Lab 1: Set Covering

First lab + peer review. List this activity in your final report, it will be part of your exam.

Task

Given a number N and some lists of integers $P = (L_0, L_1, L_2, \dots, L_n)$, determine, if possible, $S = (L_{s_0}, L_{s_1}, L_{s_2}, \dots, L_{s_n})$ such that each number between 0 and $N-1$ appears in at least one list

$$\forall n \in [0, N-1] \ \exists i : n \in L_{s_i}$$

and that the total numbers of elements in all L_{s_i} is minimum.

Instructions

- Create the directory `lab1` inside the course repo (the one you registered with Andrea)
- Put a `README.md` and your solution (all the files, code and auxiliary data if needed)
- Use `problem` to generate the problems with different N
- In the `README.md`, report the the total numbers of elements in L_{s_i} for problem with $N \in [5, 10, 20, 100, 500, 1000]$ and the total number on $nodes$ visited during the search. Use `seed=42`.
- Use `GitHub Issues` to peer review others' lab

Notes

- Working in group is not only allowed, but recommended (see: Ubuntu and Cooperative Learning). Collaborations must be explicitly declared in the `README.md`.
- Yanking from the internet is allowed, but sources must be explicitly declared in the `README.md`.

Deadline

- Sunday, October 16th 23:59:59 for the working solution
- Sunday, October 23rd 23:59:59 for the peer reviews

In [ ]:

```python
import random
```

In [ ]:

```python
def problem(N, seed=None):
    random.seed(seed)
    return [
        list(set(random.randint(0, N - 1) for n in range(random.randint(N // 5, N /
        for n in range(random.randint(N, N * 5))
    ]
```

In [ ]:

```
import logging


def greedy(N):
    goal = set(range(N))
    covered = set()
    solution = list()
    all_lists = sorted(problem(N, seed=42), key=lambda l: len(l))
    while goal != covered:
        x = all_lists.pop(0)
        if not set(x) < covered:
            solution.append(x)
            covered |= set(x)

    logging.info(
        f"Greedy solution for N={N}: w={sum(len(_) for _ in solution)} (bloat={(sum
    )
    logging.debug(f"{solution}")
```

In [ ]:

```
logging.getLogger().setLevel(logging.INFO)
for N in [5, 10, 20, 100, 500, 1000]:
    greedy(N)
```

```
INFO:root:Greedy solution for N=5: w=5 (bloat=0%)
INFO:root:Greedy solution for N=10: w=13 (bloat=30%)
INFO:root:Greedy solution for N=20: w=46 (bloat=130%)
INFO:root:Greedy solution for N=100: w=332 (bloat=232%)
INFO:root:Greedy solution for N=500: w=2162 (bloat=332%)
INFO:root:Greedy solution for N=1000: w=4652 (bloat=365%)
```

In [ ]:

```
%timeit greedy(1_000)
```

```
INFO:root:Greedy solution for N=1000: w=4652 (bloat=465%)
INFO:root:Greedy solution for N=1000: w=4652 (bloat=465%)
INFO:root:Greedy solution for N=1000: w=4652 (bloat=465%)
INFO:root:Greedy solution for N=1000: w=4652 (bloat=465%)
INFO:root:Greedy solution for N=1000: w=4652 (bloat=465%)
INFO:root:Greedy solution for N=1000: w=4652 (bloat=465%)
INFO:root:Greedy solution for N=1000: w=4652 (bloat=465%)
INFO:root:Greedy solution for N=1000: w=4652 (bloat=465%)
1.21 s ± 85 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
```

In [ ]: