Lab08-Graph Exploration

CS214-Algorithm and Complexity, Xiaofeng Gao & Lei Wang, Spring 2021.

∗ If there is any problem, please contact TA Yihao Xie.
∗ Name:RenyangGuan Student ID:519021911058 Email: guanrenyang@sjtu.edu.cn

1. Given an undirected graph $G = (V, E)$. Prove the following propositions.

   (a) Let e be a maximum-weight edge on some cycle of connected graph $G = (V, E)$. Then there is a minimum spanning tree of G that does not include e. Moreover, there is no minimum spanning tree of G that includes e if e is the unique maximum-weight edge on the cycle.

   (b) Let T and T' are two different minimum spanning trees of G. Then T' can be obtained from T by repeatly substitute one edge in $T\backslash T'$ by one edge in $T'\backslash T$ and meanwhile the result after each subsitution is still a minimum spanning tree.

   Solution. (a) The process of proof should be divided into two parts. One is to prove *there is a minimum spanning tree of G that does not include e.* and the other is to prove *there is no minimum spanning tree of G that includes e if e is the unique maximum-weight edge on the cycle.*

   (1) Let the cycle containing edge e be denoted as $\{e, e_1, e_2, \cdots, e_k\}$. The proof is done by **contradiction**. Suppose none of the minimum spanning tree contains edge e. For each particular minimum spinning tree T_i, there exists an edge $e_i \in \{e, e_1, e_2, \cdots, e_k\} - \{e\}$, where $w(e_i) \leq w(e)$. The operation which remove e and insert e_i will reduce the total weight of the tree, which is a contradiction.

   (2) Let the two vertexes connected by e are v_i and v_j. The proof is also done by **contradiction**. Suppose that there is one minimum spinning tree T_i containing edge e. The operation of deleting e will divide the tree into two strong connected components, which are denoted as SCC_i, SCC_j containing v_i and V_j respectively. Since there must be another edge e' which connects SCC_i and SCC_j (otherwise the degree of v_i or v_j is 1), adding e' into the graph will connect the two strong connected components without adding cycle, which will reduce the total weight of T_i. This is a contradiction of the property of minimum spinning tree.

   (b) The proof is done by **contradiction**.
   Suppose that one particular operation makes T'' not a minimum spinning tree, which substitutes e in $T\backslash T'$ by e' in $T'\backslash T$ and generate T''. It means $w(e') < w(e)$. After deleting e, T is divided into two strong connected components SCC_i and SCC_j and both of them are still trees. As a result, e and e' are involved in the same cycle in the raw graph, otherwise SCC_i or SCC_j will be connected by e or e' in T.
   According to the conclusion in the previous question, the total weight of T will be reduced if e is substituted by e'. so T is actually not a minimum spinning tree. This is a contradiction.

   □

2. Let $G = (V, E)$ be a connected, undirected graph. Give an $O(|V| + |E|)$-time algorithm to compute a path in G that traverses each edge in E exactly once in each direction. Describe how you can find your way out of a maze if you are given enough coins to apply your algorighm.

每一条边都是双向的。

Solution. Assuming that some vertices in V is reachable while the others are not, the connected and undirected graph $G = (V, E)$ could denote the maze. Given a starting vertex and a finishing vertex, the algorithm is shown below

Algorithm 1: $EXPLORE(G, v, V')$

Input: A graph $G = (V, E)$, starting vertex v, an array V' initially set to $\emptyset$
Output: A path of vertices V'

```
1 V' ← V' ∪ {v};
2 if v is v_finish then
3     return V'
4 for ∀ edge(u, v) ∈ E do
5     if u ∉ V' and u is reachable then
6         EXPLORE(G, u, V');
7         V' ← V' ∪ {v};
```

单纯的DFS能访问到的边是一棵树，但是DFS能够访问V中的所有的点。如果要访问全图，必须要对每一个点的所有边进行检验。

此算法错误，只能找到开始结点Vstart到出口结点Vend的路径，不是走迷宫的方式。走迷宫的时候发现走不通了就必须要回头。

Time complexity analysis:
Over the course of the algorithm, each node will be explored at most once and each edge will be traversed at most twice.
As for the vertexes, if the point is not on the path, it will be added V' at the end of the deeper recursion to represent that the traveler needs to pass through the node again when he returns.
As for the edges, each edge is explored at most twice. Once is when it is firstly explored and the other is when the edge is not on the path from v_{start} to v_{finish}.
As a result, the time complexity is

$$Time\ complexity = \ O(|V| + 2|E|) = O(|V| + |E|)$$

□

3. Consider the maze shown in Figure **??**. The black blocks in the figure are blocks that can not be passed through. Suppose the block are explored in the order of right, down, left and up. That is, to go to the next block from (X, Y), we always explore $(X, Y + 1)$ first, and then $(X + 1, Y)$,$(X, Y - 1)$ and$(X - 1, Y)$ at last. Answer the following subquestions:

   (a) Give the sequence of the blocks explored by using DFS to find a path from the "start" to the "finish".

   (b) Give the sequence of the blocks explored by using BFS to find the shortest path from the "start" to the "finish".

   (c) Consider a maze with a larger size. Discuss which of BFS and DFS will be used to find one path and which will be used to find the shortest path from the start block to the finish block.

Solution.

(a) The sequence of blocks explored by DFS is

$$(A, A), (B, A), (B, B), (B, C), (C, C), (A, C), (A, D), (A, E), (B, E), (C, E), (D, E), (D, D)$$

while the path is

$$(A, A), (B, A), (B, B), (B, C), (A, C), (A, D), (A, E), (B, E), (C, E), (D, E), (D, D)$$

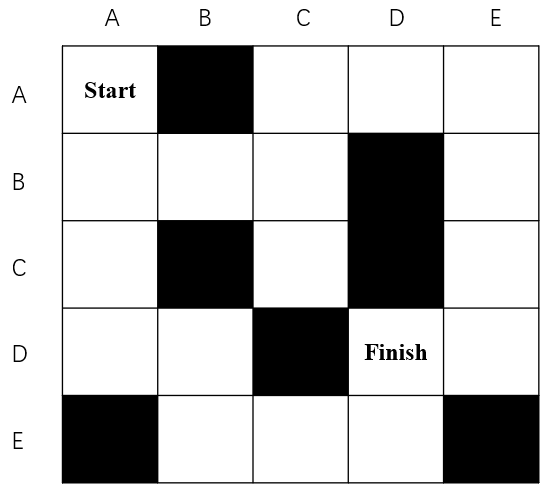

图 1: The blocks in the maze.

(b) The sequence of blocks explored by DFS is

$$(A,A),(B,A),(B,B),(C,A),(B,C),(D,A),(C,C),(D,B),(A,D),(E,B),(A,E),(E,C),(B,E),$$

while the shortest path is

$$(A,A),(B,A),(C,A),(D,A),(D,B),(E,B),(E,C),(E,D),(D,D)$$

(c) **BFS is more suitable for searching the shortest path**. DFS needs to traverse every possible path from start to end while BFS doesn't. Although BFS searches almost every node in the graph, there must be some node left.
DFS is more suitable for searching an existing path. When the maze is of a huge size, there is a high possibility that the path exist in the former half of the maze and all the paths are of the similar length. As a result, the DFS will only traverse three forth of the vertices while the BFS will traverse all the vertices.

□

4. Given a directed graph G, whose vertices and edges information are introduced in data file "SCC.in". Please find its number of Strongly Connected Components with respect to the following subquestions.

(a) Read the code and explanations of the provided C/C++ source code "SCC.cpp", and try to complete this implementation.

(b) Visualize the above selected Strongly Connected Components for this graph G. Use the *Gephi* or other software you preferred to draw the graph. (If you feel that the data provided in "SCC.in" is not beautiful, you can also generate your own data with more vertices and edges than G and draw an additional graph. Notice that results of your visualization will be taken into the consideration of Best Lab.)

Solution. The graph of strong connected components is shown in Fig. **??**. It merges all the node belonging to the same strong connected component in a node and uses the size of node to denotes the number of nodes the strong connected component containing.

□

Remark: Please include your .pdf, .tex, .cpp files for uploading with standard file names.

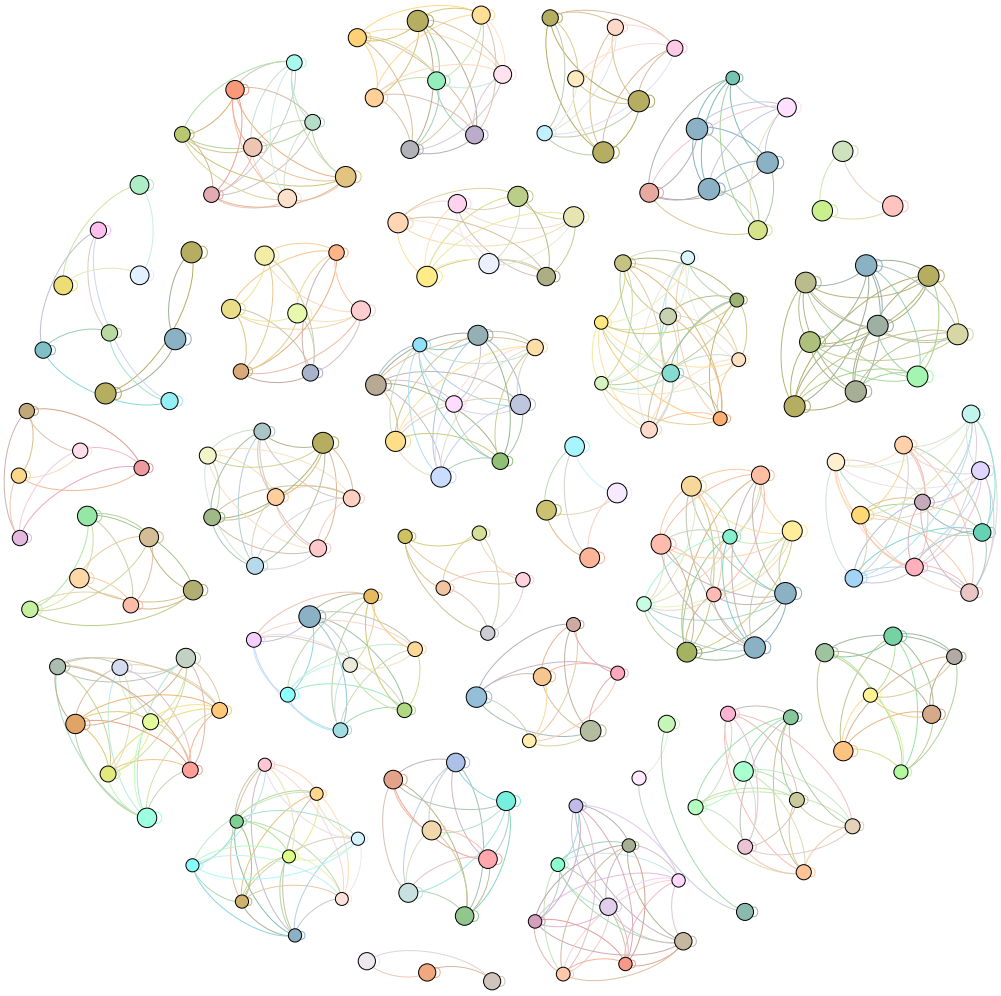

图 2: The graph of strong connected components.