A.I. Assignment-2

Nishant Kumar, 2022326

Problem statement:

- A simplified Kabaddi variant is played on a grid split into two halves.
- Two teams (Team A and Team B), each with **2 players**, start in their own half.
- Each team has a **gold treasure** at a fixed spot in its territory.
- **Objective to win:** steal the opponent’s treasure and bring it back into your own half.
- **Movement:** at each time step, a team may move **one or both** of its players **by one grid cell**.
- **Capture rule:** a player **in enemy territory** is captured (removed) if an opponent **moves onto the same cell**.

Game Modes to Simulate:

- Turn-based: Team A moves (all its players), then Team B moves.
- Simultaneous: both teams choose and apply moves at the same time.

Goal & Deliverables:

- Build a simulation environment that supports **both modes**.
- Implement four agents: **Random**, **Greedy (heuristic)**, **Alpha–Beta search**, and **MCTS**.
- Run **head-to-head matches** between agents, collect results (wins/draws, trends), and **analyze** performance.
- Present findings clearly (methods, experiments, results, insights).

Problem Formulation:

- **Game:** 2v2 Kabaddi on a rectangular grid split into two halves.
- **Assets:** Each team protects one **gold** at a fixed cell in its own half.
- **Objective:** Steal the opponent's gold and bring the carrier back into **your** half.
- **Moves:** In each time-step, a team may move **one or both** of its alive players **one cell**
  Capture rule: If players from both teams land on the **same cell in enemy territory**, all enemy-half occupants on that cell are **captured and removed**.
  If a carrier is captured, the gold **returns to base**.

Problem Formulation:

Win/Draw conditions:

- Win if your **carrier reaches your own half**.
- Win if the **entire enemy team is eliminated**.
- **Draw** if the episode reaches the **max step limit**.

- **Two environment modes:**
  - **Turn-based:** Team A moves (possibly both players), then Team B.
  - **Simultaneous:** Both teams commit moves; moves are applied together; then captures are resolved.

Algorithm Used:

Random Agent

- Samples a legal joint action uniformly from `legal_joint_actions(team)`.

Greedy Agent

- For each alive player, ranks local moves by distance to its target:
  - If **carrying gold:** target is the **home edge**.
  - Else: target is the **opponent's treasure cell**.
- Enumerates a small set of **top-k per-player moves** and chooses the joint action with the best **one-step evaluation** (or average vs sampled opponent in simultaneous mode).

Algorithm Used:

Alpha–Beta (Minimax with pruning) – Turn mode

- Depth-limited search alternating max/min by `to_move`.
- Uses the **heuristic evaluation** at leaves; terminal states return ±large values.
- Action set is pruned to **greedy candidates** to keep branching reasonable.
- In **simultaneous mode** (optional), approximates with **1-ply expectation** vs sampled opponent actions.

- **MCTS (UCT) – Both modes**
  - Tree policy: UCT selection `Q/N + c*sqrt(log(N_parent)/N)`; expand one untried action.
  - **Rollouts**: random playouts up to a depth; convert final/leaf value to a scalar and backpropagate.
  - In simultaneous mode during expansion, it **samples** an opponent action to advance the state.

Data Structures (DS) & State Representation:

- **Core dataclasses:**
  - `Move(dx, dy)` – discrete actions (↑↓←→, stay).
  - `Player(team, idx, x, y, alive, has_treasure)`.
  - `Treasure(team, x, y, at_base)`.
  - `KabbadiState(W, H, players, treasures, to_move, mode, step, max_steps)`.

- **Environment:** `KabbadiEnv`
  - `reset()`, `step_turn()`, `step_simul()`, `_resolve_captures()`, `legal_joint_actions(team)`.
  - **Turn-based** flips `to_move`; **simultaneous** uses an **intents** list to apply both moves before capture logic.

Inputs:

Board setup

- **Grid width `W`** — horizontal size of the field. Larger boards ⇒ longer paths, more branching.
- **Grid height H** — vertical size of the field. Larger ⇒ more lanes to attack/defend.

Inputs:

Episode configuration

- **Max steps per episode `max_steps`** — hard cap on the length of one game.
  Bigger ⇒ fewer step-limit draws but longer runtime per game.

- **Episodes per pairing `episodes`** — how many games we play for each matchup.
  Bigger ⇒ more stable win-rates (better stats), runtime grows linearly.

Search agent knobs

- **Alpha–Beta depth `ab_depth`** — lookahead depth for Alpha–Beta (turn mode).
  Higher ⇒ stronger search, slower moves.
- **MCTS iterations per move `mcts_iters`** — simulations tried before choosing a move.
  Higher ⇒ better decisions, heavy cost (especially in simultaneous mode).
- **MCTS rollout depth `mcts_depth`** — how far each simulation plays out.
  Higher ⇒ better long-term signal, slower per simulation.

Mode toggle

- **Include Alpha–Beta in simultaneous?** — Alpha–Beta is exact for **turn** mode; in **simultaneous** it's only an approximation and slower. Keeping **N** speeds up large experiments.

How these choices affect experiments:

- **Accuracy vs. Time:** `episodes` ↑ and search settings (`ab_depth`, `mcts_iters`, `mcts_depth`) ↑ ⇒ better quality results but longer runs.
- **Draw rate:** If many games end by time limit, increase `max_steps`; if runs are too slow, decrease it.
- **Scaling:** Bigger `W×H` tests robustness on larger maps but increases branching/search cost.

Rule of thumb: overall runtime ≈ `episodes × average_steps_per_game × (search_cost_per_move)`
(search cost rises fastest with **MCTS iterations** and **Alpha–Beta depth**).

Objective / Evaluation Function (Heuristic)

Used by Greedy, Alpha–Beta leaves, and for non-terminal MCTS rollouts.

Let `team` be the side being evaluated. The score is a **weighted sum**:

- **Alive players (ours):** `+0.5` each.
- **Our carrier:** `+8.0 − 0.7 * distance_to_home_edge`.
- **Our non-carriers:** `+ (5.0 − 0.4 * Manhattan distance to enemy treasure)`.
- **Opponents alive:** `−0.5` each.
- **Opponent carrier (they're carrying our gold):** subtract a mirrored home-edge bonus (bad for us).

Objective / Evaluation Function (Heuristic)

- **Opponents near our treasure:** subtract up to ~`4.0` scaled by distance.
- **Our treasure at base:** `+0.8` safety bonus.

Terminals short-circuit:

- **Win:** very large positive.
- **Loss:** very large negative.
- **Draw:** `0`.

MCTS rollout fallback maps heuristic v to a probability with a **sigmoid**: `1/(1+exp(-v/8))`.

Assumptions:

- Each action moves **0/1 cell** per alive player; **no diagonal** moves.
- **Pick-up** occurs by stepping on the enemy treasure cell (if `at_base`).
- If a **carrier** is captured, their opponent's treasure immediately **returns to base**.
- The field is split by `mid = W//2`; "own half" and "enemy half" use that split.
- Start positions: both teams begin near their treasure on their home side.
- In simultaneous mode:
  - Both teams' **intended positions** are computed first; then applied; **then** pick-ups and captures are resolved—removing order bias.

Takeaways:

- **MCTS is the most robust** across both modes; **AlphaBeta is excellent in turn-based** and roughly ties MCTS in simultaneous.
- **Greedy beats Random** comfortably but is **beaten by search** agents.
- **Simultaneous mode ⇒ more draws** and closer margins due to concurrent actions and more stalemates.
- **Heuristic design is key**—it directly drives AlphaBeta decisions and guides MCTS when rollouts are short.

Limitations:

- Fixed heuristic; no learning or opponent modeling.
- Limited search budget (AB depth, MCTS iters) ⇒ occasional sub-optimal choices.
- Small grid/starting symmetry can produce **many step-limit draws**, especially in simultaneous mode with strong defenders.

How the Game Ends:

- **Win:** (i) a carrier enters **own half**; or (ii) the **enemy has no alive players**.
- **Draw:** the counter reaches `max_steps` (e.g., 120/300/500), see `state.is_terminal()`.

Heuristic:

- Encourages **progress toward the objective** (approach gold; escort home).
- Rewards **possession and safety** (carrier nearing home; our gold at base).
- Penalizes **opponent pressure** and **opponent survivability**.
- Works as a **smooth potential function** that aligns with terminal rewards and guides search.

Results:

```
=== Kabbadi AI — Interactive Setup ===
Enter grid width W [9]: 9
/Desktop/AI Assignment-2/Assignment.py  per episode [120]: 120
Enter episodes per pairing (recommend 20/50/100) [20]: 20
Alpha—Beta search depth [3]: 3
MCTS iterations per move [120]: 120
MCTS rollout depth [25]: 25
Include Alpha—Beta in SIMULTANEOUS mode? [y/N]: y

=== TURN—BASED MATCHUPS ===
Running TURN:  Random vs Greedy ...
Running TURN:  Random vs AlphaBeta ...
Running TURN:  Random vs MCTS ...
Running TURN:  Greedy vs Random ...
Running TURN:  Greedy vs AlphaBeta ...
Running TURN:  Greedy vs MCTS ...
Running TURN:  AlphaBeta vs Random ...
Running TURN:  AlphaBeta vs Greedy ...
Running TURN:  AlphaBeta vs MCTS ...
Running TURN:  MCTS vs Random ...
Running TURN:  MCTS vs Greedy ...
Running TURN:  MCTS vs AlphaBeta ...
```

Mode	A	B	A_wins	B_wins	Draw	Episodes	Score	Winner
TURN	Random	Greedy	5	15	0	20	5—15—0	Greedy
TURN	Random	AlphaBeta	0	20	0	20	0—20—0	AlphaBeta
TURN	Random	MCTS	0	11	9	20	0—11—9	MCTS
TURN	Greedy	Random	16	4	0	20	16—4—0	Greedy
TURN	Greedy	AlphaBeta	0	10	10	20	0—10—10	AlphaBeta
TURN	Greedy	MCTS	4	16	0	20	4—16—0	MCTS
TURN	AlphaBeta	Random	17	3	0	20	17—3—0	AlphaBeta
TURN	AlphaBeta	Greedy	10	0	10	20	10—0—10	AlphaBeta
TURN	AlphaBeta	MCTS	8	12	0	20	8—12—0	MCTS
TURN	MCTS	Random	16	0	4	20	16—0—4	MCTS
TURN	MCTS	Greedy	15	4	1	20	15—4—1	MCTS
TURN	MCTS	AlphaBeta	10	10	0	20	10—10—0	Draw

```
=== SIMULTANEOUS MATCHUPS (INCLUDING AlphaBeta) ===
Running SIMUL: Random vs Greedy ...
Running SIMUL: Random vs AlphaBeta ...
Running SIMUL: Random vs MCTS ...
Running SIMUL: Greedy vs Random ...
Running SIMUL: Greedy vs AlphaBeta ...
Running SIMUL: Greedy vs MCTS ...
Running SIMUL: AlphaBeta vs Random ...
Running SIMUL: AlphaBeta vs Greedy ...
Running SIMUL: AlphaBeta vs MCTS ...
Running SIMUL: MCTS vs Random ...
Running SIMUL: MCTS vs Greedy ...
Running SIMUL: MCTS vs AlphaBeta ...
```

Mode	A	B	A_wins	B_wins	Draw	Episodes	Score	Winner
SIMUL	Random	Greedy	10	10	0	20	10—10—0	Draw
SIMUL	Random	AlphaBeta	6	14	0	20	6—14—0	AlphaBeta
SIMUL	Random	MCTS	1	11	8	20	1—11—8	MCTS
SIMUL	Greedy	Random	11	9	0	20	11—9—0	Greedy
SIMUL	Greedy	AlphaBeta	7	13	0	20	7—13—0	AlphaBeta
SIMUL	Greedy	MCTS	9	11	0	20	9—11—0	MCTS
SIMUL	AlphaBeta	Random	15	5	0	20	15—5—0	AlphaBeta
SIMUL	AlphaBeta	Greedy	10	10	0	20	10—10—0	Draw
SIMUL	AlphaBeta	MCTS	9	11	0	20	9—11—0	MCTS
SIMUL	MCTS	Random	10	1	9	20	10—1—9	MCTS
SIMUL	MCTS	Greedy	10	10	0	20	10—10—0	Draw
SIMUL	MCTS	AlphaBeta	9	11	0	20	9—11—0	AlphaBeta

```
Done.
```

Interpretation:

- **Ranking:** MCTS ⪆ AlphaBeta > Greedy >> Random across most matchups.
- **Turn-based:** AlphaBeta and MCTS clearly beat Greedy/Random; Greedy easily beats Random.
- **Simultaneous mode:** More **draws** and **tighter margins**–concurrent moves create stalemates and reduce the gap between strong and weak agents.
- **Heuristic impact:** The evaluation (distance to treasure, carrier progress, own-treasure safety, alive count) strongly guides AlphaBeta and stabilizes MCTS rollouts.
- **MCTS vs AlphaBeta:** Generally close-small edge varies by pairing/order; both far stronger than Greedy/Random.
- **Why many draws:** Strong defense + symmetric layout + step limit ⇒ frequent stalemates, especially in simultaneous play.

```
=== Kabbadi AI — Interactive Setup ===
Enter grid width W [9]: 9
Enter grid height H [5]: 5
Enter max number of game steps per episode [120]: 300
Enter episodes per pairing (recommend 20/50/100) [20]: 500
Alpha–Beta search depth [3]: 4
MCTS iterations per move [120]: 120
MCTS rollout depth [25]: 25
Include Alpha–Beta in SIMULTANEOUS mode? [y/N]: N

=== TURN–BASED MATCHUPS ===
Running TURN:  Random vs Greedy ...
Running TURN:  Random vs AlphaBeta ...
Running TURN:  Random vs MCTS ...
Running TURN:  Greedy vs Random ...
Running TURN:  Greedy vs AlphaBeta ...
Running TURN:  Greedy vs MCTS ...
Running TURN:  AlphaBeta vs Random ...
Running TURN:  AlphaBeta vs Greedy ...
Running TURN:  AlphaBeta vs MCTS ...
Running TURN:  MCTS vs Random ...
```

- **Ranking:** MCTS ≈ Alpha-Beta > Greedy >> Random.
- **Greedy vs Random:** Greedy easily wins (goal-directed), Random aimless.
- **MCTS vs Alpha-Beta:** Close fight; MCTS slightly more robust overall.
- **Max-steps 300:** Fewer time-limit draws, but runs longer.
- **Episodes 500:** Win-rates more stable/credible (low variance).
- **Symmetry:** Side alternation removes first-move bias; small board can cause stalemates.