Q 2.1)

Begin by creating a min-heap containing customer in time. The min-heap will sort based on customers’ minimum prices, so popping from the heap will remove the customer in the heap with the lowest minimum price.

Then iterating through jewelry prices, if the current item is less than the value of the current heap loot (which takes to see or pop) then iterating to the next item without popping from the heap. Otherwise, allocating that item to the customer from the loot and pop him out before moving on to the next item. (In order to assign the least jewelry prices which is at least customers’ minimum prices to many customers possible). Continue until there is no more customers or have iterated through all elements.

This would allocate items to as many customers as possible because the customers with lower minimum prices would always have more choices for items due to the fact that they would be able to choose lower price jewelries (yet higher than the minimum prices) that the other with higher minimum prices couldn’t choose and could also choose all items the higher minimum price ones capable to pick. By assigning the least jewelry prices possible to each customer would lead to minimize the number of customers who walk away with nothing since the lower minimum price ones would never steal an opportunity to get an item of those with higher minimum price ones unless they share only 1 same item (which would allocate to the lowest minimum price amongst them but the number of people walking away remains the same).

Overall time complexity is initializing the heap, for iterating through length array, totaling .

Q 2.2)

By having a counter for each element in array counting customers’ opportunity to buy each item starts with 0. And a counter counting for the index of the first element we would like to start with when iterating through an array (to know which customer we are trying to allocate an item at a time) starts with 0.

For each element of array and (iterating at the same time), starting with index (customer), iterating through array and check for every item with price in between the customer’s minimum price and budget then increasing those items’ by 1. If the current customer is not the customer then we wouldn’t increase the that is 0 even though the customer is able to afford it (because we are trying to allocate item to customer at a time and seeking to see if the customer is stealing other’s opportunity to buy items or not).

Whenever we iterate through the last element of array and , we then allocate the item with least to the customer. Reset each item’s back to 0 and increasing by 1 then repeat it again starting with index of array and until .

Doing it this way minimize the number of customers who walk away with nothing because no one would steal other’s opportunity to buy an item unless they are sharing a few opportunities which at least one customer would still walk away with that opportunity affecting nothing to the number of people walking away.

Overall time complexity is . By iterating through length array and for each element we iterate through length array which makes this cost . Since we are repeating that with ( is increased by 1 every loop from 0 to ), the times we are actually repeating it would be

Which .

Q 2.3)