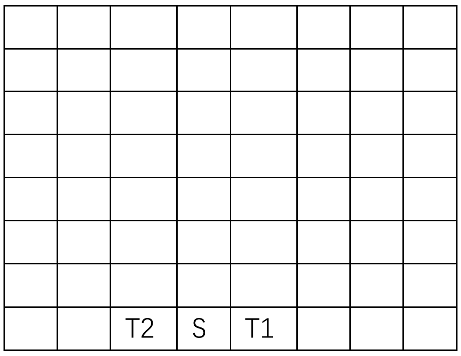
CS 520 Final: Question 1 - Decision Making

Name: Xinghao Wang
NetId: xw354

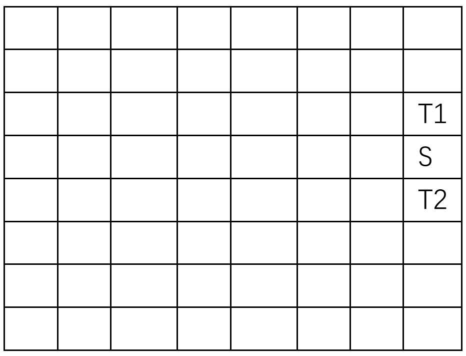
(1) The goal is to pin the sheep in the upper left corner of the field and the rule is when the sheep has a neighbor of dog, it can't move to that direction. So, I notice that if the dogs could somehow stick to the sheep, there would be at most 2 directions for the sheep to move. Going further, if the dogs stick to the sheep in the right side and lower side, the sheep would only have two choices to go, up and left. Eventually, the sheep would end up in the upper left corner and the game is over.

The next step is how to stick the dogs to the sheep. My method is to search the path between them. I let one dog search the right neighbor of the sheep and the other dog search the neighbor below the sheep. In order to minimize the number of rounds, I need to get the shortest path, so I choose BFS search algorithm. During each round, first I compute the two paths and then I command the dog to move one step further along the path. And if the dog reaches the target, then hold the position till the target is changed. Notice that I have two BFS algorithm in my code, because I want the dog go horizontally first whose destination is below the sheep and I want the other dog go vertically first. By doing so, I can avoid some strange endless loops.

One problem I encountered was that if the sheep is in the right and lower boundary of the map, a dog would lose the target because of the illegal point out of the map. Therefore, I changed the target of the dog under certain circumstances.



the sheep is in the middle of lower border



the sheep is in the middle of right border

T2							
S	T1						

the sheep is in the lower left corner

					T1		S
							T2

the sheep is in the upper right corner

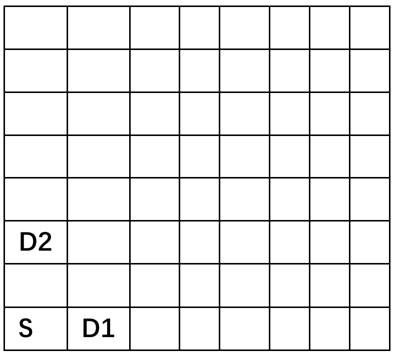
							T1
						T2	S

the sheep is in the lower right corner

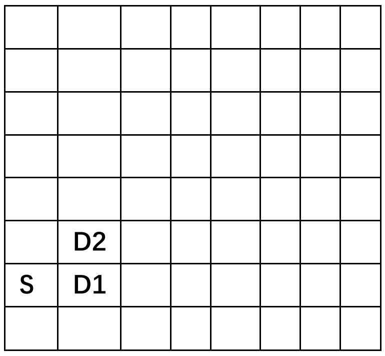
When the sheep is in the middle of the border, I want to squeeze it out, and when the sheep is in the corner, I want to give it some space to get out of the corner. In this way, the problem is solved.

(2) The initial state is sheep (5, 1), dog1 (7, 4), dog2 (1, 8). After running 10000 times, the average rounds are 18.9868.

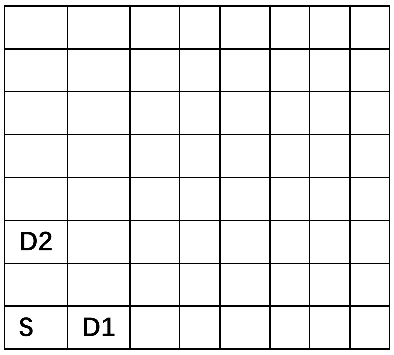
(3) The worst possible initial state is that the sheep is in (8, 8) and two dogs are in (1, 1) and (1, 2). The reason is that we can think of this process as three stages. The first stage is from the beginning to the first touch of the sheep. The maximal rounds it takes are 16 because the Manhattan distance between any two points in the map is less than twice the length of the map. The second stage is from the first touch of one dog to the first touch of the other dog. The maximal rounds in stage2 is unclear because there are some weird situations that may last forever. For example, assume at a certain round, the state is as follows.



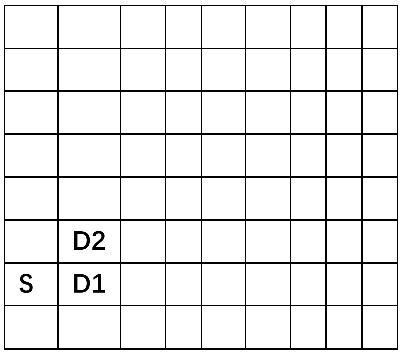
Round x



Round x+1



Round x+2

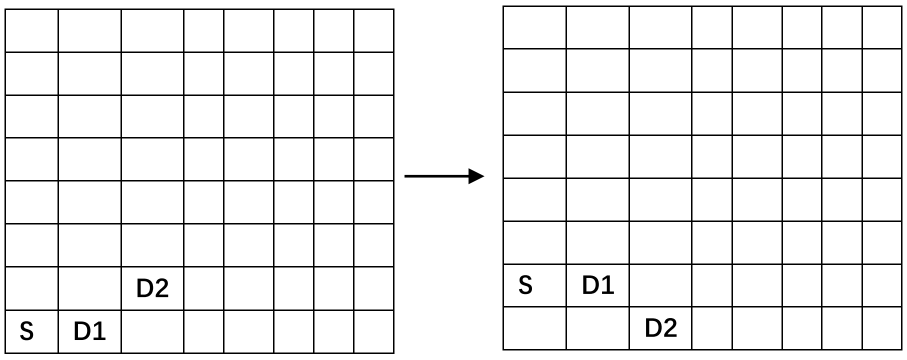


Round x+4

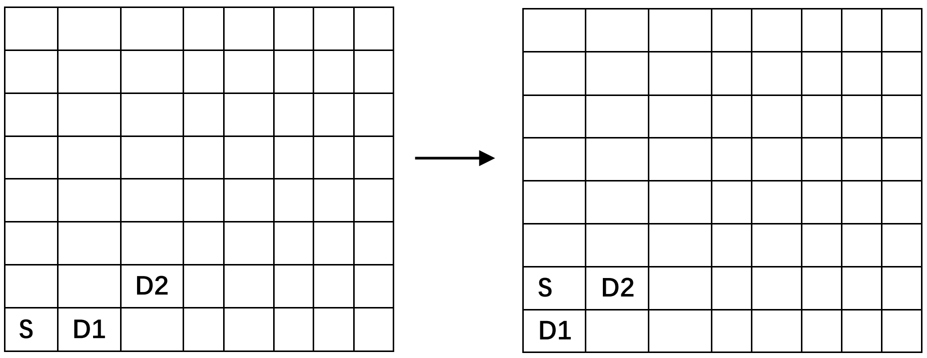
As the graphs shown, if the sheep is always jump between (8, 1) and (7, 1), it will be an endless loop. However, the expectation of rounds getting out of this loop are 4 because when the sheep is in (7, 1), the expectation of times it finally goes up is 2. The third stage is from the end of stage2 to the end of the game. After stage2, the two dogs have been stuck to the sheep. So, the sheep would move straight to the upper left corner with the maximal rounds 16. Therefore, in order to maximize the total rounds, what I can do is make sure the rounds of stage1 and the rounds of stage3 are as big as possible at the same time.

(4) Let one dog in (8, 8) and the other in (8, 7). The reason is the same as question (3), but now let the rounds of stage1 and stage3 as little as possible. With that in mind, we can place the two dogs in the lower right corner.

(5) Yes, because my algorithm is based on the current location of the sheep and I did not consider the influence that the sheep's next step has on my dogs. If I can figure out how to use that information, the model must be more efficient. The possible example is as follows.



Assume the graph on the left is the current state and according to my algorithm, the next state should be the graph on the right. Notice that there is still only one dog attached with the sheep. However, since we know that the sheep on the current state has only one direction to move, we can take advantage of this prediction change our move as follows.



Notice that both of the two dogs are attached with the sheep. Therefore, better strategies must exist.