Algorithmics
Final Exam #3 (P3)

Undergraduate 2^{nd} year - S3
Epita

January 5, 2021 - 9 : 30

Instructions (read it) :

- □ You must answer on **the answer sheets provided.**
  - No other sheet will be picked up. Keep your rough drafts.
  - Answer within the provided space. **Answers outside will not be marked**: Use your drafts!
  - Do not separate the sheets unless they can be re-stapled before handing in.
  - Penciled answers will not be marked.
- □ The presentation is negatively marked, which means that you are marked out of 20 points and the presentation points (maximum of 2) are taken off this grade.
- □ **Code:**
  - All code must be written in the language `Python` (no C, Caml, Algo or anything else).
  - **Any `Python` code not indented will not be marked.**
  - All that you need (classes, types, routines) is indicated where needed!
  - You can write your own functions **as long as they are documented** (we have to know what they do).
    In any case, the **last** written function should be the one which answers the question.
- □ Duration : 2h

Exercise 1 (In the depth of the spanning forest – *3 points*)

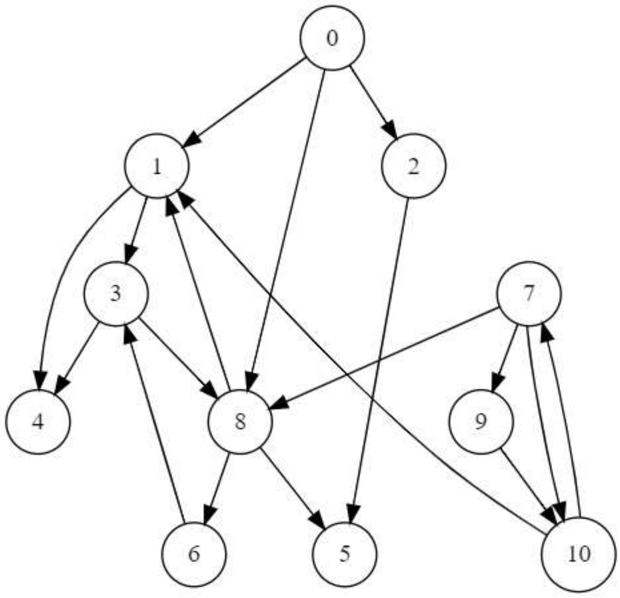

Figure 1: A digraph

1. Build the spanning forest of the graph `G` for the depth first search from vertex 0. Vertices are encountered in increasing order. *Add to the forest the various kinds of edges met during the traversal, with an explicit legend.*

2. Fill-in the vectors containing the meeting order of vertices in prefix and suffix established with a single counter starting at 1 corresponding of the previous traversal.

Exercise 2 (Union-Find – *4 points*)

Let G be the graph $< S, A >$ with vertices numbered from 0 to 13. The algorithms seen in lecture `find` (with path compression) and `union` (union by rank) give the following vector p from the list of edges A:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
p	5	8	5	8	11	-4	5	10	-6	12	8	12	-4	8

1. Give the number of vertices of each connected component of G (the order does not matter).

2. Which additional edges will **be enough** to make the graph connected?

3. Among the following chains, which can not exist in G?
   - $3 \leftrightsquigarrow 7$
   - $11 \leftrightsquigarrow 6$
   - $0 \leftrightsquigarrow 13$
   - $4 \leftrightsquigarrow 9$

4. We add the edge 7 – 4 to the graph G (with union by rank and path compression). Give the new vector p.

Exercise 3 (Distance from start – *5 points*)

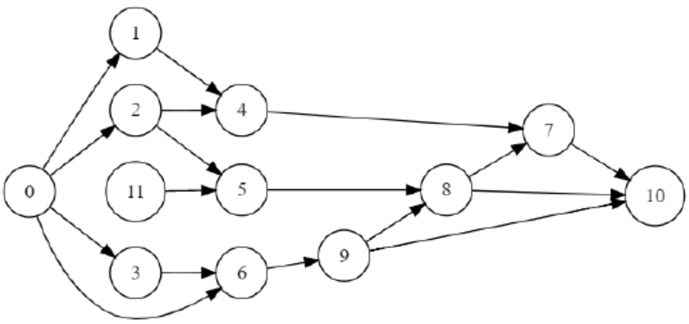

Figure 2: Digraph `G1`

The aim here is to find the set of vertices whose distance from a starting vertex is in the interval $[d_{min}, d_{max}]$.

Write the function `dist_range(`G`,` src`,` $dmin$`,` $dmax$`)` that returns the list of vertices that are at a distance between $dmin$ and $dmax$ from the vertex src in the graph G (with $0 < dmin \leq dmax$).

```
>>> dist_range(G1, 0, 2,3)
[4, 5, 9, 7, 8, 10]

>>> dist_range(G1, 0, 2,2)
[4, 5, 9]

>>> dist_range(G1, 0, 1,2)
[1, 2, 3, 6, 4, 5, 9]
```

Exercise 4 (Get cycle – *5 points*)

Using **imperatively a depth-first search**, write the function `get_cycle(`G`)` that searches for a cycle in the undirected graph G. If a cycle is found (any one, see examples below), the function returns it as a vertex list. Otherwise the function returns an empty list.

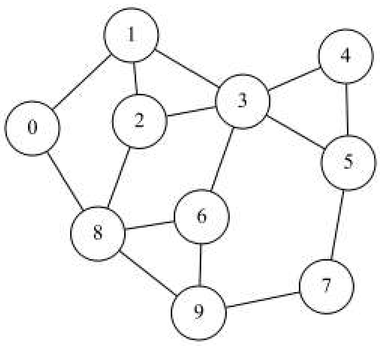

Figure 3: Graph `G2`

Examples of different results (different versions of the function) on the graph in figure 3:

```
>>> get_cycle(G2)
[1, 0, 8, 2, 1]

>>> get_cycle_2(G2)
[0, 8, 2, 1, 0]

>>> get_cycle_3(G2)
[1, 2, 3, 1]
```

Exercise 5 (What is this? – *3 points*)

The following functions are defined:

```python
def __build(G, x, D, P, NG):
    for y in G.adjlists[x]:
        if D[y] == None:
            D[y] = D[x] + 1
            __build(G, y, D, P, NG)
            NG.addedge(x, y)
        else:
            if D[y] < D[x] and not P[y]:
                NG.addedge(x, y)
    P[x] = True

def build(G):
    D = [None] * G.order
    P = [False] * G.order
    NG = Graph(G.order, True)
    for s in range(G.order):
        if D[s] == None:
            D[s] = 0
            __build(G, s, D, P, NG)
    return NG
```

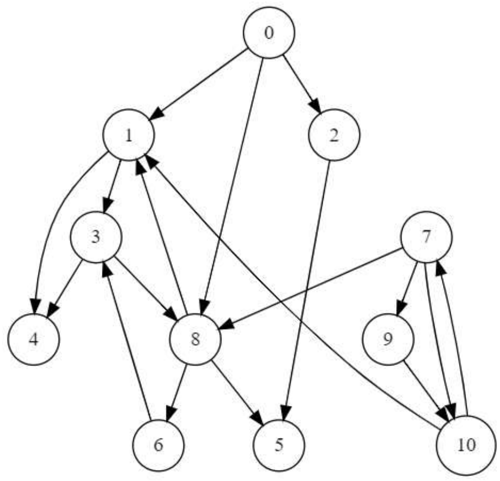

FIGURE 4 – Digraph G_4

1. Draw the graph resulting of the call `build(`G_4`)` where G_4 is the digraph in figure 4 (adjacency lists are sorted in increasing order).

2. For each vertex s, during the traversal:

   (a) What does `D[`s`]` represent?

   (b) What does `P[`s`]` represent?

Appendix

Classes `Graph` and `Queue` are assumed to be imported.

Graphs

All exercises use the implementation with adjacency lists of graphs.

Graphs we manage cannot be empty. There is neither multiple edges nor loops.

```
class Graph:
    def __init__(self, order, directed = False):
        self.order = order
        self.directed = directed
        self.adjlists = []
        for i in range(order):
            self.adjlists.append([])

    def addedge(self, src, dst):
        self.adjlists[src].append(dst)
        if not self.directed and dst != src:
            self.adjlists[dst].append(src)
```

Queues

- `Queue()` returns a new queue
- q`.enqueue(`e`)` enqueues e in q
- q`.dequeue()` returns the first element of q, dequeued
- q`.isempty()` tests whether q is empty

Others

- `range`
- `min, max`
- on lists:
  - `len(L)`
  - `L.append(elt)`
  - `L.pop()`
  - `L.pop(index)`
  - `L.insert(index, elt)`

  and any operator...

Your functions

You can write your own functions as long as they are **documented** (we have to know what they do).

In any case, the last written function should be the one which answers the question.