TODO:

* Build 50 grid world environments and store them
* Implement forward/backward/Adaptive A\* and display or store results (does not specify exactly what to do with result)
* Answer 6 questions from assignment instructions (copied at bottom)

Issues:

* Stack for best\_path is empty in certain cases when the program is run
* In other cases, the stack contains only a few nodes.
  + These path nodes are printed in the resulting maze at the end of the program, and show that the attempt to make a path has begun but was ended after expanding some nodes and pushing a few to the best\_path stack

Questions (1-6):

1.

a) Explain in your report why the first move of the agent for the example search problem from Figure 8 is to the east rather than the north given that the agent does not know initially which cells are blocked.



b) This project argues that the agent is guaranteed to reach the target if it is not separated from it by blocked cells. Give a convincing argument that the agent in finite gridworlds indeed either reaches the target or discovers that this is impossible in finite time. Prove that the number of moves of the agent until it reaches the target or discovers that this is impossible is bounded from above by the number of unblocked cells squared.

Part 2 - The Effects of Ties [15 points]: Repeated Forward A\* needs to break ties to decide which cell to expand next if several cells have the same smallest f-value. It can either break ties in favor of cells with smaller g-values or in favor of cells with larger g-values. Implement and compare both versions of Repeated Forward A\* with respect to their runtime or, equivalently, number of expanded cells. Explain your observations in detail, that is, explain what you observed and give a reason for the observation. [Hint: For the implementation part, priorities can be integers rather than pairs of integers. For example, you can use c \* f(s) - g(s) as priorities to break ties in favor of cells with larger g-values, where c is a constant larger than the largest g-value of any generated cell. For the explanation part, consider which cells both versions of Repeated Forward A\* expand for the example search problem from Figure 9.]



Part 3 - Forward vs. Backward [20 points]:

* Implement and compare Repeated Forward A\* and Repeated Backward A\* with respect to their runtime or, equivalently, number of expanded cells.
* Explain your observations in detail, that is, explain what you observed and give a reason for the observation.
* Both versions of Repeated A\* should break ties among cells with the same f-value in favor of cells with larger g-values and remaining ties in an identical way, for example randomly.

Part 4 - Heuristics in the Adaptive A\* [20 points]:

* The project argues that “the Manhattan distances are consistent in gridworlds in which the agent can move only in the four main compass directions.” Prove that this is indeed the case.
* Furthermore, it is argued that “The h-values hnew(s) ... are not only admissible but also consistent.” Prove that Adaptive A\* leaves initially consistent h-values consistent even if action costs can increase.

Part 5 - Heuristics in the Adaptive A\* [15 points]: Implement and compare Repeated Forward A\* and Adaptive A\* with respect to their runtime.

* Explain your observations in detail, that is, explain what you observed and give a reason for the observation.
* Both search algorithms should break ties among cells with the same f-value in favor of cells with larger g-values and remaining ties in an identical way, for example randomly.

Part 6 - Memory Issues [10 points]:

* You performed all experiments in gridworlds of size 101 \* 101 but some real-time computer games use maps whose number of cells is up to two orders of magnitude larger than that. It is then especially important to limit the amount of information that is stored per cell. For example, the tree-pointers can be implemented with only two bits per cell.
* Suggest additional ways to reduce the memory consumption of your implementations further. Then, calculate the amount of memory that they need to operate on gridworlds of size 1001 \* 1001 and the largest gridworld that they can operate on within a memory limit of 4 MBytes.