INF234 — Algorithms

Compulsory Assignment 1, due October 2nd, 20:00

1. Set theory

For each of the following operators, do the following:

1. State the name of the operator
2. Explain what it means
3. Give two correct expressions using the operator
4. Give two incorrect expressions using the operator

Operators and symbols:

a. $\emptyset$
b. $\cup$ and $\cap$
c. $\subseteq$ and $\supseteq$
d. $\subset$ and $\supset$
e. $\in$ and $\notin$
f. $\setminus$
g. $|\cdot|$

Solution to (a.):

1. $\emptyset$ is the *empty set symbol*
2. It is the symbol referring to the set with no elements, $\{\}$.
3. Correct:
   - : $\emptyset = \{\}$
   - $\emptyset \subseteq X$
4. Incorrect:
   - $\emptyset \supseteq \{1\}$
   - $|\emptyset| > 3$

2. Python random integer

Open a Python 3 interpreter and run the following code, with *your username* substituted for `abc123`.

```
import random
random.seed("abc123")
print(random.randint(1, 9))
```

What is the output?

3. Algorithms

Here are X problems. You are supposed to do only the one that corresponds with the answer in the previous problem.

Use the following "API" for graphs and intervals.

```
from collections import namedtuple as T
Graph = T("Graph", "V N")
Interval = T("Interval", "s f")
```

The intended meaning of these "APIs" are to be used like this:

```
G = Graph(V=[0, 1, 2, 3], N=[[1, 2], [0, 2, 3], [0, 1], [1]])

for v in G.V:
    print(v, G.N[v])

intervals = [Interval(1, 3), Interval(2, 4), Interval(1, 4)]
for I in intervals:
    for J in intervals:
        if I.s < J.s < I.f:
            print(f"{I} overlaps {J}")
```

Scoring. You get points for correctness, clarity, elegance, and optimal running time, as well as for a correct and convincing argument of correctness.

1. Bipartite graphs Give Python code for a given connected and undirected graph that either outputs a bipartition, or outputs an odd cycle.

The bipartition should map each vertex to `0` or `1`, and the first vertex in `G.V` should get labelled `0`.

The function should have a docstring explaining expected input and output.

Give a convincing argument of correctness.

2. Topological ordering Give Python code computing a topological ordering for a connected DAG. The output should be a list of vertices in topological order.

The function should have a docstring explaining expected input and output.

Give a convincing argument of correctness.

3. Strongly connected components Give Python code computing the set of SCCs for a connected directed graph. The output should be a list of lists, each list containing an SCC.

The function should have a docstring explaining expected input and output.

Give a convincing argument of correctness.

4. Interval scheduling Give Python code for scheduling a maximum number of non-overlapping intervals. You may assume unique start and finish times.

Output the intervals that are selected.

The function should have a docstring explaining expected input and output.

Give a convincing argument of correctness.

5. Interval partitioning Give Python code for partitioning intervals into non-overlapping partitions. You may assume unique start and finish times.

Output a list of lists, the list of intervals belonging to each partition.

The function should have a docstring explaining expected input and output.

Give a convincing argument of correctness.

6. Scheduling to minimize lateness Using the following "API",

```
from collections import namedtuple as T
Job = T("Job", "t d")  # timespan and deadline
```

give Python code for scheduling intervals in a non-overlapping way to minimize the maximum lateness.

Output a mapping from each interval to its `start` and `finish` time, sorted by starting time.

The function should have a docstring explaining expected input and output.

Give a convincing argument of correctness.

7. Kruskal's clustering algorithm Give Python code for clustering vertices into clusters using Kruskal's clustering algorithm. The function should take as input a graph and an integer k denoting the number of clusters in the output.

Output a list of lists, where each list is the vertices belonging to a specific cluster.

The function should have a docstring explaining expected input and output.

Give a convincing argument of correctness.

8. Dijkstra's algorithm Give Python code for computing single source shortest path. The input should be a directed weighted graph and a vertex s. You may use something like this to represent a weighted graph:

```
D = Graph(
    V=[0, 1, 2], N=[[(1, 0.1), (2, 0.5)], [(2, 0.3)], [(1, 0.2)]]
)

for v in D.V:
    for u, weight in D.N[v]:
        print(v, u, weight)
```

Output a mapping of each vertex to its distance from s.

The function should have a docstring explaining expected input and output.

Give a convincing argument of correctness.

9. Huffman code Give Python code for computing the Huffman encoding of a string. When two tokens have the same weight, take the lexicographically smallest. This ensures unique (but still optimal) encoding.

Output a mapping of each symbol to its (binary) encoding.

The function should have a docstring explaining expected input and output.

Give a convincing argument of correctness.