Names: Uday Patel (uyp4) and Kush Patel (kmp355) ****worked on all parts together****

To run program run the **agentOne.py, **agentTwo.py**, **agentThree.py**, **agentFour.py** files**

Question 1: What algorithm is most useful for checking for the existence of these paths? Why?

When we created our method we used a DFS tree search to find if paths exist in the generated mazes. After learning the A* algorithm and using it for the rest of the project this would have been the most useful algorithm for checking the path. This is because A* strictly only checks the path that is closest to the goal node, avoiding checking longer unnecessary paths.

Question 2: Agent 2 requires multiple searches - you'll want to ensure that your searches are efficient as possible so they don't take much time. Do you always need to replan? When will the new plan be the same as the old plan, and as such you won't need to recalculate?

We don't always have to recalculate a new route. If all the nodes on the current path are not on fire as the fire advances then we do not need to recalculate a new route. If any of the nodes on the path catch on fire then it would be necessary to recalculate a route.

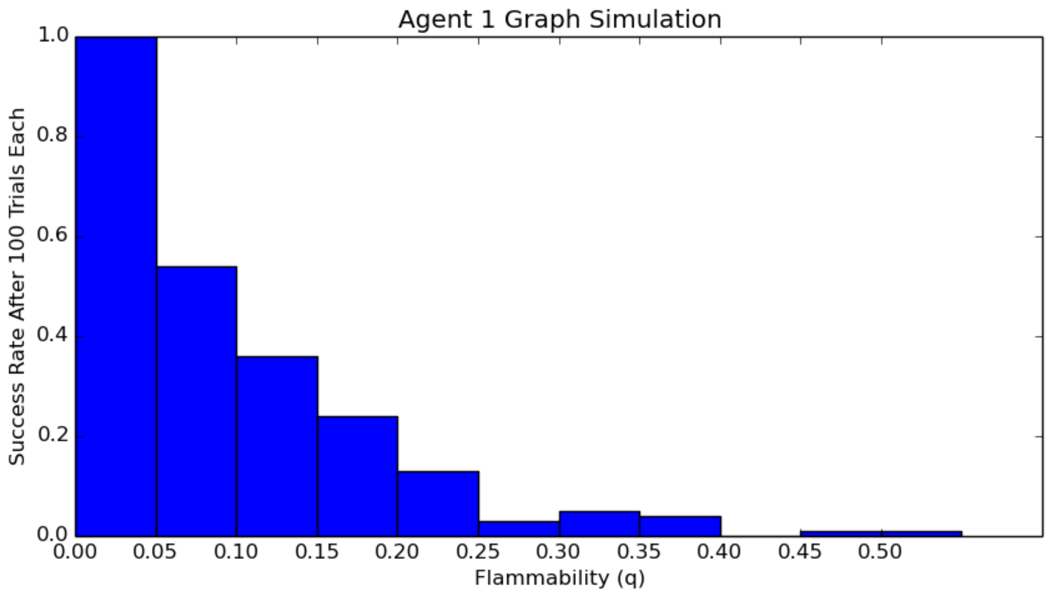
Question 3: Agent 3 requires multiple searches - you'll want to ensure that your searches are efficient as possible so they don't take much time. Additionally, if Agent 3 decides there is no successful path in its projected future, what should it do with that information? Does it guarantee that success is impossible?

If the projected fire in 3 steps shows there is no path that DOES NOT mean success is impossible. This is because the future map is just a prediction of what might happen and might not become "reality". In our project, we checked 2 steps in the future if 3 steps led to no paths. If 2 steps also gave no valid paths we checked one in the future and so on.

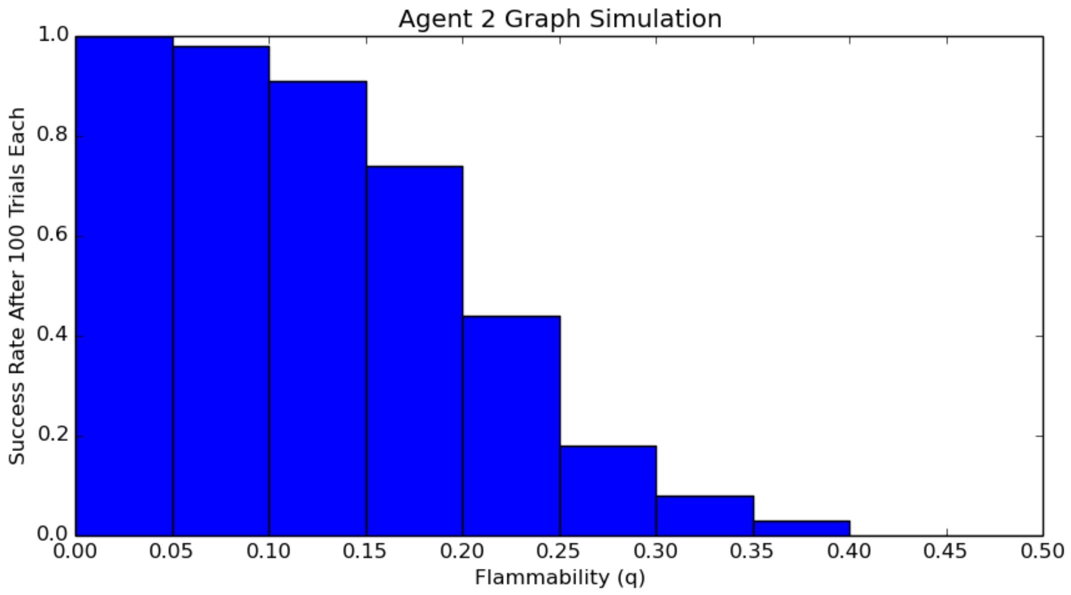
AGENT 4: For our agent 4 we took future predictions into account but tried to make it more efficient than agent 3. Agent 4 will look 3 steps into the future and plan a route to use in reality. Then agent 4 will move 3 steps without recalculating if any of the paths catches on fire in "reality", then check 3 steps into the future again. If any of the paths catch on fire at any time, only then will we recalculate. The hope of this is to still get the results on par with agent 3, trading intelligence for slightly better efficiency.

- For all 4 agents we ran tests on the following q values:
    - q = [0.0 , 0.05 , 0.1 , 0.15 , 0.2 , 0.25 , 0.30 , 0.35 , 0.4 , 0.45 , 0.5]
- We used these values because we realized any values above 0.5 had almost a guaranteed absolute failure rate.
- We ran each agent against each of the above q values 100 times

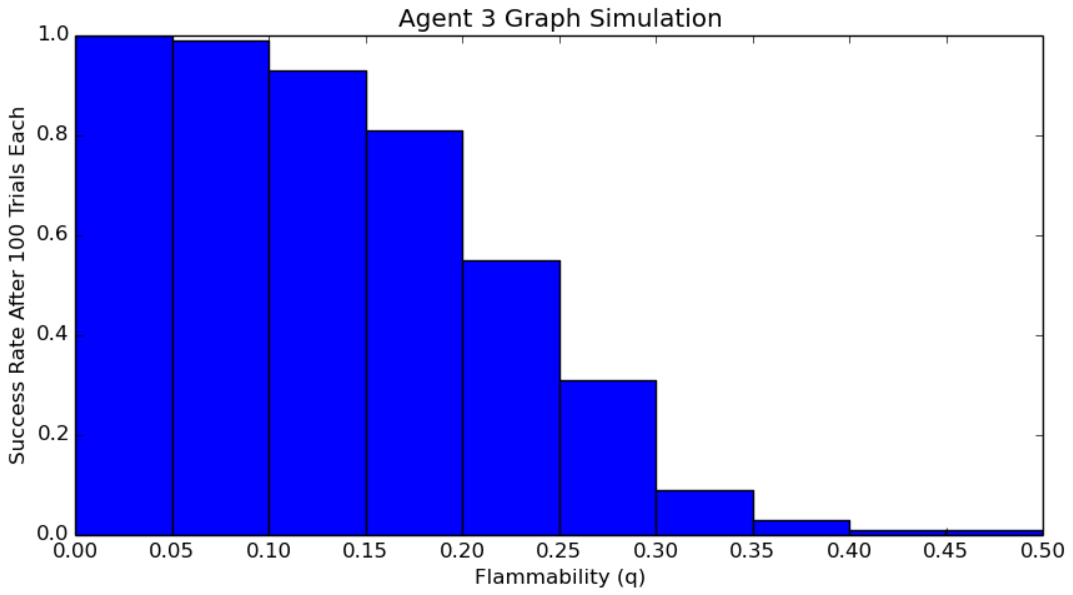
Agent 1



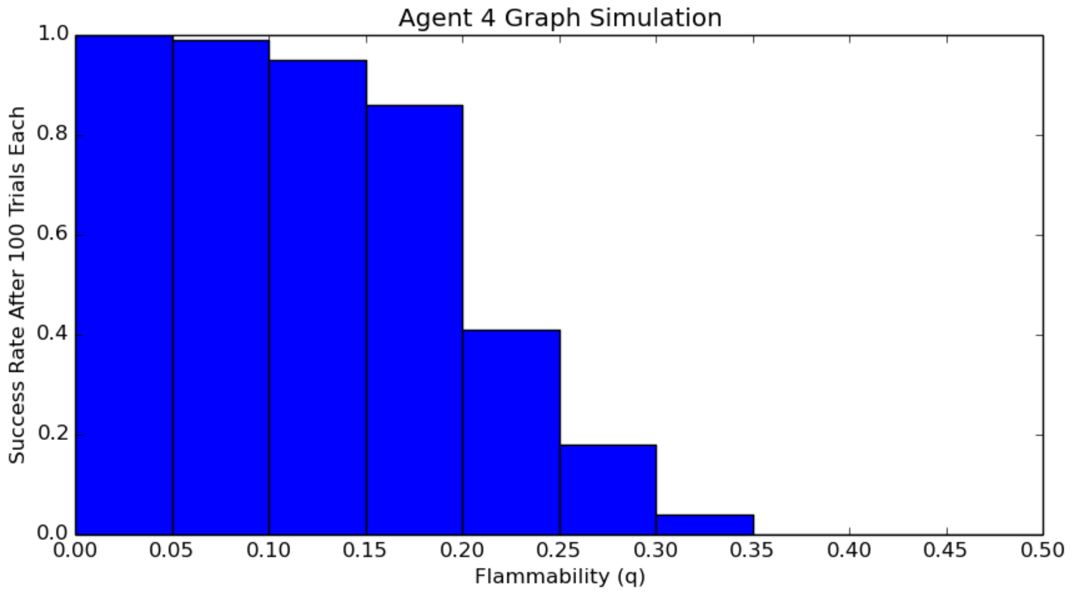
Agent 2



Agent3



Agent 4



Question: How did Agents 1, 2, and 3 compare?

Agent 1 ran much faster than the two others. This is because we did not account for the fire. Consequently, agent 1 had a much lower success rate than the other two agents. For the lower flammability rates, agents 2 and 3 had very similar success rates. As q got larger agent 3 was able to outperform agent 2 due to its ability to "predict" where the fire was going to spread. Because of this agent, 3 took longer to run than agent 2. Agent 3 sacrifices efficiency for intelligence.

Question: How did your Agent 4 stack up against the others? Why did it succeed, or why did it fail?

Agent 4 has some successes as well as some shortcomings. For lower flammability, agent 4 ran as expected. It was much faster than agent 3 while still performing similarly. The shortcomings came with higher flammabilities. The trade-off for efficiency meant the program could not adapt as quickly as agent 3. Although in some cases Agent 4 was marginally better than Agent 3, it fell short in having the necessary intelligence to navigate the maze at higher flammabilities.

Question: If you had more time and resources, what's the most advanced Agent 5 you could imagine?

The most advanced Agent we can think of would likely combine reality with near-accurate future predictions and run relatively efficiently. Additionally, A* could have an extra Heuristic (in addition to g(n) and h(n)) that looks at nearby fire probabilities, specific to this problem. It could take into account how the fire might move in its calculation of f(n). In reality, these future assumptions can be put up against the path generated in reality. If both are consistent with enough confidence, go ahead. Otherwise, do marginal analysis to see which next step has the highest yield.