the total remaining health of the other army otherwise.

- `nodesExpanded`: is the number of nodes chosen for expansion during the search. For example:
  `A(0,0),B(1,0),A(0,1);4;25;`

  Please make sure you use the exact string formats specified for the tests to pass. In this project, you must not try to eliminate repeated states.

3. **Example:** A sample problem is provided below:
   Initial state string: `"3,1,4,2;3,5;A;"`
   Solution: `A(1,0),B(0,1),A(0,0);3;8;`



MAX
Team A: hlth: [3, 4] dmg: [1, 2]
Team B: hlth: [3] dmg: [5]
value = 3
A(0, 0)
A(1, 0)
MIN
Team A: hlth: [3, 4] dmg: [1, 2]
Team B: hlth: [2] dmg: [5]
value = -1
Team A: hlth: [3, 4] dmg: [1, 2]
Team B: hlth: [1] dmg: [5]
value = 3
B(0, 0)
B(0, 1)
B(0,0)
B(0, 1)
MAX
Team A: hlth: [0, 4] dmg: [1, 2]
Team B: hlth: [2] dmg: [5]
value = 4
Team A: hlth: [3, 0] dmg: [1, 2]
Team B: hlth: [2] dmg: [5]
value = -1
Team A: hlth: [0, 4] dmg: [1, 2]
Team B: hlth: [1]dmg: [5]
value = 4
Team A: hlth: [3, 0] dmg: [1, 2]
Team B: hlth: [1] dmg: [5]
value = 3
A(1, 0)
A(0, 0)
A(1, 0)
A(0, 0)
MIN
Team A: hlth: [0, 4] dmg: [1, 2]
Team B: hlth: [0] dmg: [5]
value = 4
Team A: hlth: [3, 0] dmg: [1, 2]
Team B: hlth: [1] dmg: [5]
value = -1
Team A: hlth: [0, 4] dmg: [1, 2]
Team B: hlth: [0] dmg: [5]
value = 4
Team A: hlth: [3, 0] dmg: [1, 2]
Team B: hlth: [0] dmg: [5]
value = 3
B(0, 0)
MAX
Team A: hlth: [0, 0] dmg: [1, 2]
Team B: hlth: [1] dmg: [5]
value = -1

Figure 1: A visualization of the full search tree with the best solution highlighted with a bold border.

4. **Groups:** You may work in groups of at most 3. Make sure you submit your team member details by October 26th at 23:59 using the following link `https://forms.gle/a7ZSHyLHB8qCjhSSA`. Only one team member should submit this for the whole team. After this deadline, we will be posting on the CMS a team ID for each submitted team. You will be using this team ID for submission. Any student enrolled in the course that does not have a submitted team will be **considered a team of one**. No random assignment will be carried out.

5. **Deliverables**

   a) Source Code submitted as the `src` folder containing both the `battle` and `tests` packages in addition to any other packages you implemented. This submission must contain all the starter classes with all the initially provided attributes and functions.

6. **Regulations** The use of online sources and LLM tools is allowed as an aid (not copy and paste) but the final work submitted must be done by the team members.

7. **Grading Criteria:**

   - Grading is based entirely on the percentage of tests passed.
   - There are private and public tests and both will contribute to the final grade.
   - The incorrect implementation of minimax alone and minimax with alpha-beta pruning (not using search or incorrectly implementing them will result in nullifying the corresponding test grades).
   - when the `ab` flag in `solve` is set to false, then alpha-beta pruning must not be used. When the flag is true then alpha-beta must be used. Not fulfilling this requirement will result in a significant deduction.

8. **Important Dates** Source code On-line submission by November 18th at 23:59. Your submission must be a .zip file containing the following and named with your team ID (team ID number only, no additional text).

   - `src` folder of the project
   - A `.txt` file containing team member names, IDs and tutorial numbers.

   Submission link: `https://forms.gle/FqTNj5z2qfnuR54CA`