```
brute_force_algo(adj: AdjacencyList)
    (global) cycles <- (empty set)
    testCombinations(adj, [ ], adj.nodes())
    return cycles

check_cycle(adj, AdjacencyList, path: list(Node))
    if path.length = 0
        return false
    else if path.length = 1
        return path[0].linksto(path[0])

    else
        for n in (0 to path.length - 2)
            if not path[n].linksto(path[n+1])
                return false

    if path[path.length-1].linksto(path[0])
        return true
    else
        return false

test(adj: AdjacencyList, path: list(Node))
    if check_cycle(adj, path)
        cycles.add(unique(path)) # unique will be a function that ensures all cycles are
unique


combinations_helper(adj: AdjacencyList, curPath: list(Node), remainingVertices: set(Node))
    test(adj, curPath)

    for n in remainingVertices:
        newRemaining = remainingVertices
        newRemaining.add(n)
        newPath = curPath
        newPath.append(n)
        combinations_helper(adj, newPath, newRemaining)
```