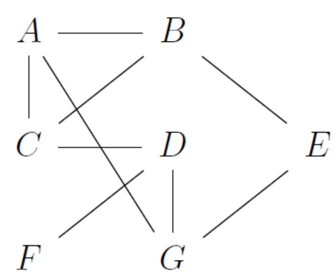
Homework4 for EECS 340

Yu Mi,yxm319

March 26, 2018

1 Warm-up: BFS

Given the graph:



a Draw the shortest path tree found by a BFS.

Answer : The shortest path tree is shown as Fig.1

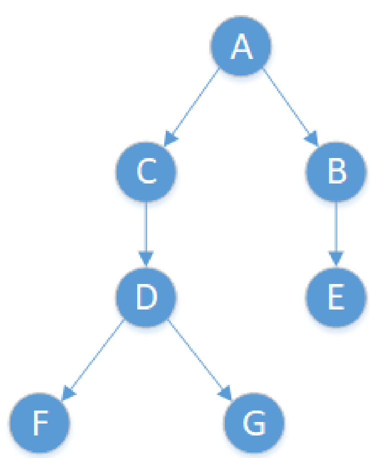


Figure 1: Shortest path tree

b Annotate the tree in part(a) with the order vertives are visited in your BFS.

Answer : The annotated graph is shown as Fig.2

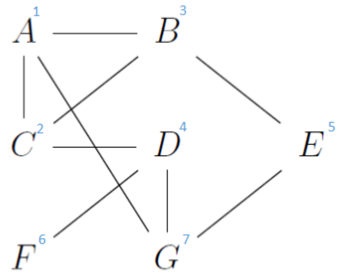


Figure 2: Annotated graph

c List the cross-edges of the BFS.

Answer : The cross edges of the BFS could be described as follows:

$$\{(B, C)\}$$

2 Warm-up: Topological Sorting

$$\begin{tikzcd}
(0,0) & (0,1) \arrow[l] & (0,2) \arrow[l] \\
(1,0) \arrow[u] & (1,1) \arrow[l] \arrow[u] & (1,2) \arrow[l] \arrow[u] \\
(2,0) \arrow[u] & (2,1) \arrow[l] \arrow[u] & (2,2) \arrow[l] \arrow[u]
\end{tikzcd}$$

a Find four distinct topological ordering of the nodes in the following directed acyclic graph, presented here as a 3×3 grid.

Answer : The topological ordering of the nodes could be:

1. $\{(2,2), (2,1), (1,2), (2,0), (1,1), (0,2), (1,0), (0,1), (0,0)\}$
2. $\{(2,2), (1,2), (2,1), (2,0), (1,1), (0,2), (1,0), (0,1), (0,0)\}$
3. $\{(2,2), (2,1), (1,2), (2,0), (0,2), (1,1), (1,0), (0,1), (0,0)\}$
4. $\{(2,2), (2,1), (2,0), (1,2), (0,2), (1,1), (1,0), (0,1), (0,0)\}$

b Describe how to generalize your orderings to the case of a graph with the same connectivity pattern on an $n \times n$ grid.

Answer : As is shown in Fig.3, the nodes can be classified into 5 different classes according to their depth from the starting node $(2, 2)$. As we can see from the graph, the two indexes of the nodes is always larger or equal than 0, and the deeper the node is, the less the sum of two indexes would be. This observation also corresponds to the classification we made as above. Since the nodes in the same class have the same depth from the starting node, we can arrange them randomly (in class) in the topological order. However,

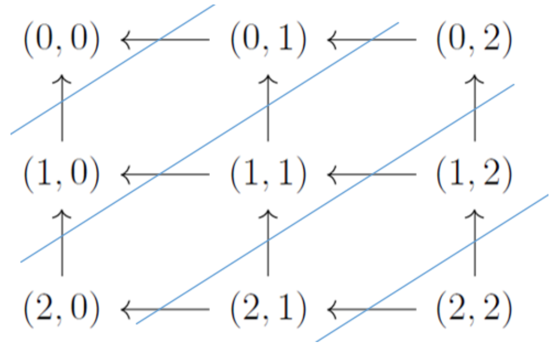


Figure 3: Classify the nodes

as to the ordering between different classes, we can only follow the order from the starting node to the end point.

To describe the topological order, we define **class(n)** as the nodes which have their two indexes' sum is n. For example, $class(n) = \{(n, 0), (n-1, 1), ..., (1, n-1), (0, n)\}$. Thus, the topological order for an $n \times n$ grid would be: $\{class(n), class(n-1), ..., class(0)\}$. One thing to node here is the $class(n)$ does not necessarily mean a ordered set, so that it could generate $n!$ possible orderings. As to the $n \times n$ grid, the number of possible ordering is $\prod_{i=0}^{n} n!$.

3 A Unicycle Problem

Prove that a cycle exists in an undirected graph if and only if a BFS of that graph has a cross-edge. Your proof may use the following facts from graph theory:

1. There exists a unique path between any two vertices of a tree.

2. Adding any edge to a tree creates a unique cycle.

Proof :

($\Longrightarrow$): Since the BFS of a graph will build a tree according to the nodes' distance from the starting node, the depths of nodes is the same as their distance from the starting node. However, if their is a cross-edge in the BFS search tree, it means there is at least one edge in the graph except the BFS search tree. According to the fact 2 provided by the question, there is a circle in the graph.

($\Longleftarrow$): Since there is a cycle exists in the undirected graph, there is at least two different paths from nodes in the cycle, say A and B. However, as the fact 1 provided, the BFS tree will only cover one unique path between A and B. And there is also another path from A to B in the graph, which means there is at least one cross-edge with one end is either A or B. So that the BFS of that graph has a cross-edge.

Based on the discussion above, the statement made by the question stands.

4 A Bicycle Problem

C-13.15 : Let G be an undirected graph with n vertices and m edges. Describe an $O(n+m)$ time algorithm to determine whether G contains at least two cycles. *Answer* : To solve this problem, we can perform DFS search on the graph and return a number of cycle detected. To determine a cycle, we need to maintain a list of visited vertices to indicate the order of DFS. Once we have reached a vertice that have already been visited, that means we detected a cycle. The algorithm is shown as follows:

Algorithm DetectCycle(V, E, n, m)
Input: n vertices V, and m edges E. The graph is represented by link list. The link list is named as *link* with size of *links*

output: The number of cycles detected.

visited $\leftarrow$ empty bool list with length of n and initial value of False
for $i \leftarrow 0$ to n **do**
  if $indegree(V[i])$=0 **then**
    $start \leftarrow i$ ▷ Let the node with no *in degree* to be the start
  end if
end for
return DFS($start$) ▷ Start from the initial node.

def DFS(x) ▷ x is the index of vertice to be search in this call stack
if $x.links = 0$ **then**
  return 0
else
  $cycles \leftarrow 0$
  for $i \leftarrow 0$ to $x.links$ **do**
    if $visited[x.link[i]]$ **then**
      return 1;
    else
      $visited[x.link[i]] \leftarrow$ True
      $cycles \leftarrow cycles$+DFS($x.link[i]$)
    end if
  end for
  return *cycles*
end if

5 Maximal Independent Set

An *independent set* I of an undirected graph $G = (V, E)$ is a subset $I \subseteq V$ such that if $u, v \in I$, then $(u, v) \notin E$. A *maximal independent set* M is an independent set which cannot have any vertices added to it without losing the independent set property. Describe an efficient algorithm to compute the maximal independent set of such G.
Answer : Since the question defines the *independent set* as those

6 Application: Procrastinator's Sort

Answer :

7 Application: Zombie Apocalypse

Answer :

8 EECS454 only

a R-17.2

Use a truth table to convert the Boolean formula $B = (a \leftrightarrow (b + c))$ into an equivalent formula in CNF. Show the truth table and the intermediate DNF formula for $\overline{B}$

b C-17.7

Consider the 2SAT version of the CNF-SAT problem, in which every clause in the given formula S has exactly two literals. Note that any clause of the form $(a+b)$ can be thought of as two implications, $(\overline{a} \rightarrow b)$ and $(\overline{b} \rightarrow a)$. Consider a graph G from S, such that each vertex in G is associated with a variable, x, in S, or its negation, $\overline{x}$. Let there be a directed edge in G from $\overline{a}$ to b for each clause equivalent to $(\overline{a} \rightarrow b)$. Show that S is not satisfiable if and only if there is a variable x such that there is a path in G from x to $\overline{x}$ and a path from $\overline{x}$ to x. Derive from this rule a polynomial-time algorithm for solving this special case of the CNF-SAT problem. What is the running time of your algorithm?

c C-17.8

Suppose an oracle has given you a magic computer, C, that when given any Boolean formula B in CNF will tell you in one step whether B is satisfiable. Show how to use C to construct an actual assignment of satisfying Boolean values to the variables in any satisfiable formula B. How many calls do you need to make to C in the worst case in order to do this?