DEVELOPPER REPORT – INDEX.PY / MAP.PY

INDEX.PY

FMI CLASS

PRESENTATION OF THE CLASS

Object FMI is used to index a sequence of nucleotides in which we want to search a pattern (target sequence).

CREATION OF AN INSTANCE OF THE OBJECT

An FMI Object can be created with the function `FMI(genome)`.

ATTRIBUTES

SA (SUFFIX ARRAY)

- Type : `list`
- List of the same size as the genome which contains the indexes of the suffixes of the genome in alphabetical order.
- Dependencies: `import tools_karkkainen_sanders as tks`
    - Use of the function `simple_kark_sort(genome)` imported from `tks`

BWT (BURROWS WHEELER TRANSFORM)

- Type: `list`
- List of the same size as the genome which contains the Burrows Wheeler Transform of the genome sequence (aka the 2nd letter of each suffix of the genome in alphabetical order).
- Dependencies:
    - Needs the computation of the SA before.

F

- Type: `list`
- List of the same size as the genome which contains the 1st letter of the suffixes of the genome in alphabetical order.
- Dependencies:
    - Needs the computation of the SA before.

N

- Type: `Dictionary`
- Dictionary with 5 keys (“$”, “A”, “T”, “G”, “C”) which contains for each character key its number of occurrences in the genome.
- Dependencies:
    - Needs the computation of the BWT before.

R

- Type: `list`
- List of the same size as the genome which contains the index of each character in the genome for its type
    - For example, `R[2] = 1` means that if `genome[2] == "A"`, this is the 1st "A" in the genome.
- Dependencies:
    - Needs the computation of the BWT before.

METHODS

LEFT_FIRST(SELF, ALPHA, K)

- Return i such as the k-th suffix beginning with an alpha letter is the i-th suffix in the SA.
- Parameters:
    - `alpha: char`
        - letter to find in the FMI
    - `k: int`
        - index of the alpha letter to find in the FMI
- Error raised:
    - `ValueError`
        - if alpha is not in the alphabet. Must be one of 'A', 'T', 'G', 'C', '$'
- Returns:
    - `i: int`
        - i such as the k-th suffix beginning with an alpha letter is the i-th suffix in the SA.

GET_UP(SELF, ALPHA, START, STOP)

- Detects the first occurrence of alpha in bwt for i from start to stop
    - From start go down in the bwt as long as `bwt[line] != alpha` and `line <= stop`
        - if `bwt[line] == alpha`: returns the corresponding line
        - if `line > stop`: returns -1
- Parameters:
    - `alpha: char`
        - Letter to search in the BWT
    - `start: int`
        - Index where to begin the search in the BWT
    - `stop: int`
        - Index where to stop the search in the BWT
- Returns:
    - `line : int`
        - index where you can find the last occurence of alpha in bwt
    - `-1`
        - if alpha not in the `bwt[start:stop]` list

GET_DOWN(SELF, ALPHA, START, STOP)

- Detects the last occurrence of alpha in bwt for i from start to stop
    - From stop go up in the bwt as long as `bwt[line] != alpha` and `line >= start`
        - if `bwt[line] == alpha`: returns the corresponding line
        - if `line < start`: returns -1
- Parameters:
    - `alpha: char`
        - Letter to search in the BWT
    - `start: int`
        - Index where to begin the search in the BWT
    - `stop: int`
        - Index where to stop the search in the BWT
- Returns:
    - `line : int`
        - index where you can find the last occurence of alpha in bwt
    - `-1`
        - if alpha not in the `bwt[start:stop]` list

GET_OCCURRENCES(SELF, PATTERN)

- Returns the list of positions where the pattern P can be mapped to in the object.
- Parameters:
    - `pattern : str`
        - Pattern to search in the FMI
- Error raised:
    - `ValueError`
        - if P contains characters that are not in the alphabet.
- Returns:
    - `list _occu : list`
        - list of position index of occurences of the pattern in the genome represented by the FMI

Example:

```
In [1]:
genome_file = open(reference_file, 'r')
genome = genome_file.readlines()[-1]
genome_FMI = FMI(genome)
genome_FMI.get_occurences(“ATGCATGCATGCATGC”)


Out [1]:
[1, 523, 977, 1080]
```

OPTIONS

- Options that could be used:
    - `--ref` or `-r` + filename
        - Indicates the file containing the genome/sequence to index.
        - The file must be in fasta format : starts with a line explaining the file (will not be used) and the rest of the file contains the sequence.
    - `--out` or `-o` + filename
        - Indicates the output file in which we will store the index.
        - The name must be "name.dp" since the index Object is stored in a dumped file.
    - `--help` or `-h`
        - To visualise the README notice.
    - `--verbose` or `-v`
        - To print the options of the program and the time taken by the program.
- If other option mentioned in argument, raises an error:

```
except getopt.GetoptError as err:
        print(err)
        sys.exit(2)
```

USE OF THE PROGRAM

- When this program is launched (`if __name__ == "__main__"`), reads the options used and index the sequence stored in the reference file (option `--ref`).
- The reference file must be in FASTA format (see README.md).
- If the reference file is not found, raise an error and exit.

MAP.PY

GLOBAL VARIABLES

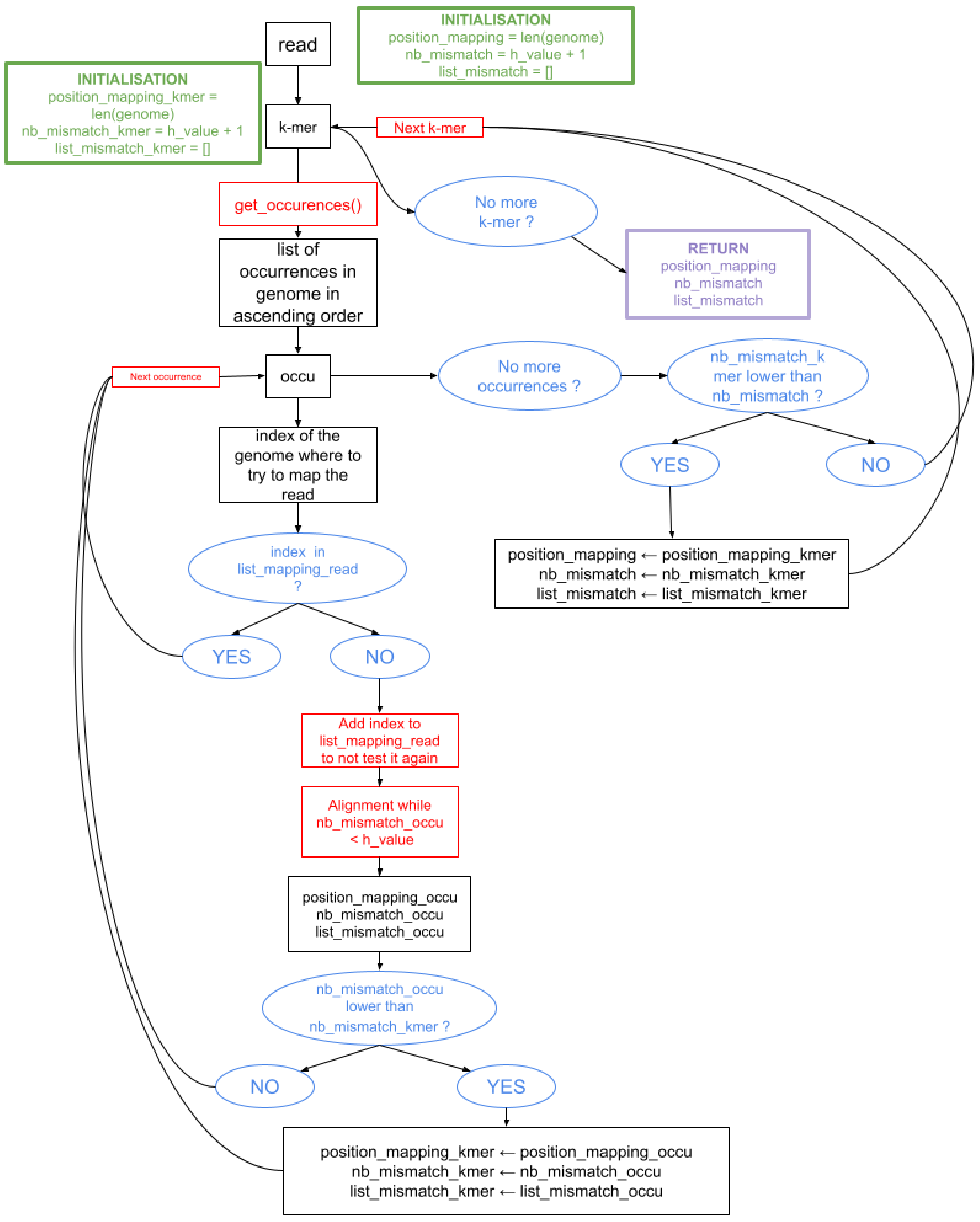
OPTIONS	List of options chosen by the user
REMAINDER	List of options chosen by the user that are not used in the program
REFERENCE_FILE	Path to the file containing the reference genome
OUTPUT_FILE	Path to the file which will contain the VCF file
INDEX_FILE	Path to the file which contain the index of the genome
READS_FILE	Path to the file containing the reads
K_VALUE	K_value chosen by the user
H_VALUE	H_value chosen by the user
M_VALUE	M_value chosen by the user
VERBOSE	True/False if option verbose was used by the user
GENOME FILE	File containing the genome
GENOME	String containing the genome
READS_TEXT	File containing the read
READS_LIST	List of string containing the reads
INDEX	Object containing the FMI of the genome indexed with index.py
SNPS_DICT	Dictionary containing the SNPs found during mapping

FUNCTIONS

SEED_AND_EXTEND(READ, K, H, INDEX, GENOME)

- Function that takes a read, a k_value, a h_value, an FMI index and the genome indexed with it to find the best mapping position for this read in the genome.
- Use a seed & extend method with k-mers of size k.
- Parameters:
  - `read: str`
    - read (sequence of nucleotides) we try to map on the genome.
  - `k: int`
    - size of the k-mers to seed.
  - `h: int`
    - maximum value of mismatch in the mapping found.
  - `index: FMI`
    - FMI object which index the genome using FM-Index method.
  - `genome: str`
    - genome (sequence of nucleotides) indexed by the FMI in which we want to search for the read.
- Returns
  - `position_mapping: int`
    - Index on the genome where the best mapping is obtained with this read. Length of the genome if the read cannot be mapped on the genome with these parameters.
  - `nb_mismatch: int`
    - Number of mismatches in the alignment. h if no mapping found.
  - `list_mismatch: list`
    - List of the position on the genome of the mismatches detected during this alignment. Empty list if no mapping found or no mismatch.

DIAGRAM - seed_and_extend() FUNCTION

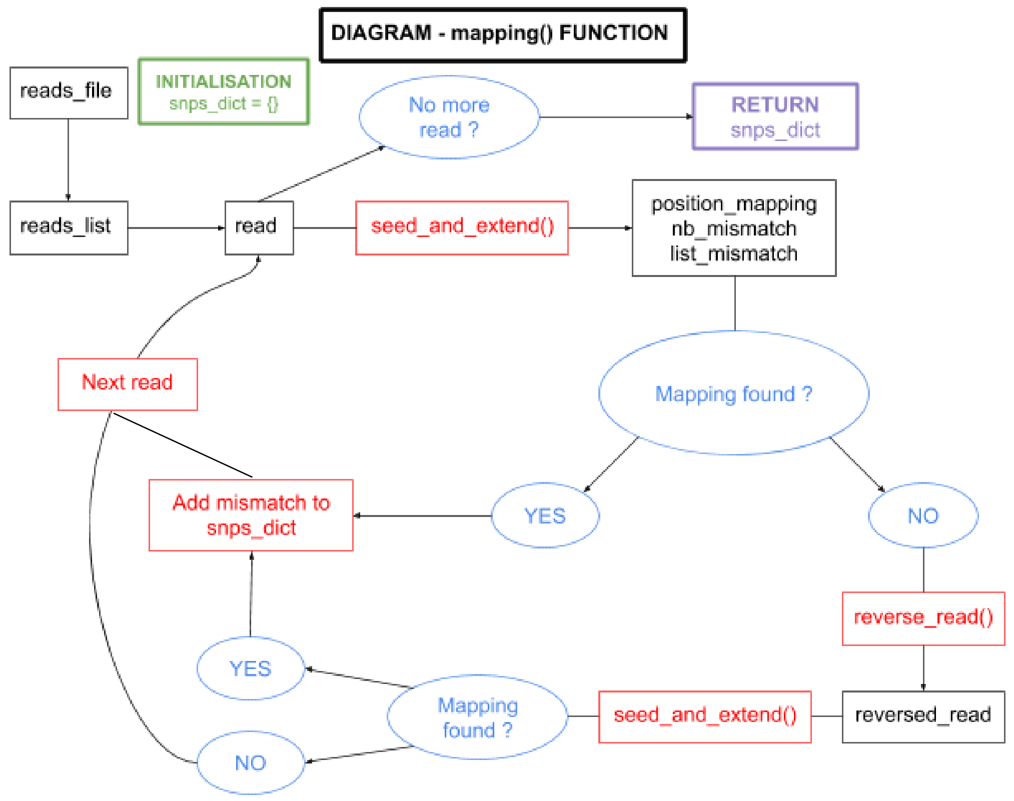

read
INITIALISATION
position_mapping = len(genome)
nb_mismatch = h_value + 1
list_mismatch = []
INITIALISATION
position_mapping_kmer = len(genome)
nb_mismatch_kmer = h_value + 1
list_mismatch_kmer = []
k-mer
Next k-mer
get_occurences()
No more k-mer ?
RETURN
position_mapping
nb_mismatch
list_mismatch
list of occurrences in genome in ascending order
Next occurrence
occu
No more occurrences ?
nb_mismatch_kmer lower than nb_mismatch ?
YES
NO
index of the genome where to try to map the read
position_mapping ⟵ position_mapping_kmer
nb_mismatch ⟵ nb_mismatch_kmer
list_mismatch ⟵ list_mismatch_kmer
index in list_mapping_read ?
YES
NO
Add index to list_mapping_read to not test it again
Alignment while nb_mismatch_occu < h_value
position_mapping_occu
nb_mismatch_occu
list_mismatch_occu
nb_mismatch_occu lower than nb_mismatch_kmer ?
NO
YES
position_mapping_kmer ⟵ position_mapping_occu
nb_mismatch_kmer ⟵ nb_mismatch_occu
list_mismatch_kmer ⟵ list_mismatch_occu

REVERSE_READ(READ)

- Creates the equivalent of the read on the reverse strand. Modify "A" to "T" and inverse and "C" to "G" and inverse.
- Parameters:
  - `read: str`
    - nucleotidic sequence to reverse
- Errors raised:
  - `ValueError`
    - If the read's sequence contains characters different from “A”, “T”, “G” or “C”.
- Returns:
  - `reversed_read: str`
    - nucleotidic sequence reversed

MAPPING(READS_LIST, K, H, INDEX, GENOME)

- Maps the reads stored in `reads_list` on the genome using `seed_and_extend()` function and creates a dictionary which will contains the snps found during mapping :
  - keys will be the index in the genome where the snps was found
  - values will be the list of characters found in reads mapped at this position.
- Parameters:
  - `reads_list: list of str`
    - list of read (sequence of nucleotides) we try to map on the genome.
  - `k: int`
    - size of the k-mers to seed using seed_and_extend().
  - `h: int`
    - maximum value of mismatch in the mapping found.
  - `index: FMI`
    - FMI object which index the genome using FM-Index method.
  - `genome: str`
    - genome (sequence of nucleotides) indexed by the FMI in which we want to search for the read.
- Returns:
  - `snps_dict: Dictionary`
    - dictionary which will contains the snps found during mapping :
      - keys will be the index in the genome where the snps was found
      - values will be the list of characters found in reads mapped at this position.



-

WRITE_VCF(SNPS_DICT)

- Write vcf file containing the list of Single Nucleotid Polymorphisms found during mapping.
- Parameters:
  - `snps_dict:` `Dictionary`
    - dictionary which contains the snps found during mapping :
      - keys are the index in the genome where the snps was found
      - values are the list of characters found in reads mapped at this position.
- Returns:
  - No return but write a VCF file containing the list of Single Nucleotid Polymorphisms found during mapping.

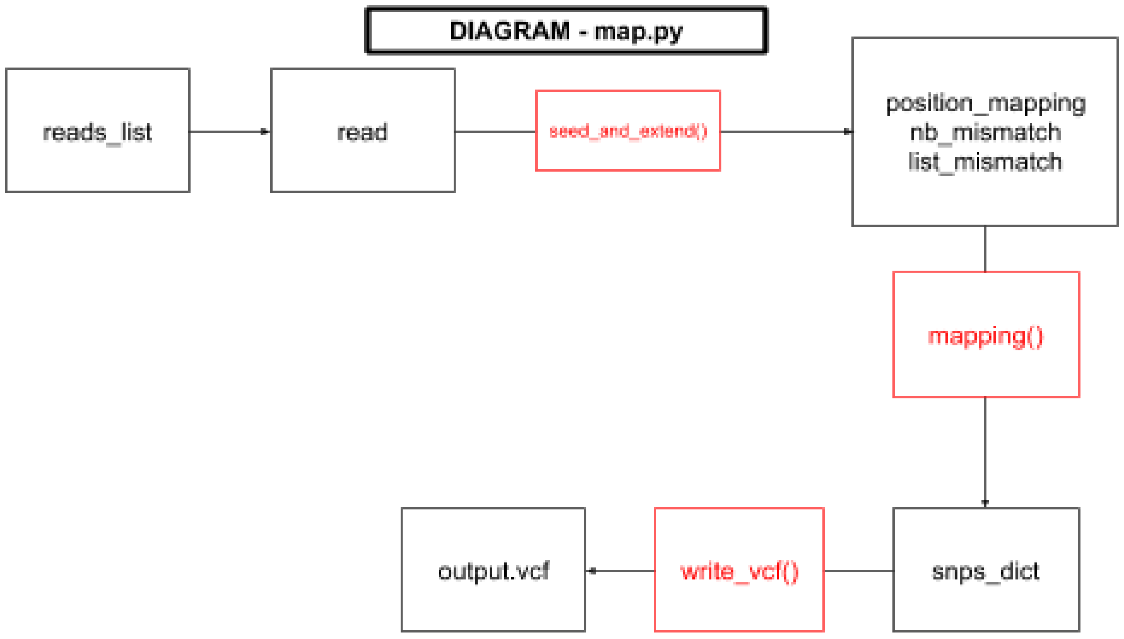
Example of VCF file:

```
#READS: my_reads.fa
#K: 14
#MAX_SUBST: 5
#MIN_ABUNDANCE: 10
129836 A T 26
145831 C G 25
```

OPTIONS

- `--ref` or `-r` + *filename*
    - Indicates the file containing the genome/sequence to index.
    - The file must be in FASTA format : starts with a line explaining the file (will not be used) and the rest of the file contains the sequence.
- `--index` or `-i` + *filename*
    - Indicates the file in which we stored the index.
    - The name must be "name.dp" since the index Object is stored in a dumped file.
- `--reads` or `-r` + *filename*
    - Indicates the file containing the reads to map on the genome.
    - The file must be in FASTA format: for each read, we have a line starting with ">" that names the read and after that a line containing the read.
- `--out` or `-o` + *filename*
    - Indicates the file in which we will store the results of our mapping.
    - The name must be "name.vcf".
- `--k_value` or `-k` + *int*
    - Indicates the value used to cut the read into k-mers of length k.
    - The value must be between 1 and the length of the read.
- `--max_hamming` or `-h` + *int*
    - Indicates the value used to map the read in the genome with a number of substitutions inferior to h.
    - The value must be between 1 and the length of the read.
- `--min_abundance` or `-m` + *int*
    - Indicates the value used to count the Single Nucleotid Polymorphisms.
    - Only SNPs listed more than m times will be recorded.
- `--help` or `-h`
    - To visualise the README notice.
- `--verbose` or `-v`
    - To print the options of the program and the time taken by the program.

USE OF THE PROGRAM



- When this program is launched (`if __name__ == "__main__"`), reads the options used and reads the different files: reference file (option `--ref`), reads_list (option `--read`) et index (option `--index`).
- The reference file must be in FASTA format (see README.md).
- If the reference file is not found, raise an error and exit.