Practical Session 3

In silico evolution!

What are we going to do?

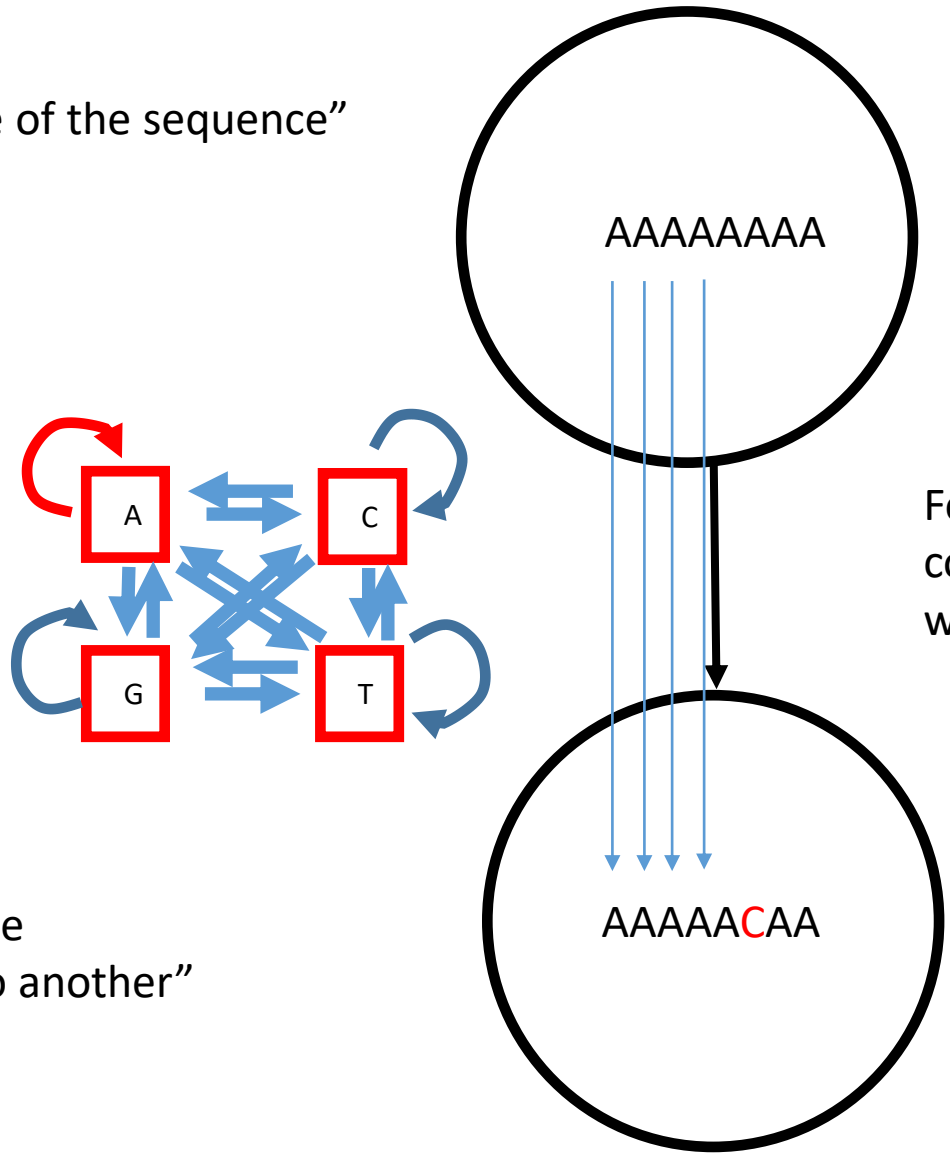
- We will evolve a sequence given a transition matrix between the nucleotides. For that, we will use a class called Evolution

How evolution works?

"Iterate over each nucleotide of the sequence"

MUTATION

"Decide if we stay in the same nucleotide or if we change to another"

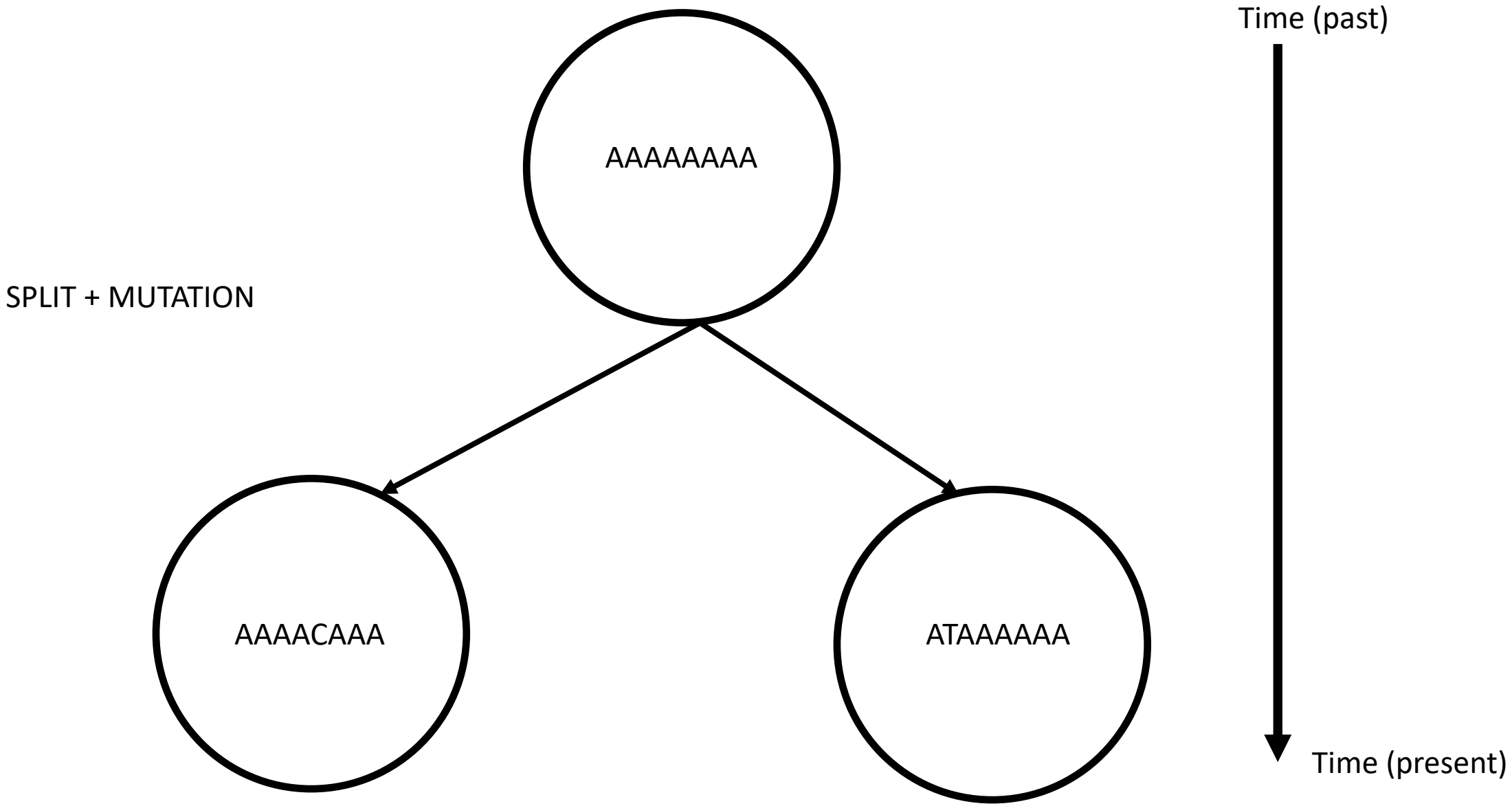


For each position we will count the number of times we mutate

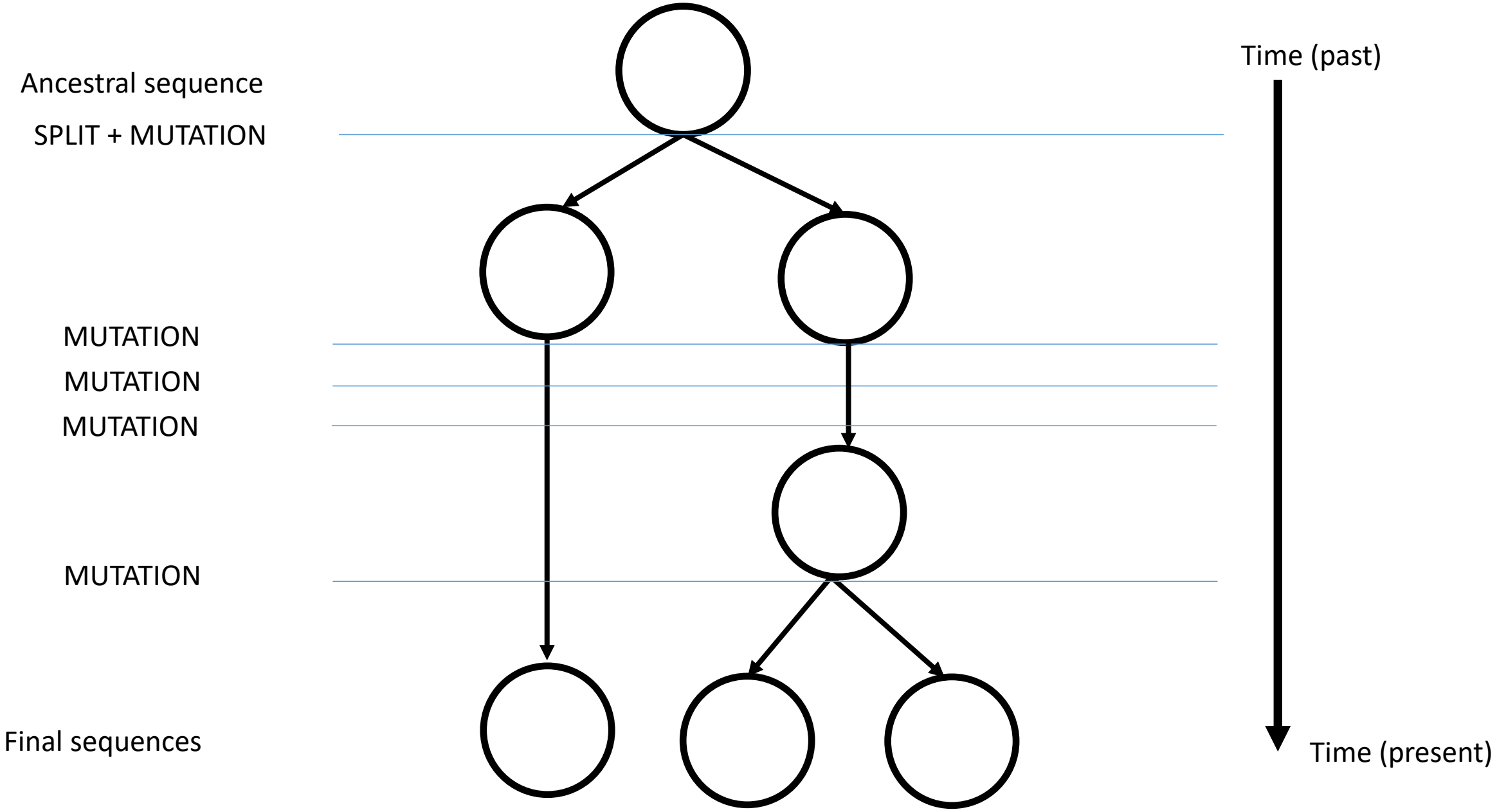
Time (past)

Time (present)

How evolution works?



Evolution


Ancestral sequence
SPLIT + MUTATION
MUTATION
MUTATION
MUTATION
MUTATION
Final sequences
Time (past)
Time (present)

Python code

- Use the class Sequence (load it). Comment each line of code. Why are we using list(str)? Place your answer when defining the constructor. What is this mutations=None doing???

```
def __init__(self, name, string_of_nucleotides,mutations=None):
        '''
        Constructor
        '''
        self.name = name

        if isinstance(string_of_nucleotides, str):
            self.string_of_nucleotides = list(string_of_nucleotides)
        elif isinstance(string_of_nucleotides,list):
            self.string_of_nucleotides = string_of_nucleotides
        else:
            raise TypeError("Use a string to initialize the sequence or a list of nucleotides")


        if mutations== None:
            self.mutations = [0]*len(string_of_nucleotides)
        else:
            self.mutations = mutations
```

Python code

- Check the class Evolution (load it)

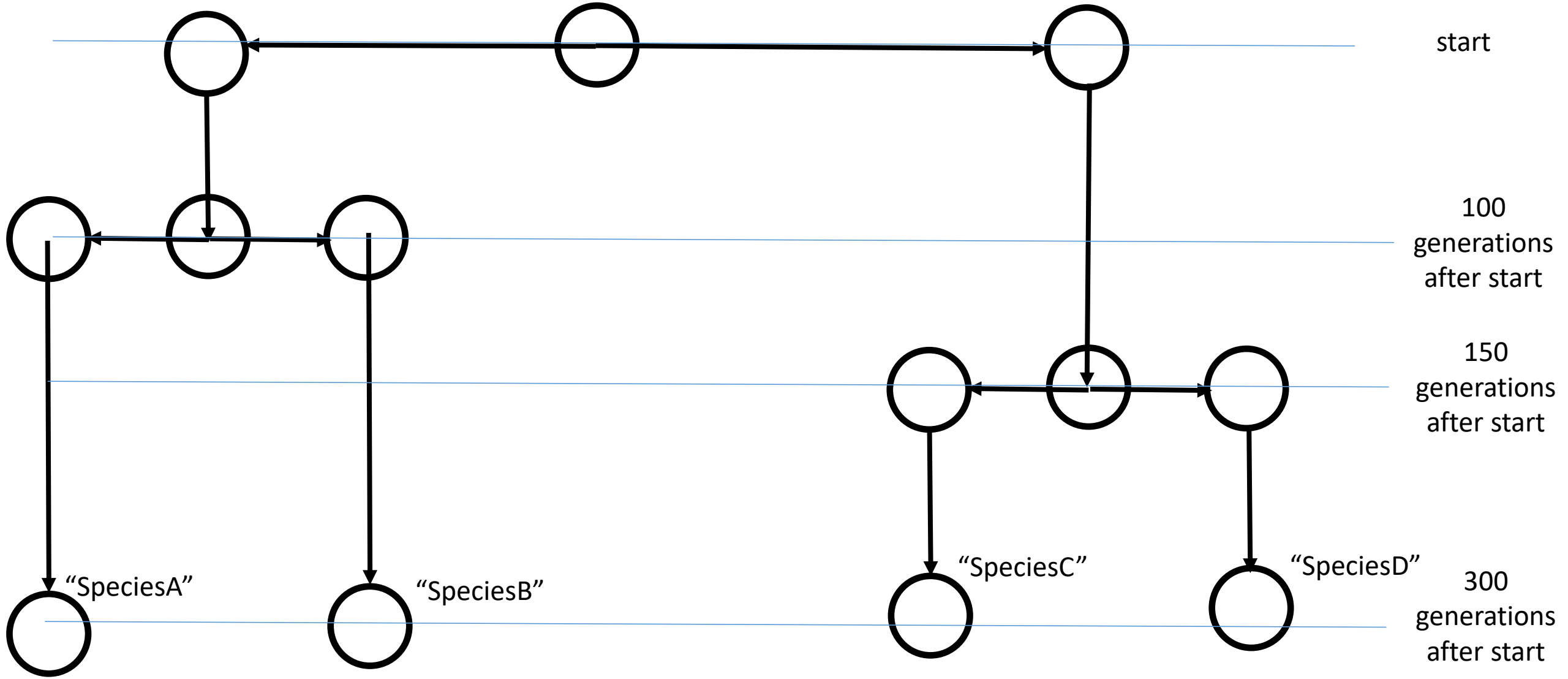
```
def __init__(self, ancestral_sequence, transition_matrix):
        '''
        Constructor
        '''
        if not isinstance(ancestral_sequence, Sequence):
            raise TypeError("ancestral_sequence must be a Sequence object!")

        self.evolving_sequences = {}
        self.evolving_sequences[ancestral_sequence.get_name()] = ancestral_sequence
        self.transition_matrix = transition_matrix
```

Python code

- Comment line by line the methods in Evolution

Implement this model in the Evolution class



Create a class called ToolsToWorkWithSequences

- Create a method called nucleotide_statistics that uses as parameter an object of type sequence and returns a dictionary with the percentage of A, C, T, G found in the sequence. Apply the method to each of the evolved sequences
- Create a method called observed_pairwise_nucleotide_distance that takes as parameters two sequences. Returns the number of nucleotides that for the same position are different in the two sequences