Nazara Ibrat

CSCI 313

Problem 5

The longest increasing subsequence problem is a popular question known by computer scientist. To solve the problem, we need to find a subsequence of a given sequence in which the subsequence's elements are in sorted order, lowest to highest. We want to use an array based stack and the time complexity O(nlog(n)). This is not an example of dynamic programming; this is dealing which is known for doing this type of question. O(nlog(n)) runs on quicksort, mergesort, and heapsort. The length of the LIS but we are looking to for LIS (longest increasing subsequence). Using a parent array, the parent[i] has to be the predecessor of the element with index i.

The LIS is (the proper way to use the parent array):

arr[S[lastElementofS]]'

arr[parent[S[lastElementOfS]]]'

arr[parent[parent[S[lastElementOfS]]]].

We need to create the proper nodes and properly initialize it with the parent Array. Like the previous questions it's important to have the proper functions to make sure all the predeceasing requirements are meet to complete the task. Constructing the node from the current element and the index. The next function is to ignore the current element if current one is already present. Getter and setters are being use in this inserted node. There needs to be a delete function if the node is not inserted at the end, then it needs to be deleted. The get function updates the parent array and we lastly need a print function to map the array.