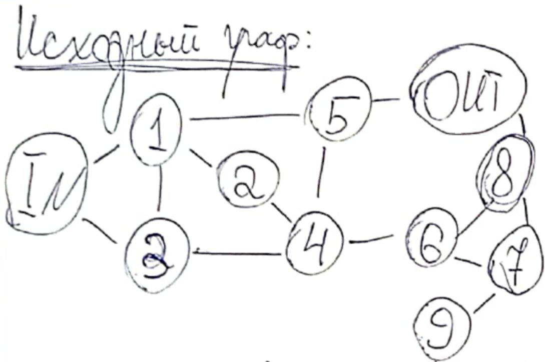
Исходный граф:



Краткое описание функций:

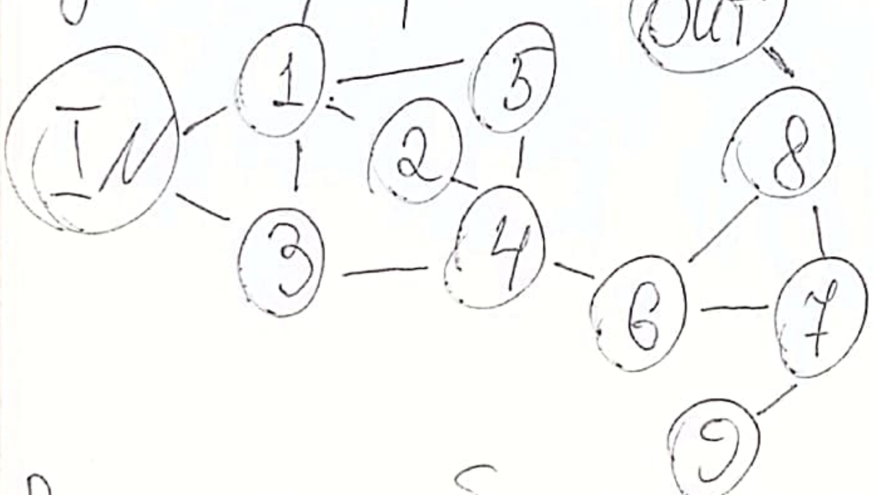
- BFS - основной алгорит, совершающий итерации и ~~учитывающ~~ учитывающий условие останова.
- make_a_move - функция, совершающая шаг по всем путям (аналог цикла)
- add_ways - функция, совершающая шаг по определённому пути.

Итерация:	0-ая	1-ая	2-ая	3-я
Список ways:	((IN))	((3 IN) (1 IN))	((5 1 IN) (2 1 IN) (4 3 IN))	((6 4 3 IN) (OUT 5 1 IN))
Список visited:	(IN)	(3 1 IN)	(4 2 5 3 1 IN)	(OUT 6 4 2 5 3 1 IN)

Описание: На 0-ой итерации алгоритм сделает шаг, так как нет пути от In до Out, в результате чего будет два пути, т.к. из вершины In можно попасть только в две вершины - 3 и 1. На 1-ой итерации по той же причине будет совершаться шаг. Рассмотрим его подробнее: первый переданный путь - (3 IN), шагнуть можно только в комнату 4, т.к. в комнатах (вершинах) In и 1 алгоритм уже был. Второй переданный путь - (1 IN): Из вершины 1 мы можем пойти в такие вершины, как IN, 3, 2 и 5. Вот только в двух вершинах алгоритм ещё не был - 2 и 5, соответственно, шагаем по ним и получаем пути (5 1 IN) и (2 1 IN).

2-ая итерация: Первый переданный путь - (5 1 IN) в данном случае алгоритм может пойти только в вершину OUT, так как в остальных он уже был. Получаем новый путь (OUT 5 1 IN). Второй переданный путь - (2 1 IN). Данный путь является тупиковым, потому что в вершинах 1 и 4 алгоритм уже был, поэтому алгоритм "забывает", то есть удаляет, этот путь. И последний путь - (43 IN). Алгоритм шагнёт в вершину 6 и получится новый путь - (6 4 3 IN). На третьей итерации алгоритм завершит свою работу, т.к. ф-ция check_end выдаст искомый путь - (IN 1 5 OUT)

Изменим граф:



Рассмотрим работу алгоритма по итерациям

Итерация	visited	ways
0	(IN)	((IN))
1	(1 3 IN)	((1 IN) (3 IN))
2	(4 2 5 1 3 IN)	((4 3 IN) (2 1 IN) (5 1 IN))
3	(6 4 2 5 1 3 IN)	((6 4 3 IN))
4	(7 8 6 4 2 5 1 3 IN)	((7 6 4 3 IN) (8 6 4 3 IN))
5	(Out 9 7 8 6 4 2 5 1 3 IN)	((Out 8 6 4 3 IN) (9 7 6 4 3 IN))