Mathematisch-Naturwissenschaftliche Fakultät
Institut für Informatik, Anja Rey

Exercise Sheet 1

for the lecture on

Advanced Programming and Algorithms – Part II

This exercise sheet contains exercises for self-study and discussion.
If you would like feedback on your solutions, please upload a PDF via ILIAS until Monday, 15th April, 12:30 pm.

Submissions are no longer required to pass this course.

Discussion in the exercise classes from 15th April until 19th April, 2024.

Problem 1 to hand in: *Recursive Algorithms*

The following Python code and pseudocode contains errors. For each subtask determine an error and write down how it can be repaired.

Choose one subtask and prove formally that the repaired version is now correct.

a) Given a non-negative integer n, the following code should compute the power 3^n recursively.

```
1    def power_of_three(n):
2        if n == 0:
3            return 1
4        power_of_three(n - 1) * 3
```

b) Given a list a that contains n integers, the following code should compute the index of the minimum entry in a recursively.

```
5    def minimum_index(a, n):
6        if n == 1:
7            return 0
8        min = minimum_index(a[1:n], n - 1)
9        if a[min] < a[0]:
10           return min
11       else:
12           return 0
```

c) Given a directed graph $G = (V, E)$, a source vertex $s \in V$, and a target vertex $t \in V$, the following code should compute the number of different paths from s to t.

number_of_paths($G = (V, E)$, s, t):

1 **if** $t = s$ **then**
2 | **return** 1
3 **return** $\sum_{(u,t)\in E}$ number_of_paths(G, s, u)

Problem 2 as a programming exercise: *Depth First Search*

a) Implement a recursive version and an iterative version of Depth First Search in Python. Compare the two versions – what are their advantages and disadvantages?

Hint: Before you look it up, think about how you would solve this yourself.

b) Take a closer look into how Depth First Search is implemented within `networkx`. What differences are there and when would you use what?

c) Implement an algorithm that returns a topological order of the vertices of a given acyclic graph.

Problem 3 for discussion: *Acyclic Graph*

Design an algorithm that determines whether a given undirected graph $G = (V, E)$ contains a cycle. How can this be achieved with a running time in $\mathcal{O}(|V|)$ (indepented of the number of edges).

Problem 4 for discussion: *Recursive vs Iterative Algorithms*

Take a closer look at the three (repaired) functions from Problem 1. How can they be implemented iteratively? What are their respective running times? What are other advantages and disadvantages of recursive functions?

Problem 5 for discussion: *Solving Recurrences*

In the context of running times of recursive algorithms, we will come across recurrences of the form

$$T(n) = \begin{cases} c & n = 1 \\ a \cdot T(\frac{n}{b}) + f(n) & n > 1. \end{cases}$$

How can we solve them to obtain a closed form for $T(n)$ and classify them asymptotically?

Examples:

a) $T(n) = \begin{cases} 1 & n = 1 \\ 2 \cdot T(\frac{n}{2}) + 1 & n > 1. \end{cases}$

b) $T(n) = \begin{cases} 1 & n = 1 \\ 9 \cdot T(\frac{n}{3}) + n^2 & n > 1. \end{cases}$

c) $T(n) = \begin{cases} 1 & n = 1 \\ 16 \cdot T(\frac{n}{2}) + n^3 & n > 1. \end{cases}$

d) $T(n) = \begin{cases} 1 & n = 1 \\ 16 \cdot T(\frac{n}{4}) + n^3 & n > 1. \end{cases}$

e) $T(n) = \begin{cases} 1 & n = 1 \\ 2 \cdot T(\sqrt{n}) + \log n & n > 1. \end{cases}$