CS263: Design and Analysis of Algorithm

Name: Hritik Kumar

Roll no.: 202051088

1 Problem 1

Suppose your course instructor has made n groups of students in the class which contains unqeual number students. For each group a TA has been assigned. Each TA's has checked the assignment of their respective group and sorted the assignments in their roll number. TA's has to submit the total assignments to the course instructor in sorted form. How TA's will combine the assignments so that they minimized the total number of comparison to sort the files.

Algorithm:

- Start
- First of all, we will check the first element of the 'n' groups and store the smallest element.
- We will remove the first smallest number from that group.
- Now the second element will be at the first position in that group.
- Now we will again check the first element of all the groups and store the smallest number. This smallest number is the 2^{nd} smallest element in the total no. of elements.
- We will perform this process until all the group is not empty.
- Once all the elements are over we will get the sorted array.
- End

2 Problem 2

In your Gandhinagar city, there are various locations such as Vidhan sabha, Achardham, Gandhi Asharam,...,etc. There exists a road network that connects all the locations. Due to elections in Gujarat and rallies, all the paths between any two locations act as one-way. You and your friends have decided to visit all the major locations as today is a holiday. You have to start your traveling plan from the hostel and after visiting each location you return to the hostel. The condition is you cannot follow the same path which you have already visited. Write an algorithm that gives you the efficient route to successfully execute your plan otherwise you drop your today's plan.

Approach:

Depth-first search is an algorithm for traversing or searching tree or graph data structures. The algorithm starts at the root node (selecting some arbitrary node as the root node in the case of a graph) and explores as far as possible along each branch before backtracking. So the basic idea is to start from the root or any arbitrary node and mark the node and move to the adjacent unmarked node and continue this loop until there is no unmarked adjacent node. Then backtrack and check for other unmarked nodes and traverse them. Finally, print the nodes in the path.

Algorithm:

- Create a recursive function that takes the index of the node and a visited array.
- Mark the current node as visited and print the node.
- Traverse all the adjacent and unmarked nodes and call the recursive function with the index of the adjacent node.