Intro to AI Project 1
Author: Sihua Zhou, NetID: sz583
Note: README is included in the project folder in case you need to run the code.

Project 1: Out of the Frying Pan and Into the Flyer

Purpose of this project: "It is another day on the deep space vessel Archaeopteryx, and you are a lonely bot. You're responsible for the safety and security of the ship while the crew is in deep hibernation. As an example, in the case that there is a fire on the ship, you are responsible for pressing the button to trigger the fire suppression system. Of course, you have to get to it first." (Cited from project 1 writeup)

Strategies and algorithm implemented in this project (for each bot):

Bot 1 - This bot plans the shortest path to the button, avoiding the initial fire cell, and then executes that plan. The spread of the fire is ignored by the bot.
Algorithm: Simple BFS

Bot 2 - At every time step, the bot re-plans the shortest path to the button, avoiding the current fire cells, and then executes the next step in that plan.
Algorithm: Simple BFS

Bot 3 - At every time step, the bot re-plans the shortest path to the button, avoiding the current fire cells and any cells adjacent to current fire cells, if possible, then executes the next step in that plan. If there is no such path, it plans the shortest path based only on current fire cells, then executes the next step in that plan.
Algorithm: Simple BFS

Bot4 – A bot of your own design.
Algorithm: A*(A star) Search Algorithm

Detailed Explanation on the Design and Algorithm for Bot 4 (Q1):

- Usage of the general A star search algorithm to for every time stamp to find a way to reach the button (Referred Pseudocode from canvas note).
    - Usage of Priority Queue, HashMap, Stack (for DFS to calculate heuristic).
    - Priority Queue is sorted in ascending order (representing the least cost goes first)
- In this specific implement (bot4), the priority is calculated with tempDist(the cost to the child) + heuristic(estimated cost to the goal).
    - Note: since the child all have cost 1 from its parent, then the heuristic is significant part for determining the priority of the child.

- In detail:
  - tempDist = cost of the parent + the cost from parent to child
  - heuristic in this case is determined with two factors:
    - the distance from the child to the button (child2button)
    - the distance from the child to the fire (child2fire)
  - The reason of using DFS is to find **a fire cell** and use the coordinate of that fire cell to calculate the distance from that fire cell the button as one of the factors. The child and button coordinate is known by the bot, so the distance from the child and button can be calculate with performing another algorithm.
    - Reason for using the DFS in BFS for searching a fire cell is related to time complexity concern that BFS gives accurate result for calculating heuristic, where DFS spit out result faster that BFS in this case.
- More for the heuristic:
  - The distance factors mentioned above is compared in these three conditions:
    - **Child2button < child2fire**
      - Since the child favors the bot for getting closer to the button and away from fire.
      - In this case, it will get assigned with **heuristic = child2button** and get execute earlier in the priority queue, since the priority queue is in ascending order.
    - **Child2button > child2fire**
      - The child is in least favorable condition to the bot that led the bot getting closer to fire and away from the button.
      - In this case, it will still get assigned with **heuristic = child2button**, because it is larger than child2fire and get least priority on executing (priority queue is in ascending order)
    - **Child2button == child2fire**
      - If the distance for the bot to button and fire are the same then the bot has to decide the heuristic for this child depends on the probability of the fire.
      - The formula below:
        - **Child2button – (child2fire * (1 – probability))**
      - In the formula, the higher the probability for fire spreading the less priority (high heuristic) the bot gives for this child. Because the higher the probability the fire is the more risk the bot has to take if the bot ran into that cell. On the opposite the less the probability is the safer the bot to go through that way, cause the probability of catching fire by entering this child cell is low.

Statistic of the Result(Q2):

Two simulations were performed:
#1:

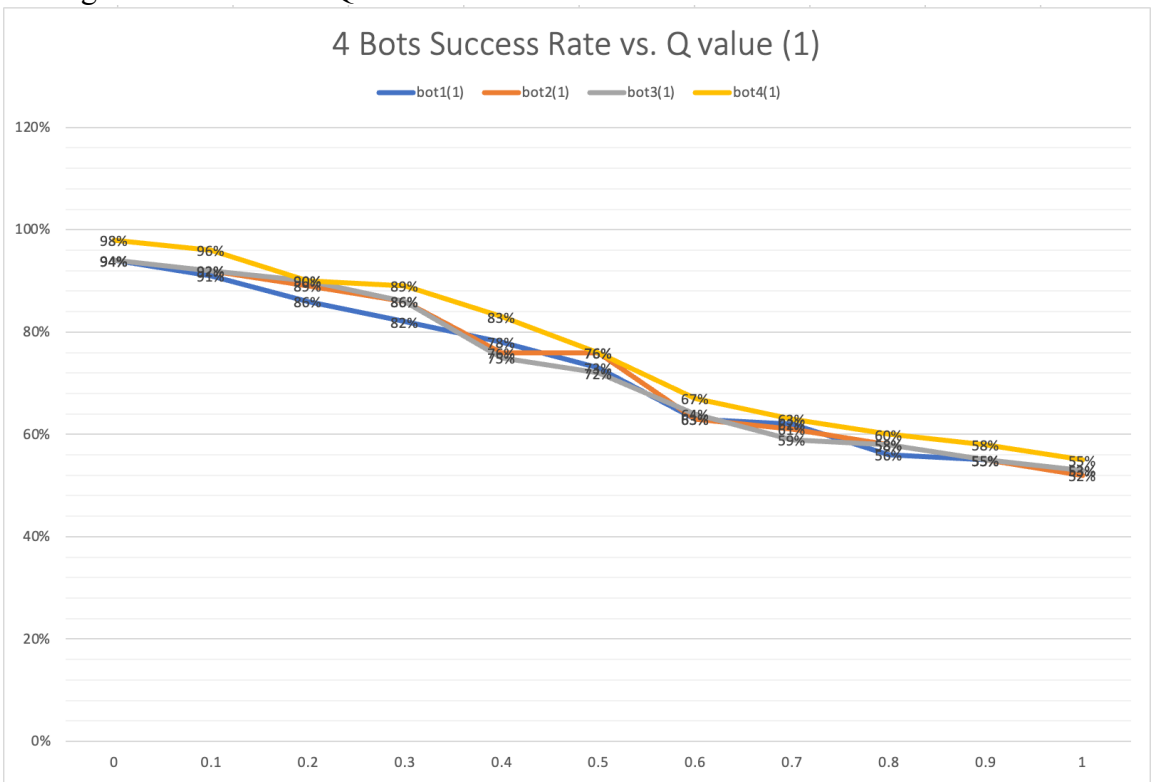
```
Bot 1: 94 91 86 82 78 73 63 62 56 55 53
Bot 2: 94 92 89 86 76 76 63 61 58 55 52
Bot 3: 94 92 90 86 75 72 64 59 58 55 53
Bot 4: 98 96 90 89 83 76 67 63 60 58 55
Time cost: 7323.92 seconds
```

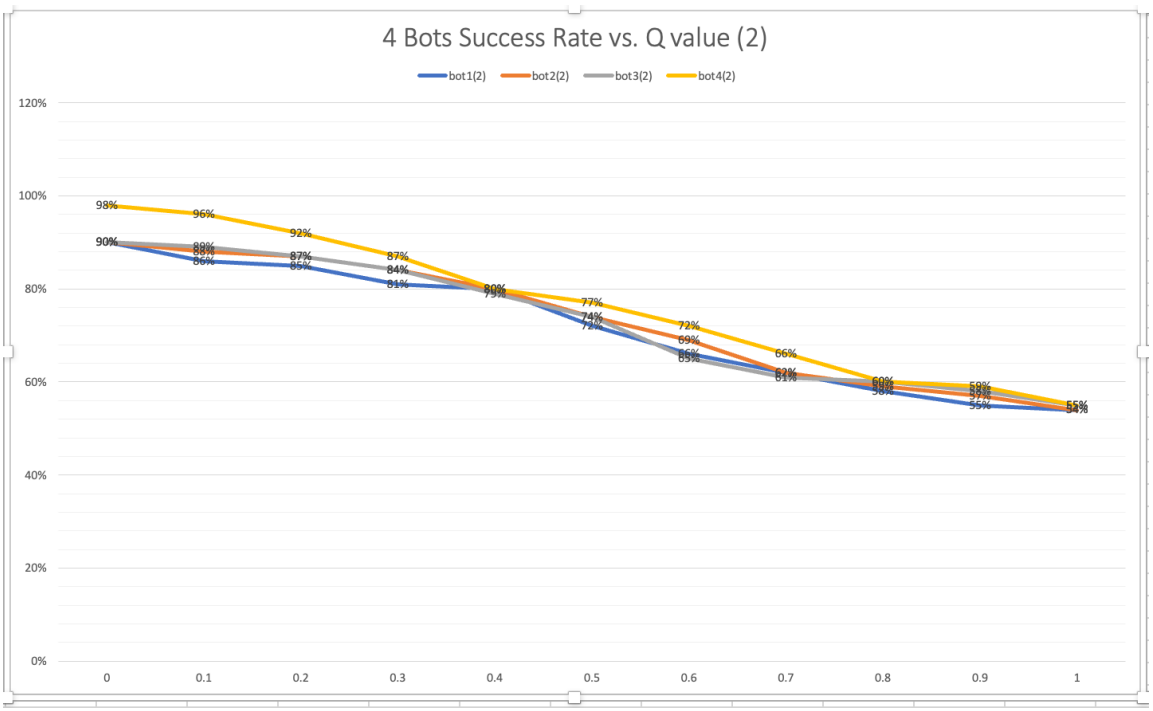
#2:

```
Bot 1: 90 86 85 81 80 72 66 62 58 55 54
Bot 2: 90 88 87 84 80 74 69 62 59 57 54
Bot 3: 90 89 87 84 79 74 65 61 60 58 55
Bot 4: 98 96 92 87 80 77 72 66 60 59 55
Time cost: 5845.66 seconds
```

Total trials number is 100 * 11 * 4 * 2 = 8800 trials(times)
Explanation: 100 ship layouts are generated and time for all 4 bots with 11 different probabilities for **q** and ran for 2 times.

Average Success Rate vs Q value:





Observation: Bot4 generally has the better performance compared to the other 3 bots.

Note: when q value is 0, in general, the situation should be 100% for success rate. However, there are rare case that random position of the fire, bot and button can at a position that at initial stage there is no way to get to the button. Therefore, the success rate for all bots at q value 0 wouldn't be 100%

When bots fail or get trapped by trapped by the fire, why do they fail? Was there a better decision they could have made that would have save them? Why or why not?(Q3)

- Instead of talking about the situation in general, in the cases were happened in my simulations for the bots, the bot would fail because of three reasons. First, the bot itself catches on fire. Second, the bot got trapped by fire and the button didn't. Lastly, the button catches on button and the bot failed to find a way to go to the button.
- Specifically in my simulation for all bots, bot 1 to 3 can definitely make better decision on the information they had, for example, current fire cell list (except bot 1) and button position. For trails performed by bot 1 to 3 that are failed can make better prediction for fire and find a way out, like bot 4, instead of in pursue of finding the shortest path.
- In addition, for the bot 4 as far as I can tell and considering about my capability that there are not much of improvement can be done by me to make better decision to reach the goal. Most of the time my bot 4 strategy ends not because of it catches on fire but get trapped from escape from fire and get to the button at the same time or the button itself catches on fire where not additional action can be taken to save the situation.

Speculate on how you might construct the ideal bot. What information would it use, what information would it compute, and how?(Q4)

- In my opinion, my ideal bot for the given situation, I would calculate one more distance between button and fire. And add that factor into the consideration of heuristic for better outcome. In addition, another feature I wasn't capable to implemented in this project is to run simulate for the current ship for more time stamps and draw an estimation of where the fire is going in the future then base on that estimation to make a better decision on next move.