Большая программа - задание номер 12 (Поиск в лабиринте)
Автор: Маннанов Айрат, 325 группа. 2023 год

Цель работы
Написать программу для поиска пути от одной вершины до другой с учётом того, что через любую вершину можно пройти только один раз, то есть в выходном пути не должно быть две одинаковой вершины. Дополнительным условием является запрет на прохождение внутри алгоритма через любую вершину дважды.

Алгоритм для поиска пути
В качестве алгоритма для поиска пути был выбран алгоритм BFS (breadth-first search) - поиск в ширину, одновременно проходящий по всем возможным путям до тех пор, пока один из путей не попадёт в вершину назначения. Данный алгоритм обладает такими достоинствами, как:

- через любую вершину алгоритм проходит не более одного раза (выполнение дополнительного условия);
- алгоритм находит кратчайший путь между вершинами.

Представление данных
Данные о графе подаются через результат функции get_graph. Граф представлен в виде списка, содержащий пары формата (V1, V2), где V1 и V2 - вершины, а наличие такой пары утверждает, что между данными вершинами существует путь. Данные о графе преобразуются в результате работы функции с накапливающим параметром get_doors_to. После данного преобразования граф представляется в формате списка, содержащего списки (V1 V11 V12 … V1n), где V1 - вершина, а V11, V12, … V1N - вершины, смежные вершине V1.

Принцип работы алгоритма
Рассмотрим вспомогательные функции:

1. check_not_in - функция-предикат, выдающее значение T, если атом room не содержится в списке visited и nil - в противном случае;
2. get_rooms - функция, выдающая список из вершин, смежные вершине room;
3. check_end - функция-предикат, проверяющая есть ли в списке формата: ((V1n … V11) … (Vmn … Vm1)) вершина end среди первых

элементов подсписков, то есть проходя через элементы верхнего списка, функция проверяет на равенство вершины V1n и end, …, Vmn и end, если равенство выполняется, то выдает данный подсписок в перевернутом виде. Рассмотрим основные функции:

1. BFS - функция среди параметров имеет to - атом (вершина), в которую нужно найти путь, doors - граф уже в преобразованном формате, visited - список, содержащий атомы (вершины, в которых алгоритм уже был) и ways - список из списков, являющимися путями из исходной вершины. Функция проверяет с помощью функции check_end не имеется ли среди путей уже искомый, если это так, то выдает его в формате (from V1 … Vn .. to), где from - исходная вершина (источник), a to - конечная вершина. Иначе алгоритм вызывает функцию make_a_move, тем самым делая шаг по каждому из путей дальше, - совершает одну итерацию, и вызывает функцию BFS с измененными параметрами visited и ways.
2. make_a_move - функция с накапливающим параметром (в качестве параметра подаётся пустой список, являющимся представлением новых путей). Функция рекурсивно проходит по списку ways и для каждого элемента этого списка вызывает функцию add_ways, тем самым добавляя в накапливающий параметр новые пути. В конце работы функция выдает пару в формате (visited new_ways), где visited - измененный список, содержащий пройденные вершины, new_ways - список из обновленных путей.
3. add_ways - функция, добавляющая в накапливающий параметр new_ways новые пути. В качестве параметров подаются: visited - список, содержащий пройденные вершины; way - путь; rooms - список из вершины, в которые можно попасть из вершины, находящейся первой в списке way. Функция рекурсивно пробегается по элементам списка rooms, если в списке visited элемент не находится, то добавляем его в списки visited и way, который, в свою очередь, добавляем в список new_ways, иначе ничего не добавляем. Достигнув конца списка rooms, функция выдает список формата (visited new_ways).
4. get_way_from_to - функция-обёртка, принимающая в качестве параметров: from - начальная вершина (источник); to - конечная вершина и граф в исходном представлении. Вызывает функцию BFS с преобразованными данными.

Подробней работа программы описывается в файле example. Первое изображение - описание того, что происходит на каждой итерации алгоритма, а на второй дополнительный пример, если усложнить граф - сделать путь между вершинами IN и OUT длиннее.

В рамках дополнительного задания была реализована функция get_way, получающая в качестве параметров graph - исходное представление графа и список rooms в формате (from V1 V2 … Vn to). Функция выдает путь от вершины from до вершины to, при условии, что в данном пути должны содержаться остальные вершины из списка rooms. Данная функция использует основную - BFS, как вспомогательную функцию, за счёт которой выдаётся путь между вершинами from и V1, V1 и V2, …, Vn и to, соответственно. После чего данные пути последовательно соединяются - получается искомый путь, который выдаёт функция.