Practical session 4

Implement the UPGMA & NJ in a class called DistanceBasedAlgorithms. The output must be in Newick format

- What we have so far?
  - Sequence
  - Evolution
  - ToolsToWorkWithSequences

Helper classes

```
'''
Class to store a symmetric distance matrix

@author: olao
'''

class DistanceMatrix(object):

    '''
    Requires a list with the name of the species
    '''

    def __init__(self, name_of_species):
        '''
        Constructor
        '''
        self.name_of_species = name_of_species
        # create the list of lists
        self.m = []
        # create for each row the column
        for r in range(len(name_of_species)):
            #
            c = [0]*len(name_of_species)
            #
            self.m.append(c)
```

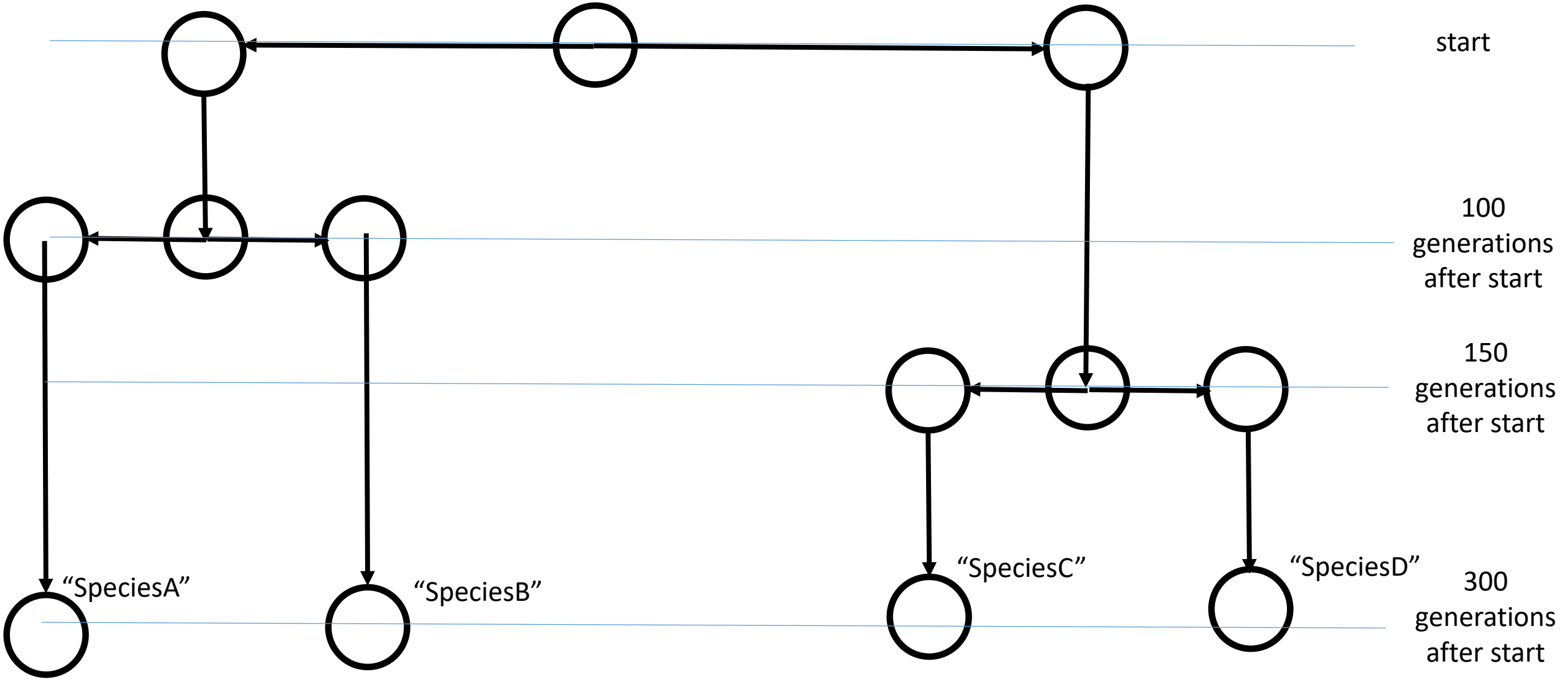
Create a new class called DistanceBasedAlgorithms

```
'''
@author: olao
'''

class DistanceBasedAlgorithms(object):
    '''
    classdocs
    '''

    def __init__(self, d_matrix):
        '''
        Constructor
        '''
        self.d_matrix = d_matrix



    def UPGMA(self):

        return 0


    def NJ(self):

        return 0
```

Workflow

- From Evolution, simulate four sequences following the demography in Session 3.
- Create an object of class DistanceMatrix to store the distances between pair of sequences.
- Use ToolsToWorkWithSequences to compute between each pair of sequences the observed distance. Store for each pair the value in the object of DistanceMatrix.
- Create an object of class DistanceBasedAlgorithms.
- Run upgma() and print the result.

Implement this model in the Evolution class



Pseudocode of UPGMA