```swift
//
//  BFS.swift
//  WhiteBoardPreparation
//
//  Created by Michael Kramskoy on 10/22/16.
//  Copyright © 2016 Connector. All rights reserved.
//

import Foundation

extension Array {

    mutating func dequeue() -> Element? {

        if self.count == 0 {
            return nil
        }
        else {
            return self.removeFirst()
        }
    }
}

class BreadthFirstSearch {

    class func copyWithBFS<T: Hashable> (_ graph: Graph<T>, _ graphCopy:
        Graph<T>) where T: Comparable {

        var copiedNodes = Dictionary<T, Node<T>>()

        for node in graph.nodes where node.visited == false { // this line
            is needed only for connected graphs

            var queue = [node]
            node.visited = true

            let nodeCopy = graphCopy.addNodeWith(value: node.value)
            copiedNodes[nodeCopy.value] = nodeCopy

            while let node = queue.dequeue() {
                for neighbour in node.neighbours {
                    if !neighbour.visited {
                        queue.append(neighbour)
                        neighbour.visited = true

                        let neighbourCopy = graphCopy.addNodeWith(value:
                            neighbour.value)
                        copiedNodes[neighbourCopy.value] = neighbourCopy
                    }

                    if let nodeCopy = copiedNodes[node.value], let
                        neighbourCopy = copiedNodes[neighbour.value]  {
                        nodeCopy.add(neighbour: neighbourCopy)
                    }
                }
            }
        }
    }
}
```