MATEMATISKA INSTITUTIONEN
STOCKHOLMS UNIVERSITET
Avd. Beräkningsmatematik
Examinator: Lars Arvestad

Tentamensskrivning i
DA3018 Datalogi för matematiker
7.5 hp
2019-06-05

This is a translation to English of an old exam in Swedish.

- No aids allowed, except writing materials.
- **Write clearly.** Hard-to-read answers risk zero points.
- Justify all your answers (unless otherwise stated).
- **Grading thresholds:** E: 25, D: 30, C: 35, B: 40, A: 45

---

1. Define the following terms (as defined/used in the course).

(a) Asymptotic complexity. (2p)

(b) Stack. (2p)

(c) Topologic sorting in a directed graph. (2p)

(d) Breadth first search. (2p)

2. Binary trees have zero, one, or two children.

(a) Write pseudo code for a recursive function that counts the number of vertices in a binary tree with exactly one child. (3p)

(b) Argue for why your algorithm is correct. (1p)

(c) Analyze the time complexity. (1p)

3. Consider the following computational problem.

- Input: a string t and a string p, where t is longer than p.
- Output: indices in t where instances of p are found, or None if p not present in t.

Here is pseudo code for an algorithm that finds p in a string t.

```
def find_pattern(p, t):
    for i in range(len(t)-len(p)+1):
        for j in range(len(p)):
            if t[i+j] != p[j]:
                break          # Leave the j-loop
        else:                  # else belongs to the for-loop!
            return i           # Returns i if j-loop finishes
    return None                # Returns None if p not in t
```

(a) Is the algorithm suitable for the computational problem? Justify your answer. (2p)

(b) What is the time complexity of `find_pattern`? (2p)

4. Let s be a string with n characters, and $s[i : j]$ the substring that starts at position i and ends at position $j - 1$. A *suffix* of s is a substring $s[i : n]$, for some $0 \leq i < n$.

If A is a *suffix array* for s, then $A[j]$ is a starting position for the suffix that is found at position j in an alphabetic sort of all suffices of s. For all $1 \leq i < n$ we have that $s[A[i-1] : n] < s[A[i] : n]$.

Suppose you are given a string t with its suffix array A and want to find an index for a string p in t, or `false` if p is not in t.

(a) Suggest an algorithm that is faster than `find_pattern` from question 3. (3p)

(b) Analyze the time complexity of your algorithm. (2p)

5. A friend of yours in the tourist business contacts you with the following questions: ”Uh, like, at work we want to guide tourists to a number of addresses in the Old Town, but we would rather walk as little as possible. What should we do?”

Formalize a computational problem that solves your friend’s dilemma. (5p)

6. Suppose you work in a project with geographical information and need a hash function for addresses stored as objects in the class `Address`:

```
public class Address {
  private String street;
  private int    street_number;
  private String town;
```

(a) Write pseudo code for a hash function for `Address` objects to a hash table of size s. (3p)

(b) Argue why your hash function is suitable, with regards to how a data structure is expected to work. (2p)

7. Suppose we use a binary search tree, without balancing, to sort data in an array a, using this algorithm:

```
def bst_sort(a):
   bst = new BinarySearchTree()    # Create empty tree
   for elem in a:
      bst.insert(elem)
   ordered_data = array(a.size()) # Create array of same size as a
   i = 0
   for elem in bst.traversal():    # Iterate through elements in order
      ordered_data[i] = elem
      i++
   return ordered_data
```

(a) What sorts of traversal is needed by the iterator `traversal()`? (2p)

(b) What is the time complexity of `bst_sort`? (2p)

(c) Is the sort stable? (2p)

8. Suppose you work with graphs of variable size, where the average vertex degree is a constant α.

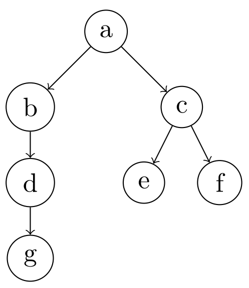
(a) What graph representation would you choose? (3p)

(b) What is the asymptotic time complexity of Breadth-First-Search on these graphs, given the degree constraint? (2p)

9. Write pseudo code for a function `level_order` that print the vertices of a rooted binary tree ordered by their level in the tree. The root is the first level, and its children are on level two, etc. The alphabetical order of the elements in the tree in figure 1 is also in level order, *abcdefg*, but the solution is not unique: *acbfdeg* is also a correct solution.

Argue why your algorithm is correct and make a time complexity analysis. Hint: use a queue.

(5p)



Figur 1: Illustration for question 9, a tree in four levels. You get a ”level order” of elements in this tree from their alphabetical order.