1. **[20 points]**
   (a) Give a detailed pseudocode to implement a max-priority queue (operations \textsc{Insert}, \textsc{DeleteMax}, \textsc{IncreaseKey}) using a linked list. Design the structure so that \textsc{DeleteMax} becomes as fast as possible. What are the asymptotic worst-case running times of each operation?
   
   (b) Consider the special case where first the keys 1, 2, 3, \ldots, n are inserted into the priority queue you implemented in part (a), and then n \textsc{DeleteMax} operations are performed. What is the asymptotic running time for this whole sequence of operations?
   
   (c) As part (b), but the keys are inserted in the order n, n − 1, n − 2, \ldots, 3, 2, 1.
   
   Justify all the running times briefly (with a few sentences, without going into the details of the pseudocode).

2. **[10 points]** Give a brief description of B-trees: what can be stored in each node, and what conditions must be satisfied. Explain, using an example and pictures, how the necessary conditions are maintained during an insert operation.

3. **[10 points]** What is meant by a sorting algorithm being \textit{stable}? Give an example of a situation that requires a stable sorting algorithm.

   Give a concrete example of a situation where some sorting algorithm covered in the course is seen to be unstable. That is, pick a suitable algorithm and input and demonstrate enough of how the algorithm works that the instability can be seen.

4. **[20 points]** Consider a data communication network with \(n\) computers \(c_1, \ldots, c_n\). The communications are represented as a list of triples \((i, j, s)\) with the interpretation “computers \(c_i\) and \(c_j\) are connected by a link that can transfer \(s\) bits per second in either direction.” There are \(m\) such links.

   We can move \(s\) bits per second from computer \(c_i\) to computer \(c_j\), if we can trace a connection from \(c_i\) to \(c_j\), possibly over several consecutive links, such that each individual link of the connection can transfer at least \(s\) bits per second. (This of course is a simplification of the real situation.)

   Further, we say that the \textit{capacity} of the whole network is the largest number \(s\) such that between any two computers in the networks it is possible to move \(s\) bits per second.

   (a) Give an efficient algorithm that receives as input a network as described above and a number \(s\), and checks whether the capacity of the network is at least \(s\).

   (b) Give an efficient algorithm that receives as input a network as described above, and returns the capacity of the network.

   Both in part (a) and part (b), justify that your algorithm indeed gives the desired result, and state the time complexity of your algorithm. You may consider as known any algorithm or result given in the course material.

---

Tehtävät suomeksi käänöpuolella