```
Assigment02
├─ Solution_02_MartinNord.pdf
├─ Task01
│  └─ ...
├─ Task02
│  └─ ...
└─ ...
```

Figure 1: Exemplary directory structure

Assignment 3

Submission procedure

Your submissions must meet the following requirements:

1. Please organize your solution into a single tar or zip file.
2. Please submit your solution as pure Python files (.py).
3. Provide sufficient documentation and/or tests so that we can run your code.
4. Structure and work on your submission. For example:
   - Create a directory for the Assignment, e.g. name the directory "Assignment3" for Assignment 3.
   - Create a subdirectory for each task with the task number as name, e.g. "Task01" for task 1.
   - Place the solution within the respective subdirectory.
   - Solution to pen & paper tasks must be aggregated to a single pdf file. The file name is expected to follow the pattern be "Solution_AB_Name_XY.pdf", where XY must be substituted with your name and AB with the exercise sheet number.
   - See Figure 1 for an exemplary directory hierarchy.
5. Upload the solution before deadline.

Task 1 *Selection sort*

4 p.

Implement selection sort algorithm and test it on a random list. Use a list data structure implemented in Assignment 1.

Task 2 *Quick sort*

4 p.

Implement quick sort algorithm and test it on a random list. Use a list data structure implemented in Assignment 1.

Task 3 *Graphs*

2 p.

Implement a graph data structure with `addNode(u)`, `addEdge(u,v)`, `removeNode(u)`, `removeEdge(u,v)` and all other functions needed for Task 4. You are free to choose either a list of edges or an adjacency list as the underlying data structure. Justify your choice and its implications for the following tasks. Provide a test for your data structure and show that all methods work.

Task 4 *DFS*

2 p.

Implement a function `DFS(G,v)` that searches for a node v in a graph G using depth first search. Provide an example to test this function.

Task	1	2	3	4	total
Points	4	4	2	2	12
reached					