**Part1:**

**Q: 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.**

Understanding how each presumed optimal path is calculated, we can see why the agent’s first move in this case is to the east.

In the start position of *E2*, with the agent looking to make its first move, it has three unblocked cells as options and is not aware of any blocked cells yet beyond this (excluding the end of the grid world to its south).

The program then calculates respective *f* values for the estimated distance from each option available (ignoring current position), *E1* (west), *D2* (north), and *E3* (east), to the goal state. These calculations are:

*f(E1) := g(E1) + h(E1)*, expanded to *f(E1)=* distance of start *E2* → *E1* *(1)* + Manhattan distance of *E1* → goal *E5* *(4) = 5*.

*f(D2) := g(D2) + h(D2)*, or *f(D2)= [E2 → D2 (1) + D2 → E5 (4) = 5]*.

*f(E3) := g(E3) + h(E3)*, or *f(E3)= [E2 → E3 (1) + E3 → E5 (2) =* ***3****]*.

The move with the lowest *f* value is chosen, hence the agents’ move being to the east *(E3)*.

**Q: 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.**

There are two parts to why the total number of moves *(m)* is bounded by the number of unblocked cells squared *((u)^2).*

1. Since discovered cells during an A\* call are remembered, with previously pathed cells being put into a closed list and unavailable to be backtracked on, the number of tracked cells *(t)* for one computed path is always less than or equal to the number of unblocked cells *(u)*, or *t <= u.*
2. Because paths on a cell are remembered, and will never be recomputed, the number of ComputePath() calls *(p)* will always be less than the number of unblocked cells in a grid world, or *p <= u.*

As a result, *m*, equaling *t\*p*, will always be *<= u^2*.

Considering this, if we know the gridworld’s size to be a finite number, and the total number of moves *m* to be less than the size of the gridworld, *m* is resultantly also a finite number which will result in a conclusion in finite time.