Data Structures - Assignment no. 7

Remarks:

- **Write both your name and your ID number very clearly on the top of the exercise. Write your exercises in pen, or in clearly visible pencil. Please write *very* clearly.**
- **Recall that 80% of the theoretical exercises must be submitted. The exercises can and must be worked on and submitted alone.**
- **Give correctness and complexity proofs for every algorithm you write.**
- **For every question where you are required to write pseudo-code, also explain your solution in words.**

1. Find the order of growth for the following recursively given functions. Explain your answer. Assume that $T(n)$ is constant for $n \leq 2$.

   (a) $T(n) = 8T(n/2) + 3n^2$

   (b) $T(n) = 2T(\sqrt{n}) + 1$

   (c) $T(n) = T(n-1) + 1/n$.

   (d) $T(n) = 2T(n/2) + n\log^4 n$ (Hint: The approach to solving this is similar to the approach to solving $T(n) = 2T(n/2) + n$).

2. In a double hashing (without deletions) assume in every entry i in the table, we keep a counter c_i that equals to the number of keys k for which $h_1(k) = i$.

   (a) How would you use this fact in order to reduce (usually) the number of accesses to the table entries during an unsuccessful search?

   (b) Show an example of an unsuccessful search in which the number of accesses to table entries is reduced from n to 2 (using your answer to (a)).

3. A family $\mathcal{H}$ of functions from U to $\{0, 1, \ldots, m-1\}$ is called *100-weakly universal* if for all $x, y \in U$ such that $x \neq y$ it holds that

$$Pr_{h \leftarrow \mathcal{H}}[h(x) = h(y)] \leq \frac{100}{m} .$$

   Let $U = \{0, 1, \ldots, u-1\}$, where u is much larger then m. Define f_a to be the function $f_a(x) = a \cdot x \mod m$. Show that $\mathcal{H} = \{f_a | 0 < a < m\}$ is not a 100-weakly universal family.

4. Draw a suffix tree for the string "yabbadabbado". Then show the search path for the sub-string "dab" and for the sub-string "bad".

5. Given two strings s_1, s_2, both of length n, and given an integer k, show an algorithm that finds the longest string s that appears at least k times in s_1 and at least k times in s_2. Make the algorithm as efficient as possible (in term of $O(\cdot)$ notation).
   Hint: First think of how to find all of the longest strings that appear in s_1 at least k times.

6. Let s be a string of length 10^6, which consists of $0.99 \cdot 10^6$ 'A's and $0.01 \cdot 10^6$ 'B's.

   (a) Build a Huffman tree for this string.

   (b) Calculate the length, in bits, of the Huffman encoding of the string.

   (c) Can you come up with a better method of encoding the string by a sequence of bits? Hint: $\log_2(10^6) \approx 20$.