**While building the maze, we attempted moving 2 cells at a time.**

**What would the maze look like when moving a larger number of cells?**

The vertical empty spaces mown in the grid are now always three spaces high. The amount of spaces would then be larger depending on the number. The higher the number, the larger the amount of white space.

**What would the maze look like if this number was not constant?**

In that case the vertical white spaces would vary.

**What algorithms could you use to find a path through this maze? Compare and contrast at least 2.**

1. ***DFS***

DFS (depth first search) could also have been applied. In that case each possible path would have been visited first, instead of searching from the point of origin to an ever-expanding circle of neighboring “locations”. With BFS, the algorithm we used, we move along. Depth-first search algorithms check the values along a path before moving to another path. This can be very helpful for determining *if* a path even exists

1. ***A\****

A\* is based on Dijkstra’s algorithm which computes the shortest distance from a given vertex to the rest of the vertices in a graph. Dijkstra’s looks in all directions, and the algorithm will check all neighboring vertices. We don’t need that when finding the shortest path through the maze.

Rather than simply checking the distance up to the current vertex, A\* checks the distance up to the current vertex + the estimated distance from the current vertex to the end vertex.

**How does knowing the algorithm used to generate the maze influence the best algorithm to solve it with?**

Actually it doesn’t really. The goals are different. One is to generate a maze and the other is to solve the maze. The only thing the goals have in common is exploring a path.

**As a patron picking up swag along the way, how might you best store the list of items you’ve collected?**

In a list

**If the farmer asked you to sort the items you collected before leaving the maze, what sorting algorithms would you consider using (assume a much larger list of possible swag)?**

Merge sort and quick sort

**How does the quantity and variety of swag influence your answer?**The more elements a list contains, how more elements to be compared and the more to be stored temperarely. Since the best, worst, and average time complexity are all the same in Merge sort (Θ(N\*log(N))), I’d chose merge sort.