Algorithmic Design: Optional Exercises (Exam Preparation)

30/4/2020

1. Let H be a `Min-Heap` containing n integer keys and let k be an integer value. Consider to extend above data structure to efficiently support the following *in-place* operations:

   - `InsertKey(H, k)`: inserts the value k in H;
   - `RetrieveMax(H)`: returns the maximum value in H without deleting it from H;
   - `DeleteMax(H)`: deletes the maximum value from H.

   Solve the following exercises by using the procedures seen during the course:

   (a) Write the pseudo-code of the in-place procedure `RetrieveMax(H)` to efficiently return the <u>maximum</u> value in H without deleting it and evaluate its complexity;

   (b) Write the pseudo-code of the in-place procedure `DeleteMax(H)` to efficiently deletes the <u>maximum</u> value from H and evaluate its complexity;

   (c) Provide a working example for the worst case scenario for the procedure `DeleteMax(H)` of point 1b on a heap H consisting in 8 nodes and simulate the execution of the function itself;

   (d) Let $\max(H)$ be the maximum key in H and let m be an integer value such that $m > \max(H)$. What are the asymptotic complexities of the in-place procedure `InsertKey(H, m)` in both best and worst case scenarios? Motivate the answer.

2. Let A be a array of n integer values (i.e., the values belong to $\mathbb{Z}$). Consider the problem of computing a vector B such that, for all $i \in [1, n]$, $B[i]$ stores the number of elements smaller than $A[i]$ in $A[i+1, \dots, n]$. More formally:

   $$B[i] = |\{z \in [i+1, n] |\ A[z] < A[i]\}|$$

   (a) Evaluate the array B corresponding to $A = [2, -7, 8, 3, -5, -5, 9, 1, 12, 4]$.

(b) Write the pseudo-code of an algorithm belonging to $O(n^2)$ to solve the problem. Prove the asympotic complexity of the proposed solution and its correctness.

(c) Assuming that there is only a constant number of values in A different from 0, write an efficient algorithm to solve the problem, evaluate its complexity and correctness.

3. Let T be a RB-Tree.

   (a) Give the definition of RB-tree;

   (b) Write the pseudo-code of an efficient procedure to compute the height of T. Prove its correctness and evaluate its asymptotic complexity;

   (c) Write the pseudo-code of an efficient procedure to compute the black-height of T. Prove its correctness and evaluate its asymptotic complexity.

4. Let $(a_1, b_1), ..., (a_n, b_n)$ be n pairs of integer values. They are lexicographically sorted if, for all $i \in [1, n-1]$, the following conditions hold:

   - $a_i \leq a_{i+1}$;
   - $a_i = a_{i+1}$ implies that $b_i \leq b_{i+1}$.

   Consider the problem of lexicographically sorting n pairs of integer values.

   (a) Suggest the opportune data structure to handle the pairs, write the pseudo-code of an efficient algorithm to solve the sorting problem and compute the complexity of the proposed procedure;

   (b) Assume that there exists a natural value k, constant with respect to n, such that $a_i \in [1, k]$ for all $i \in [1, n]$. Is there an algorithm to solve the sorting problem more efficient than the one proposed as solution of Exercise 4a? If this is the case, describe it and compute its complexity, otherwise, motivate the answer.

   (c) Assume that the conditions of Exercise 4b hold and that there exists a natural value h, constant with respect to n, such that $b_i \in [1, h]$ for all $i \in [1, n]$. Is there an algorithm to solve the sorting problem more efficient than the one proposed as solution of Exercise 4a? If this is the case, describe it and compute its complexity, otherwise, motivate the answer.

5. Let $G = (V, E)$ be a directed acyclic graph. Given $v \in V$, the *height $h[v]$ of v* is the maximum length of a path from v to any node without outgoing edges.

   (a) Write the pseudo-code of a procedure `Height(G)` that computes the height $h[v]$ of all the nodes $v \in V$.

   (b) Evaluate the complexity of `Height(G)` and prove its correctness.