**Universidad de La Sabana** Felix Mohr & Sergio Amortegui
Facultad de Ingenería 2026.1

# Fundamentos de Inteligencia Artificial

- Sheet 4-

**Environment & Rules.** Use Python with `numpy` as needed by the templates and grader. Keep the function and class signatures from the provided `.py` templates unchanged, since they are tested automatically on Gradescope.

**Files to implement.** `solution.py`.
Do not rename files or functions (only remove the '.template').

**Exercise 1** (Graph Search and Uniform Cost Search - 10 Points)

In this sheet you must complete 'solution.py' by implementing the required search components.

1. Implement `Path.actions(self)`. Reconstruct and return the action sequence from the root path to the current path head using the parent pointers.
2. Implement `general_graph_search(n0, succ, is_goal, better=None)` with graph-search behavior.

   *Note: Return '[]' wheen the root node already satisfies `is_goal`.*
3. Implement `uniform_cost_search(n0, succ, is_goal, cost)`.

   *Note: Return the optimal action sequence to a goal or 'None' if unreachable.*

   *Note: Assume non-negative edge costs.*

**Exercise 2** (Optional 0 Points) Briefly explain (3–6 lines each):

1. Why UCS is optimal with non-negative costs.
2. Why graph-search duplicate pruning is safe only under path-independent solution quality assumptions.