CS 4102

Advanced Programming 2

Joseph Lee

Security Clearance – writeup

For this assignment, I approached with a BFS (Breadth-first search) to find all the paths between vertices. At the end, I had O((V+E)L) as runtime where V+E is BFS and L is the list of numbers of interval of each edges without duplicates. In the BFS method (public static Boolean BFS (int source, int dest, int badges)), source is the first vertex and the dest is the vertex that the first vertex is trying to find as it traverses. The BFS method will look for the vertex and if it finds the vertex in the traversal, it will return true. In this method, priority queue is created to store the starting node and mark it as it has been visited and then it will loop through the priority queue while it is not empty as it deques the front node and iterating through all its adjacent nodes. Then, the adjacent node and unvisited vertices gets enqueued and marked as visited, BFS returns false if the dest is never reached. The method was able to solve the problem efficiently as BFS was used on every badges to find out if the given badge number will make it to the end of the room. The BFS method is called on countBadges method (int countBadges (int startRoom, int endRoom, Set <Integer> range)) where the sum is being calculated. The mothod countBadges has list called trackNum which tracks the endpoints that have been processed. Then, I traverse the TreeSet called range to get an endpoint and pass that to BFS function to check if the endpoint is valid. If it is valid then I increment result by one and add the endpoint to the trackNum. Then countBadges checks if trackNum has more than one element in it. And if there is, it subtracts the first element from the second element and if the difference between them is greater than one, it pass the first element + 1 to BFS to determine if the first and last elements have a valid interval. If BFS returns true then it adds (last element – first element – 1). It subtracts one to prevent duplicates. At the end, it removes the first element from trackNum. At the end of the method, calculated sum is being returned. The runtime of the program was significantly optimized and reduced as the edges and adjacent nodes are combined into a single data structure. The time complexity of this approach is O((V+E)L) after constant for the worst case of BFS being called is removed.