Chongen Zheng (cs353), Zihao Zheng (az376), Yiming Huang (yh714)

1. The design and algorithm for the bots

The initial setting of bot 1 is to randomly place the bot and leak in the open cell, and the leak will not appear in the initial detection square. Mark the cells outside the initial detection square as possible leak, and mark the cells in the initial detection square as no leak.

In the first step, the bot randomly selects up, down, left, and right directions and moves to the nearest open cell. For each subsequent move, the bot uses BFS to find and move to the possible leak closest to it, and then senses it. The bot's detection method is to compare the position of the leak with the cells in the detection square to see if it is consistent. If a leak is detected, the cells in the detection square are put into a queue, except cells that have been marked ”no leak“. At the same time, mark all cells outside the detection square as no leak. The bot accesses the cells in the order in which they pop out of the queue to determine the final leak. If the detection finds no leak, mark all the cells in the detection square as no leak, and then go to the nearest possible leak for detection.

Repeat the above movement and detection methods until the leak is found and stopped.

Bot 2 is an optimization based on bot 1. The same is done by randomly assigning bot and leak to any open cell in the ship. Mark the cells in the initial detection square as "no leak", and set all other cells as "possible leak". In bot2, under normal circumstances, the distance between the bot moving and finding the target cell is 2k+1.

In the first step, Bot goes to the cell with a straight-line distance of 2k+1, and the direction can be up, down, left, or right. If the cell at a straight-line distance of 2k+1 from the current location is closed, its location will be stored for use in the next move. Then the bot puts the open and unvisited cells in the four directions immediately adjacent to the closed cell into a list, uses BFS to access them sequentially, senses and removes the current cell from the list until the list is empty or a leak is found. Then based on the value of the stored closed cell, the location 2k+1 from it is calculated and set as the destination. If the cell is open, go directly; otherwise, perform the above operation, that is, traverse its open neighbor.

In addition, the bot gives priority to finding the next cell with a distance of 2k+1 in the direction of the last movement. Unless the cell at a distance of 2k+1 has been visited or is not in the ship. In this case, the bot looks for the currently stored closed cell in the four directions of up, down, left, and right, a distance of 2k+1 from it, and enters the unvisited cell on the ship. If the bot is trapped (surrounded by "no leak" cells), it will try to find the nearest unexplored cell and move to it to continue searching or go backward and return to the previous sensing location to explore other directions..

After each sense, if there is no leak in the square, the cell in the square is recorded as "no leak". When a possible leak is detected in the detection square, put the open cells in the square into a list. The bot traverses the cells in the list. If it is determined that it is not a leak, the cell is recorded as "no leak". Otherwise, record all cells except the leak as "no leak" and end.

The initial setting of bot3 is to randomly place the bot and leak in open cells. Then assign an initial probability to each open cell. This initial probability is based on the number of open cells and distributed evenly, ensuring that their probabilities sum to 1.

In the first step, the bot randomly selects four adjacent open cells above, below, left, and right to enter, and then updates the probability of each unvisited open cell according to formula P1(leak in cell i | no leak in curr)=$\frac{P(leak\ in\ i)}{1-P(leak\ in\ curr)}$. And set the probability of the visited cell to 0. Then try to listen to beep in place. If beep is heard, update the probability of unvisited open cells in the entire ship according to formula P2(leak in i | heard YES beep in curr)=

$\frac{P(leak\ in\ i)(e^{-\alpha(d[i,leak]-1})}{\sum_{all\ cells\ j} P(leak\ in\ j)(1-e^{-\alpha(d[j,leak]-1})}$; otherwise, update according to formula P3(leak in i | heard no beep in curr)= $\frac{P(leak\ in\ i)(1-e^{-\alpha(d[i,leak]-1})}{\sum_{all\ cells\ j} P(leak\ in\ j)(1-e^{-\alpha(d[j,leak]-1})}$.

After the update is completed, the bot sets the cell with the highest probability in the entire ship as the destination and uses BFS to go there. During the process of going to the destination cell, each cell updates the probability of all unvisited open cells according to formula P1 = $\frac{P(leak\ in\ i)}{1-P(leak\ in\ curr)}$. Until the destination cell is reached, if no leak is found, P1update will be used. When finished, listen to the beep. Finally, update based on detected beep in current cell or no beep in current cell. Repeat the above steps until the leak is found.

The design of Bot 4 is based on Bot 3 with some modifications. The settings in the initialization phase are the same, assigning an initial probability to each open cell, and keeping the probability of closed cells at 0. This initial probability is based on the number of open cells, ensuring that their probabilities sum to 1. In each cycle, the bot makes a movement decision based on the current situation. If the current position of the bot is the same as the leaked position, the loop ends and the game is successful.

The difference is that bot 4 divides the ship into four areas. The number of beeps for each zone is recorded independently. Determine whether the bot "hears" the leak based on the distance between the bot and the leak and the alpha value. The probability of cells is updated based on the information sensed by the bot (beep or silence). If a beep is sensed, update the leakage probabilities for all cells, increasing the probabilities for those cells that are closer. If beeps are not sensed, reduce leak probability for closer cells. Whenever the bot moves to a new location, the leak probability needs to be updated across the map based on the bot's new location and whether it detects a leak sound (beep). In addition to updating the probabilities for all cells, it also increases the beep count for the corresponding region.

When the bot moves, it first considers the "area with the most beeps". It will preferentially move to the cells with the highest probability within the area. If there are multiple cells with the same highest probability, the closest one is chosen. If the distance is also the same, choose randomly. Once the target location for the next step is determined, use BFS to find the path from the current location to the target location. The bot moves to the next location along the found path.

This process repeats until the bot finds the location of the leak.

Bot 5 detects and moves in almost the same way as bot 1. The main difference between them is that after detecting the first leak, remove that leak and mark its location as not a leak. At the same time, cells outside the current detection square will not be marked as not a leak, and after visiting all possible leak cells in the current detection square, BFS will be used to find and go to the possible leak outside the nearest current detection square to detect and find Another leak. The search logic method for the remaining leak is the same as bot1.

Bot 6 is similar to bot2. The difference between them is that after bot6 finds the first leak, it only marks the cells in the detection square as no leak and does not change the cells outside the square. The main difference between them is that after detecting the first leak, remove that leak and mark its location as not a leak. At the same time, cells outside the current detection square will not be marked as not a leak, and after accessing all possible leak cells in the current detection square, calculate based on the current position that the straight lines of 2k+1 in the four directions of up, down, left, and right are on the ship and are not. The visited cell is the destination. If the bot is trapped (surrounded by "no leak" cells), it will try to find the nearest unexplored cell and move to it.

Keep repeating move and sense until you find another leak.

Bot 7 detects and moves in much the same way as bot 3. The main difference between them is that they do not stop after detecting the first leak. Instead, remove that leak and set the probability of the cell where it is located to 0. Then update the unvisited open cells of the entire ship by the formula P1(leak in cell i | no leak in curr)=$\frac{P(leak\ in\ i)}{1-P(leak\ in\ curr)}$. After that, beep Listening, the probability is updated with only a single missing point. If there is, update according to formula P2(leak in i | heard YES beep in curr)= $\frac{P(leak\ in\ i)(e^{-\alpha(d[i,leak]-1})}{\sum_{all\ cells\ j} P(leak\ in\ j)(1-e^{-\alpha(d[j,leak]-1})}$; otherwise, update according to formula P3(leak in i | heard no beep in curr)= $\frac{P(leak\ in\ i)(1-e^{-\alpha(d[i,leak]-1})}{\sum_{all\ cells\ j} P(leak\ in\ j)(1-e^{-\alpha(d[j,leak]-1})}$. The bot selects the cell with the highest probability in the entire ship as the destination and uses BFS to go there. The remaining steps are the same as bot3, until another leak is found and stopped.

Bot 8 adds a leak based on bot 3, so their logic is similar. If the bot reaches one of the leaks, the total leak count is decremented and the leak is removed from the list. If all leaks are found, the game is over. Each time the bot moves to a new location, the distance from the bot's position to each leak point is calculated and combined with the alpha value to determine whether the leak sound is "heard". When Bot8 detects a beep, it will update the probability based on the positions of the two missing points, so the formula used is: P2(leak in i | heard YES beep in curr)= $\frac{P(leak\ in\ i)(e^{-\alpha(d[i,leak1]-1})(e^{-\alpha(d[i,leak2]-1})}{\sum_{all\ cells\ j} P(leak\ in\ j)(1-e^{-\alpha(d[j,leak]-1})}$, P3(leak in i | heard no beep in curr)=

$\frac{P(leak\ in\ i)(1-e^{-\alpha(d[i,leak]-1}(e^{-\alpha(d[i,leak2]-1}))}{\sum_{all\ cells\ j} P(leak\ in\ j)(1-e^{-\alpha(d[j,leak]-1})}$. Based on the detection results (whether leakage sound is heard or not), the leakage probability of the entire grid is updated. If the sound is heard, the probability of cells close to the bot increases; if it is not heard, the probability of these cells decreases. The bot selects the cell with the highest probability as the next target and calculates the path to that cell using the breadth-first search (BFS) algorithm. The bot moves along the calculated path to the next cell. It ends until all leaks are found.

Bot 9 place a bot and two leak points in a randomly selected open cell. Each open cell is assigned an initial leakage probability, distributed evenly based on the total number of open cells on the ship. If the bot reaches a leak point, update the total leak point count and remove the point from the leak list. If all leaks are found, the game is over.

Divide the ship into four areas in bot9 and track the number of beeps received in each area, prioritizing these areas based on the frequency of the beeps. Update the leakage probability of the bot's current position, and update the leakage probability of the entire ship based on whether the bot detects the leakage sound. That is, the relative distance between the two leak points and the current position of the bot is calculated, and the leak probability of each cell is updated based on these two distances. The bot prioritizes the cells in the highest beep area, selecting the cell with the highest leak probability as the bot's next target. If all areas have the same number of beeps, it selects the cell with the highest probability in the entire grid. Use the Breadth First Search algorithm (BFS) to find the path from the bot's current position to the target position and move the bot.

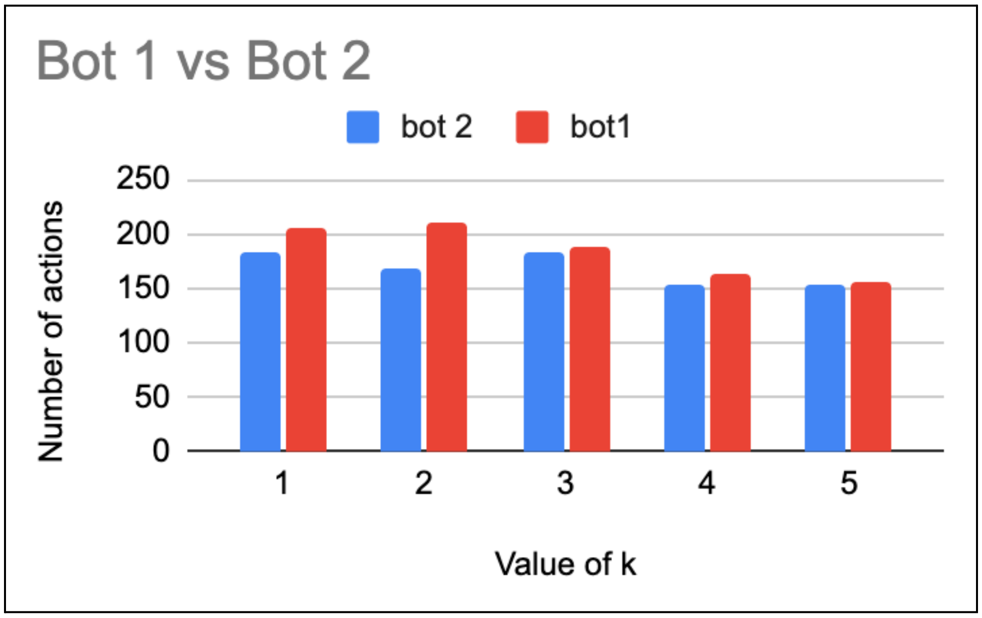
2. Updated Probability:

Bot 3/4: Initially, all possible leak locations (i.e. open cells) are given equal probability. This is a uniform distribution because initially there is no information about the location of the leak. Whenever the bot moves to a new cell, it can be sure that this cell is not a missing point. Therefore, the missing point probability of this cell is updated to 0, while the probability of other cells is increased. If the bot hears a beep in a certain cell, it means the leak is probably closer to it. At this time, the probability of missing points for cells closer to the bot should be increased, and the probability of other cells should be reduced. If no beeps are heard, the probability of missed points should be reduced for cells closer to the bot, and the probability for other cells should be increased accordingly. The above part is the probability update method of bot3 and bot4 by these formula: P1(leak in cell i | no leak in curr)=$\frac{P(leak\ in\ i)}{1-P(leak\ in\ curr)}$, P2(leak in i | heard YES beep in curr)= $\frac{P(leak\ in\ i)(e^{-\alpha(d[i,leak]-1})}{\sum_{all\ cells\ j} P(leak\ in\ j)(1-e^{-\alpha(d[j,leak]-1})}$, P3(leak in i | heard no beep in curr)= $\frac{P(leak\ in\ i)(1-e^{-\alpha(d[i,leak]-1})}{\sum_{all\ cells\ j} P(leak\ in\ j)(1-e^{-\alpha(d[j,leak]-1})}$

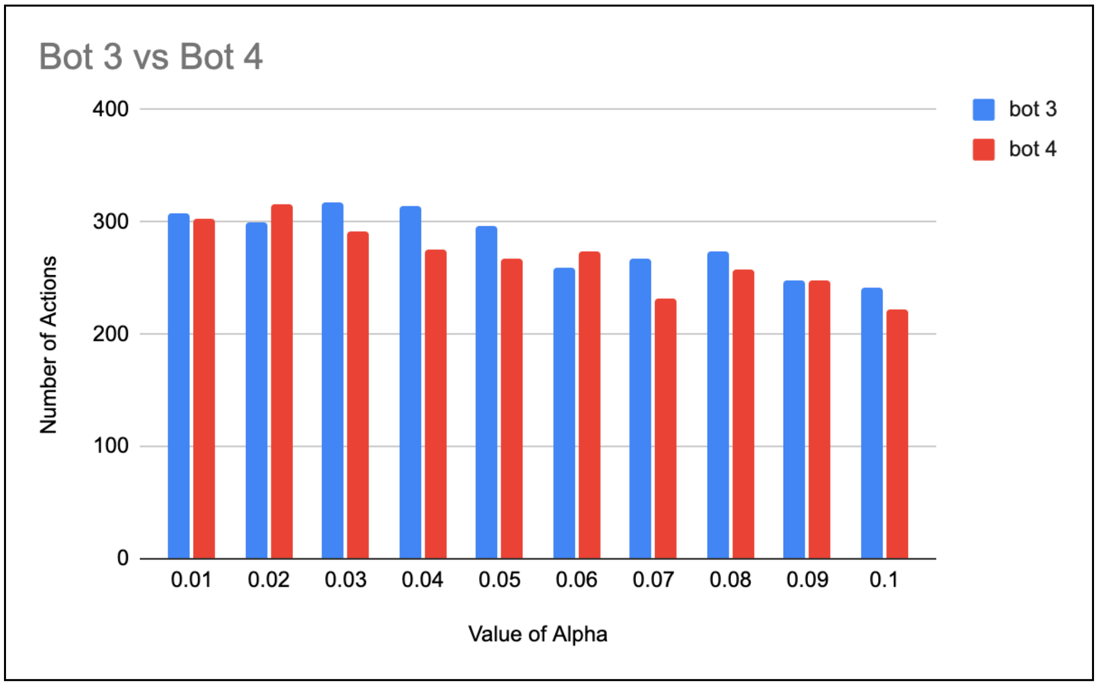
Bot 8/9: The calculation of probability must take into account each leak situation. At the same time, the key to the probabilistic update is to take into account the bot's current position, the signals it hears (such as "beeps"), and the possible locations of the missing points. If no beeps are detected, the probability increases for cells further away from the bot's position. When a beep is detected, the probability of being close to two leaking units increases. So, the formula are used by P2(leak in i | heard YES beep in curr)= $\frac{P(leak\ in\ i)(e^{-\alpha(d[i,leak1]-1})(e^{-\alpha(d[i,leak2]-1})}{\sum_{all\ cells\ j} P(leak\ in\ j)(1-e^{-\alpha(d[j,leak]-1})}$, P3(leak in i | heard no beep in curr)= $\frac{P(leak\ in\ i)(1-e^{-\alpha(d[i,leak]-1}(e^{-\alpha(d[i,leak2]-1}))}{\sum_{all\ cells\ j} P(leak\ in\ j)(1-e^{-\alpha(d[j,leak]-1})}$. And the formula P1(leak in cell i | no leak in curr)=$\frac{P(leak\ in\ i)}{1-P(leak\ in\ curr)}$ is used to update abote each move. The purpose of the probabilistic update is to estimate the likelihood of each cell in the grid being a missing point. When the bot moves to a new cell, the probability that the cell contains a leak becomes zero. Adjust the probabilities for other cells to ensure that the overall probability for the entire ship remains 1.

3. Performance between the bots:

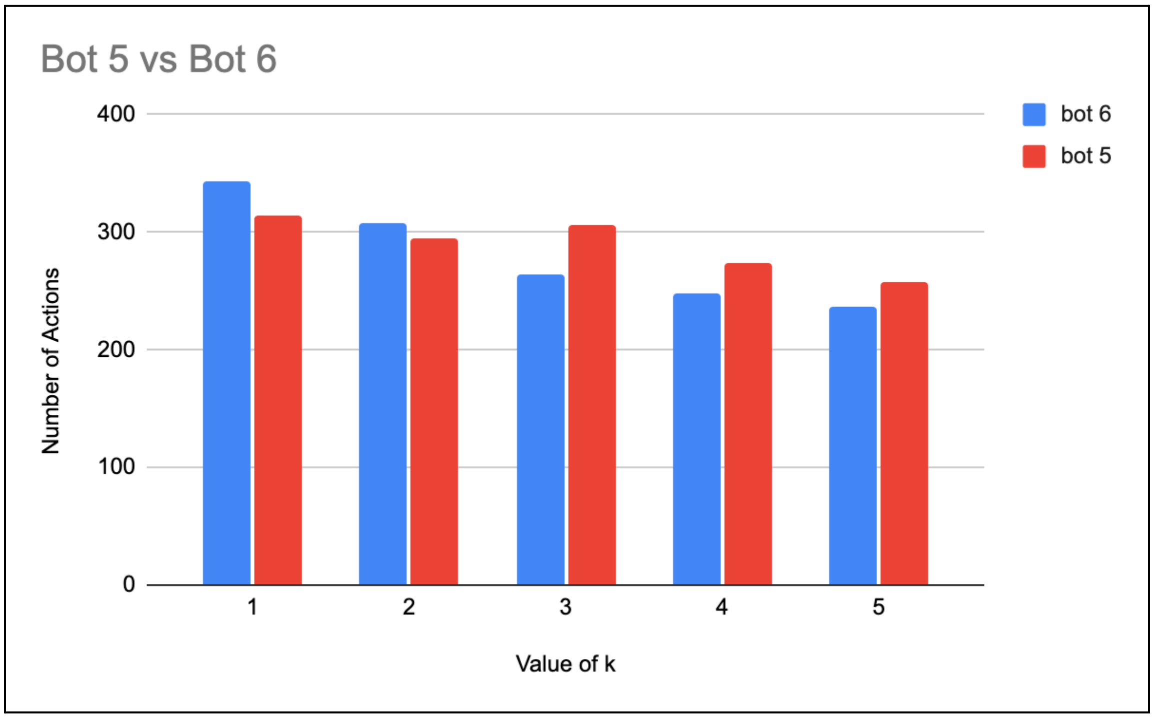
According to the graph, bot 2 is better than bot 1:



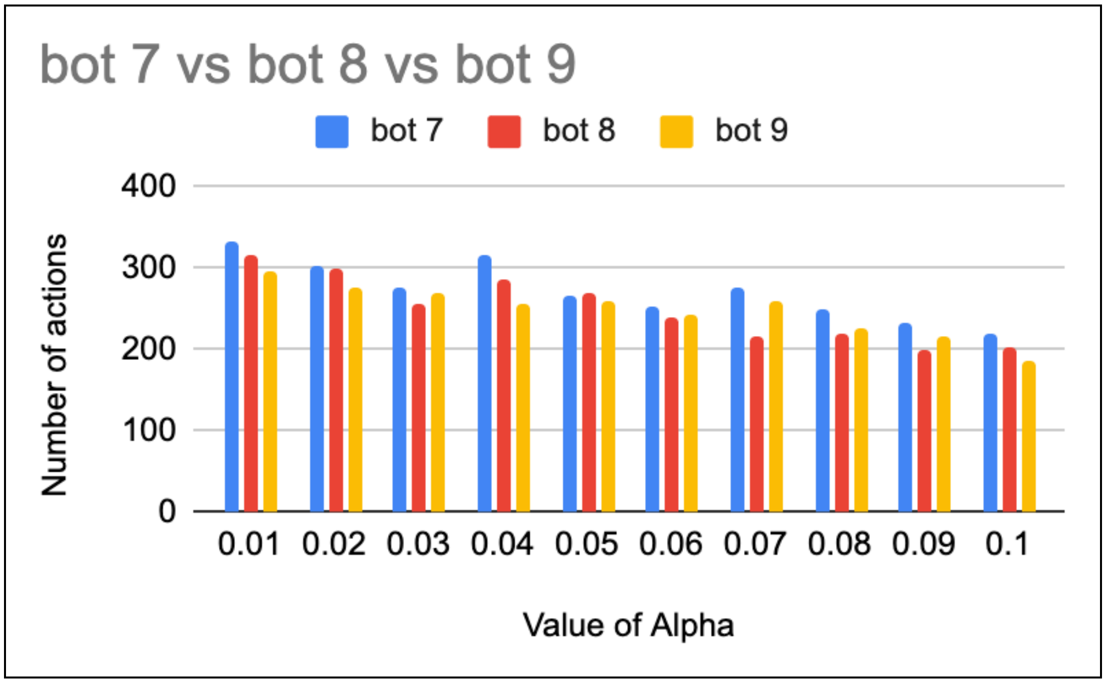
According to the graph, bot 4 is better than bot 3.



According to the graph, the performance of bot 5 and 6 is similar:



According to the graph, bot 8 and 9 are better than bot 7:



4. The Ideal bot:

The ideal bot should have detailed knowledge of the ship's layout in order to navigate effectively. It uses efficient algorithms (such as BFS, A*) to calculate the shortest, optimal path to the target cell for movement. That is, various constraints such as obstacles and movement restrictions are considered simultaneously.

Dynamic programming should also be considered to dynamically adjust strategies according to changes in the environment and goals to adapt to environmental changes and sensing updates. If new information emerges or a leak is discovered, the bot should be able to recalibrate its path immediately. Use Markov decision-making to evaluate the potential outcomes and risks of different actions.

The ideal bot should have machine learning capabilities, allowing the bot to learn and optimize its behavior through experience to improve decision-making efficiency. Learn about previously discovered leaks to inform ongoing search strategies. Make decisions quickly based on current data and flexibly adjust as new information emerges.

It has better decision-making capabilities, taking into account the leakage probability of each part and the historical accuracy of the sensing in those areas, and can detect errors in time. Use Bayesian networks to estimate vulnerability probabilities under different scenarios. By updating the probability distribution in the network, the bot can use new observations to revise its understanding of the environment.

For example, if it consistently fails to find a leak in a certain part, it might give higher priority to other parts. Consider the probability of leakage in each section and the historical accuracy of the sensors in those areas.

— TO RUN —

```
python3 botX.py A B
```

X: the number of the bot
A: ship size (in this case 50)
B: alpha value

Tip: make sure pygame is downloaded in order to run

— WHO DID WHAT —

Zihao Zheng: bot1,bot3,bot4,bot5,bot7,bot8, and bot9 code writing + debugging.

Chongen Zheng: bot 2, bot 6, write up

Yiming Huang: bot 2, algorithm brainstorming