Question 1

Some prior thoughts before going ~~dive~~ into the Question 1:

We are giving a maze represented by '0' and '1', in which '1' represents blocks and '0' represents blanks. At the first glance of this problem, it reminds me of project 1 (Maze Runner) for we are given a maze with '0' and '1'. However, there are differences:

1) Firstly, ~~you~~ we are not given a start point. Instead, ~~you~~ we are dropped into a maze randomly, which means this problem is related to probability.

2) Secondly, ~~you~~ we are not receiving feedbacks, which means we can not even ~~decide~~ judge if we are running against a 'wall'. Unless ~~you~~ we figure out where we are, we can not search ~~as~~ ~~we~~ the way we did in Project 1 (Maze Runner)

At the office hour, Professor gave out an idea to this problem: we can regard the move of Maze solving Bot as the flow~~ing~~ of possibility. This suggestion recalls the memory of what we did in Project 3 (Search And Destory), in which the flow of possibility is involved. In this problem, because we are not given prior info, the Maze Solving Bot is of possibility $P = \frac{1}{\#\text{ all blank blocks}}$ in each blank block. And when it moves, for example, to left, all the possibility moves ~~as~~ left. Below is a small eg illustrating this idea.

$\frac{1}{8}$	$\frac{1}{8}$	$\frac{1}{8}$
$\frac{1}{8}$	///	$\frac{1}{8}$
$\frac{1}{8}$	$\frac{1}{8}$	$\frac{1}{8}$

move left →

$\frac{1}{4}$	$\frac{1}{8}$	0
$\frac{1}{8}$	///	$\frac{1}{8}$
$\frac{1}{4}$	$\frac{1}{8}$	0

Above, we can see the detailed way of computing flow of possibility, when the Maze Solving Bot moves to left, we move the possibility column by column to left. If the move is blocked, then the blocked move will remain its initial possibility. With this idea, we can also move the possibility right, up and down by having our Maze Solving Bot move in the corresponding direction. And after a sequence of movement, the possibility gradually gathers together: some blocks have high possibility and some blocks have 0 as possibility.

Now we're going into Question 1.

(a) As we mentioned in prior part, we load in the 'Maze.txt', count all the 0's, and we get the sum: 1239.

$\therefore P(\text{Bot at } G_1) = P(\text{Bot at every blank block}) = \frac{1}{1239}$

(b) Question (b) requires a short sequence of moves that with probability of at least $\frac{1}{2}$ leads our Maze Solving Bot to G_1. We devide this Question (b) into two parts on which we implement our Algorithm: The two parts are as follows:

① Through a sequence of moves, we get at least one block with probability of at least $\frac{1}{2}$.

② From this (these) point(s), we find our way to G_1 (BFS)

⇒① The ~~point~~ important point to ① is how to find a rational sequence, or to say, short sequence. ~~This~~ More specifically, we need to decide which direction of move leads to a short sequence. Here we use '~~*~~Greedy strategy' to solve this problem (detail shown afterwards)

In our Algorithm, we define the following function:

(1) up() / down() / right() / left(): respectively represent Bot's moving and compute the ~~pos~~ probability distribution

(2) find_maxelem(): finds the max probability in maze to determine whether to end the process.

(3). find_dis(): returns the max distance among all distances.

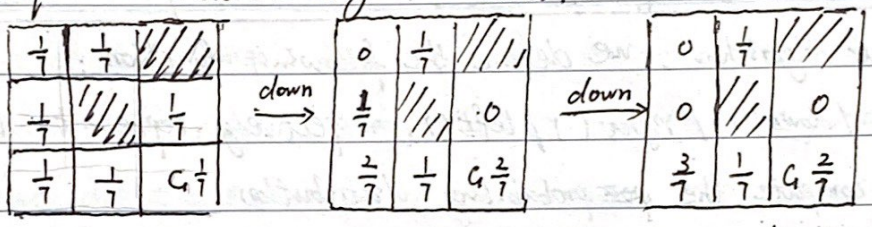
* definition of distance: if one cell [x_i, y_i] has probability that is not equal to 0, Then distance = abs(x_i - x_G) + abs(y_i - y_G)

In our main function, we have loops in following way:

we take down() ~~up~~ right() ~~left~~ up() left() orderly, If down() shrinks the max distance, then we take down() as our move. If down() doesn't shrink max distance, then we take right() to see if it shrinks the max distance. The use of max distance shows the 'Greedy strategy', ~~for it helps it~~ for in each move we choose, we ~~gradually~~ gather together the probability towards 'G'.

Also, the order of 'down() - right() - up() - left' ~~is~~ comes from 'Greedy strategy' Because point 'G' is ~~the~~ at the bottom right. So if we want to quickly gather the probability toward 'G'. we should do more 'down()' and 'right()'

There is also a little trick worth mentioning. That is, ~~when~~ when we find a 'move' doesn't shrink the max distance, we still choose it with a relatively low possibility. Let's see a small example that helps with understanding this idea.



As we can see, from pic 2 to pic 3, taking 'down()' doesn't change the max distance, but it still helps gather the probability towards G. So this trick is useful in our Algo.

[Code for this part is named as Question1_abc. and the corresponding log_abc.txt can be found in Question1's Folder]

Our Algo takes around 1000 steps to ~~make~~ get a cell with probability bigger than 0.5. And at this point, there are usually around 10 cells contains probability.

(c) ~~Question~~ Question (c) contains the same idea as Question (b). However, we need to gather all probability to one ~~point~~ cell, which is much more difficult.

As we can see in log_abc.txt, ~~we~~ when we have a cell contains probability $^{>0.5}$, there are only around ten cells that ~~contains~~ hold probability.

It takes huge steps to gather these remaining probability. The max probability may remain in a long time, and suddenly jumps to a high probability.

During the test, my Algo takes more steps than I expect.

(d) ① Compared with Question (b) & (c), Question (d) gives the observation of totally blocks around our Maze Solving Bot. This reminds me of the 'moving target' part in Project 3. Under the light of Project 3, Question (d) actually is made up of two models: Transition model and Observation model.

Based on this idea, I define following functions in our Algo:

(1) left(): represents Bot's moving left and compute the probability after moving, which is the same as left() in Question (b). It corresponds to Transition Model.

(2). observation_and_redistribution():

When given feedback of surrounding blocks, we use this to update the probability in Maze. This process is easy to implement; for example, if the feedback is '5 blocks around Bot', then we go through the whole maze, & set the probability in the cell whose surrounding blocks don't equal to five to '0'.

So the main process in part ① of Question (d) is as follows:

```
localization.observation_and_redistribution(5)
localization.left()
localization.observation_and_redistribution(5)
localization.left()
localization.observation_and_redistribution(5)
```

[code for this part is named as Question1-d.py and the corresponding output is named as log_d.txt]

② Based on our analysis in part ①, part ② is a generalization of part ①. we can define down()/right()/up()/left() in the same form as left() in part ①, and the same observation_and_redistribution().

given observations $\{Y_0\ Y_1 \cdots Y_n\}$

given actions $\{A_0, A_1, \cdots A_{n-1}\}$

our main process is as follows:

```
init_possibility
for i = 1 to n-1:
    ~~down() or right() or up() or left()~~ ~~(corresponds to A_i)~~
    observation_and_redistribution(Y_i)
    down()/right()/up()/left() (corresponds to A_i)
observation_and_redistribution(Y_n)
traverse the whole maze, find the cell with largest probability.
```