1. The sequence 2, 4, 1, 3, 5 has three inversions (2,1), (4,1), (4,3). The idea is similar to "merge" in merge-sort. Merge two sorted lists into one output list, but we also count the inversion.  
   • divide: size of sequence n to two lists of size n/2  
   • conquer: count recursively two lists  
   • combine: this is a trick part (to do it in linear time)

WAP to count inversion from a given array.

1. WAP to find the Longest common prefix of n strings

|  |  |
| --- | --- |
| **Sample Input** | **Sample Output** |
| 3  Algolab Algorithms Algeria | Alg |
| 4  Algolab Algorithms Algeria UIU | No common prefix |

1. Maximize your payment  
   Suppose you are temporarily working as a photoshop editor. You work 10 hours a day from 12:00 PM to 10 PM. Each of your customers sends you a task request for the next day. The task request contains the payment (p\_i), the duration (d\_i), and the deadline (dl\_i). You must complete a task within the given deadline to get the payment. Write the following greedy algorithm to maximize your payment. Note that you can move on to a new task leaving the current task partially complete, and then come back to it later.  
   Algorithm:  
   1. Sort the tasks by descending order of payment per hour.  
   2. Choose the task with the maximum payment per hour and do it in the last possible hours.  
   Make that hour occupied.  
   3. Choose the next maximum paid (payment per hour) task, find the last possible time that is  
   not occupied, do it at that time and make that hour occupied. If no such time exists, you  
   cannot do this task.  
   4. Move to the next maximum profitable task (based on payment per hour) and repeat step 3.

|  |  |
| --- | --- |
| Sample Input | Sample Output |
| |  |  |  |  | | --- | --- | --- | --- | | Customer | Last time (PM) | Time needed (hour) | Payment (Tk) | | a | 4 | 2 | 2000 | | b | 1 | 1 | 1000 | | c | 1 | 1 | 4000 | | d | 2 | 2 | 3000 | | c, a 6000 |
| |  |  |  |  | | --- | --- | --- | --- | | Customer | Last time (PM) | Time needed (hour) | Payment (Tk) | | a | 2 | 1 | 1000 | | b | 1 | 1 | 190 | | c | 2 | 1 | 270 | | d | 1 | 1 | 250 | | e | 3 | 1 | 150 | | c, a, e  1420 |

1. WAP to convert a sequence into Huffman coding and also decoding as well. You have to make sure that you write the function for encoding, display the Huffman tree, Huffman code for each alphabet, and decoding.
2. WAP to implement rooted tree with union by rank and path compression heuristics.
3. WAP to find a Minimum Spanning Tree from a graph using optimized disjoint set implementation. You can use solution of question 5 here as an optimized disjoint set.
4. WAP to find shortest path from a graph using Bellman ford algorithm.