# 1.

* Initial State
  + The agent begins at the top leftmost cell: (0,0)
* Actions
  + Check for goal, move left, right, diagonal or to next row or column depending on algorithm
* Transition Model
  + Check for parent/child connections, possible next moves

# 2.

* Breadth First
  + P – arrived at destination
  + E – field
  + A – move to next cell in row, move to next row
  + S – check cell for goal, identify cell children/parents
* Breadth First Diagonal
  + P – arrived at destination
  + E – field
  + A –move to next cell in column, move to next cell in row, move to next column, move to next row
  + S – check cell for goal, identify cell children/parents
* Depth First
  + P – arrived at destination
  + E – field
  + A –move to next cell in column, move to next column
  + S – check cell for goal, identify cell children/parents
* Depth First Diagonal
  + P – arrived at destination
  + E – field
  + A – move to next diagonal cell
  + S – check cell for goal, identify cell children/parents
* Best First
  + P – arrived at destination, cost of next step
  + E – field
  + A – move left, move right
  + S – check cell for goal, identify cell children/parents
* Best First Diagonal
  + P – arrived at destination, cost of next step
  + E – field
  + A – move diagonally
  + S – check cell for goal, identify cell children/parents
* Uniform Cost
  + P – arrived at destination, cost of next step
  + E – field
  + A - check cell, move to next cell in column, move to next column
  + S – check cell for goal, identify cell children/parents
* Uniform Cost Diagonal
  + P – arrived at destination, cost of next step
  + E – field
  + A - move to next cell in column, move to next cell in row, move to next column, move to next row
  + S – check cell for goal, identify cell children/parents
* A Star
  + P – arrived at destination, cost of next step relative to heuristic
  + E – field
  + A – move to next cell in column/row, move to next column/row, identify ideal path
  + S – check cell for goal, identify cell children/parents, evaluate cost of path
* A Star Diagonal
  + P – arrived at destination, cost of next step relative to heuristic
  + E – field
  + A – move diagonally, identify ideal path
  + S – check cell for goal, identify cell children/parents, evaluate cost of path

# 3.

* Observability – fully
* Agent Type - single
* Deterministic/Stochastic – deterministic
* Episodic/Sequential - sequential
* Static/Dynamic - static
* Discrete/Continuous - discrete
* Known/Unknown - known

# 4.

* . – An unsearched node
* C – The open list
* X – A searched node
* \_ - The optimal path

# 5.

All algorithms ran with a 20x20 square field

* Breadth First
  + 18738.998 ms
* Breadth First Diagonal
  + 16883.297 ms
* Depth First
  + 8966.847 ms
* Depth First Diagonal
  + 835.733 ms
* Best First
  + 1730.042 ms
* Best First Diagonal
  + 837.292 ms
* Uniform Cost
  + 18605.024 ms
* Uniform Cost Diagonal
  + 18261.864 ms
* A Star
  + 31248.248 ms
* A Star Diagonal
  + 1527.693 ms

# 6.

* Fastest Algorithm: Depth First Diagonal
  + No surprise, the algorithm with a diagonal approach performed best. In general, diagonals performed better in this example in part due to the positioning of the goal, but also because of their ability to quickly span the field. The heuristic diagonal approach was nearly the same speed.
* Slowest Algorithm: Breadth First No Diagonal
  + Since our destination is as far away from us diagonally as possible, this algorithm forces us to check every other cell first, hence it performs very poorly

# 7.

See code.

# 8.

All algorithms ran with a 20x20 square field with a max of 50 obstacles

* Breadth First
  + 16330.241 ms
* Breadth First Diagonal
  + 15697.667 ms
* Depth First
  + 7886.700 ms
* Depth First Diagonal
  + 1019.533 ms
* Best First
  + 1784.081 ms
* Best First Diagonal
  + 929.918 ms
* Uniform Cost
  + 37.207 ms
* Uniform Cost Diagonal
  + 16340.718 ms
* A Star
  + 25121.806 ms
* A Star Diagonal
  + 1925.117 ms

# 9.

* Breadth First
  + When running into several connected obstacles (a wall of sorts) it would temporarily move past and explore beyond that section of the field, although later on, when it found a route inside, it did explore all the nodes on the other side of the wall.
* Breadth First Diagonal
  + This was largely unaffected and essentially just hopped over obstacles.
* Depth First
  + This behaved very differently with obstacles and bounced around almost like a game of snake. It can get caught up in its explored path but did eventually double back and find its way to the goal.
* Depth First Diagonal
  + This similarly changed route around obstacles, but still went very quickly, so there wasn’t a dramatic effect.
* Best First
  + I think because of the preference to the first dimension, obstacles in this case have very little effect. The path did occasionally shift down, but with on exception, never backed up and took only one more step in total than the original best-first.
* Best First Diagonal
  + Very similar to depth-first diagonal. Routed around some obstacles, but more or less went straight to goal.
* Uniform Cost
  + Coincidentally, my first run on uniform cost started with an obstacle on either side of the start, and the algorithm was unable to continue. Obviously a specific use case, but interesting that there is no ability to recover in such a situation.
* Uniform Cost Diagonal
  + Very similar to breadth-first diagonal, seemed largely un-influenced by obstacles.
* A Star
  + This looks a lot like breadth-first when it is checking for routes, although it doesn’t appear to go back and look in unexplored areas it missed initially, due to the heuristic. Ultimately, the chosen route resembles that of best-first with obstacles.
* A Star Diagonal
  + Much like the other diagonal examples, it quickly diverted around obstacles and make a fairly direct path to the goal very quickly.