**Potential sources of errors**

1) **Suboptimal solution**

1.1) **Design of A\* search**

Allow nodes to be revisited if a shorter path to the same node can be found. Use a cost\_so\_far map to keep track of the *current* cost of each node. Explore a node if it hasn't been visited, or if the *new* cost of the node is less than the *current* cost.

Make sure the key used for the map is hashable. The cost\_so\_far map can double as the explored set, so it's unnecessary to keep both. Make sure the key used for cost\_so\_far is of the same type as the items in explored.

<https://edstem.org/us/courses/51451/discussion/4266217>

Content of link:

If you find your current implementation not returning an optimal path, it implies that it is not allowing nodes to be revisited if a shorter path to the same node can be found, leading to a *suboptimal* solution being returned too early. One way to address this issue is by employing a cost\_so\_far map to keep track of the current cost g(n) to each node and exploring a node if 1) it hasn't been visited, or 2) the new cost of the node is less than the current cost.

An optimal solution with the same start and goal is provided here:

A picture containing shoji, building, crossword puzzle

Description automatically generated

1.2) **Calculation of heuristic function h(n)**

Tile Puzzle: Manhattan distance. The empty tile should *not* be considered in the sum of distances.

Grid Navigation: Euclidean distance. (Optional: Octile distance is better, which can be implemented for optimization. Please note that this is entirely optional and not required to pass the autograder.)

Distinct Disks: An *admissible* (optimistic) heuristic with the largest possible *integer* value.

Hint: Let d represent the *absolute* distance between the current location of a disk n and its solved location. For d(n) = 0, h(n) = 0; for d(n) = 1, h(n) = 1; for d(n) = 2, h(n) = 1; for d(n) = 3, h(n) = 2; for d(n) = 4, h(n) = 2; for d(n) = 5, h(n) = 3; etc. What function can be used to express the relationship between d(n) and h(n)?

1.3) **Calculation of cost function g(n)**

If node n is a successor of node m, then the cost function is calculated as g(n) = g(m) + c(m, n), where g(n) is the cost function of n, g(m) is the cost function of m, and c(m, n) is the cost from m to n.

Tile Puzzle: c(m, n) = 1

Grid Navigation: c(m, n) = 1 or sqrt(2)

Distinct Disks: c(m, n) = 1

2) **IDDFS and its helper function**

2.1) **yield** **vs yield from**

The helper function should yield a solution and return once an object has been solved.

if object.is\_solved():

yield solution

return

To call the helper function recursively, use the keyword yield from.

yield from object\_successor.helper\_function(depth - 1, moves\_successor, explored)

In IDDFS, to check if a helper function returns a generator, use a boolean variable solution\_found.

solution\_found = False

...

while loop or for loop:

for solution in object.helper\_function(depth, moves, explored):

yield solution

...

2.2) **Order of conditionals**

The helper function iddfs\_helper essentially performs depth limited search. The conditionals should be arranged in the following order.

if object.is\_solved():

...

if depth == 0:

...

Reversing the order of the conditionals would return None when passed a solved state at depth == 0, causing failure at the leaf level (base case).

3) **Priority queue and tie-breaker**

Structure of node: (priority, metric2, metric3, ...)

Use a priority queue to maintain the frontier and implement a tie-breaker mechanism.

Graphical user interface, text

Description automatically generated4) **Efficiency issues (timeout, crash, infinite loop)**

Check the structure of A\* search and IDDFS.

Make sure the algorithm handles visited states correctly.

Check the calculation of the heuristic function h(n).

Check the calculation of the cost function g(n).

Make sure the algorithm appends a next move to a copy of the current list of moves.

Generally, a search algorithm performs a next move on a *deep* copy of the current object. This applies to both the TilePuzzle and DistinctDisks classes. For GridNavigation, if scene is included as one of the attributes in the constructor, there should be only *one* instance of the GridNavigation class throughout the course of A\* search. The search algorithm should not generate copies of GridNavigation objects, as doing so would be detrimental to *space efficiency* and unnecessary if designed correctly. The get\_successors function in GridNavigation should yield the coordinates of each successor (a, b) as a tuple. This is different from the get\_successors functions in TilePuzzle and DistinctDisks, which generate a new object in addition to the next move for each successor.

5) **Not passing a particular test**

Check the base case and edge cases.

Does the algorithm handle it correctly when passed a solved state?

6) **Index out of range error**

Validate indices when performing moves and generating successors. Make sure they fall within the permissible bounds.

7) **Return type**

Make sure the return type matches the problem description exactly for *all required* methods.

8) **Print statement**

Comment out *all* print statements before submission.

9) **Class design**

When implementing a class-based approach, carefully consider the design of the constructor and methods. A class is designed to represent a problem state in a search algorithm. Attributes such as path, cost and moves are more closely related to the internal state of the algorithm, as opposed to the representation of the problem state. These attributes should not be included in the constructor.

10) **Class-based approach vs function-based approach**

Hello,

For the grid navigation problem, it is recommended to create a GridNavigation class and implement the necessary components for A\* search within the class. Creating a class is unlikely to overcomplicate the solution.

The [Red Blob Games](https://www.redblobgames.com/) website provides a valuable resource for this assignment:

<https://www.redblobgames.com/pathfinding/a-star/introduction.html>

Please feel free to use the pseudocode below as a starting point and adapt it to our problem:

Text

Description automatically generated