

Scientific Theory in Informatics A1N

Complexity Theory Exercises

1. Determine the order of time complexity $O(g(n))$ of the following C++ code-segments. Show how you arrive at your answer by deriving $g(n)$ from an analysis of the frequency of execution of each statement.


```
int i, j, n;
n = 100;
while (i < n) {
    b[i] = 0;
    i++;
}
```

2. Determine the order of time complexity $O(g(n))$ of the following C++ code-segments. Show how you arrive at your answer by deriving $g(n)$ from an analysis of the frequency of execution of each statement.

```
int i, j, n;
cin >> n;
i = 5;
for (i=0; i <= n; i++)
    for (j=1; j <= sqrt(i); j++)
        cout << "Hello " << i << j;
```

3. Determine the order of time complexity $O(g(n))$ of the following C++ code-segments. Show how you arrive at your answer by deriving $g(n)$ from an analysis of the frequency of execution of each statement.

```
do {
    cin >> symbol >> probability;
    if ((probability > 0) && (probability <= 1.0)) {
        add_to_tree(tree_number, symbol, probability);
    } while (probability != 0);
```

Note that function call `add_to_tree()` has complexity , where n is the number of nodes in the tree.

4. Determine the order of time complexity $O(g(n))$ of the following C++ code-segments. Show how you arrive at your answer by deriving $g(n)$ from an analysis of the related recurrence formula.

```
// A recursive function

void f(int n) {
    if (n>0) {
        cout << n << " ";
        f(n-1);
    }
}
```

5. Determine the order of complexity of the following code segment.

[15 marks]

```
// A recursive function

void f(LIST_TYPE list, int n){

    if (n>1) {
        cout << n << " ";
        scan_list(list);
        f(list, n-2);
    }
}
```

Assume that the function `scan_list()` has complexity $O(n^2)$.

6. The recurrence formula specifying the space complexity of a recursive algorithm is given by:

$$S(n) = 4 S(n-4)$$

where $S(0) = k$, a constant.

Determine the order of space complexity of this algorithm.

7. Does the following code segment have reasonable or unreasonable complexity? Prove it.

```
void f(int n, char a, char b, char c) {
    if (n>0) {
        f(n-1, a, c, b);
        printf("%c %c %c\n", a, b, c);
        f(n-1, c, b, a);
    }
}
```

8. Explain why the Travelling Salesman problem is NP-Complete and explain how we can solve the problem, even though it is formally intractable.
9. What is an NP-Complete problem? Give an example.
10. Explain the following expression and, in doing so, explain the difference between *complexity* and *order of complexity*:

if $f(n) \leq g(n)$
then $O(f(n) + g(n)) = O(g(n))$