Algorithms in bioinformatics

The Burrows-Wheeler transform and the FM-index

TD/TP

As you progress within the tutorial, you should develop your intuition on the data structure, as well as gaining automy for good practices in algorithmics (proper definition of problems, running time analysis, high-level description of algorithms) and implementations (extensive code documentation with docstrings and comments, robust tests).

The aim of this tutorial is to develop a toy implementation of the FM-index, as a Python class. We start with the Burrows-Wheeler transform and its compression capababilities, and then program the auxiliary arrays needed for fast pattern localisation. A pretty printer is already implemented to help you debugging.

(1) The Burrows-Wheeler transform

(1.1) Construction

▶ **Question 1.** *(✎) Recall the definition of the suffix array, and the definition of the BWT that relies on the latter.*

▶ **Question 2.** *(✎) What is the Burrows-Wheeler transform of "ABRACADABRA"?*

▶ **Question 3.** *(💻) Write a method* `compute_sa` *that initialize the field* `FM_index.sa` *with the suffix array of the input sequence. You will rely on the function* `simple_kark_sort`*, from the file* `ks.py`*. You can run the following code to understand how to use the function.*

```
from ks import simple_kark_sort
s = 'GGCGGCACCGC$'
sa = simple_kark_sort(s)
print('i\tSA\tSuffixes')
for i in range(len(s)): print(f'{i}\t{sa[i]}\t{s[sa[i]:]}')
```

▶ **Question 4.** *(💻) Write a method* `compute_bwt` *that initialize the field* `FM_index.bwt` *with the Burrows-Wheeler transform of the input sequence. You can assume that the field* `FM_index.sa` *is initialized.*

(1.2) Naive inversion

▶ **Question 5.** *(✎) Recall how to reconstruct the Burrows-Wheeler matrix from the transform. What is the complexity of this method ?*

▶ **Question 6.** *(✎) What is the word whose Burrows-Wheeler transform is "IPSSM$PISSII"?*

▶ **Question 7.** *(💻) Write a method* `get_string__naive` *that retrieves the initial sequence from the Burrows-Wheeler transform, by iteratively reconstructing the Burrows-Wheeler matrix. You can assume that the field* `FM_index.bwt` *is initialized.*

(1.3) Compression capabilities

▶ **Question 8.** *(💻) Write a function* `compress_RLE`*, that turns a string into a sequence runs, described by (char, length) couples. Write a function* `decompress_RLE`*, that reconstruct the string from the sequence of couples.*

▶ **Question 9.** *(💻) For simplicity, let consider that the storage requirement of a string is proportional to the length of its RLE sequence. Compare the compression capabilities of the bwt-rle framework on random DNA strings, viral genomes (eg.* `hiv1.fa`*) and repetitive strings (eg.* `dudh.txt`*).*

(2) ■ The FM-index

(2.1) ◆ Construction

▶ **Question 10.** *(✎) Recall the definition of the* Count *map and the* Rank *array that form, together with the BWT, the FM-index. What are their respective values for the sequence "ABRACADABRA"?*

▶ **Question 11.** *(💻) Write a method* `compute_count` *that initialize the field* `FM_index.fm_count` *with the* Count *map of the FM-index. You can assume that the field* `FM_index.sa` *is initialized.*

▶ **Question 12.** *(💻) Write a method* `compute_rank` *that initialize the field* `FM_index.fm_rank` *with the (unique) rank array ()*Rank*) of the FM-index. You can assume that the field* `FM_index.bwt` *is initialized.*

▶ **Question 13.** *(💻* ***BONUS****) Rename the previous methods that should be hidden to a user by prepending their name with underscores. Initialize all the FM-index fields within the* `__init__` *function, logging advancement if the* `verbose` *argument (*`False` *by default) is set to* `True`*.*

(2.2) ◆ Fast BWT inversion

▶ **Question 14.** *(✎) Recall the LF-mapping property of the Burrows-Wheeler transform.*

▶ **Question 15.** *(✎) Show how this mapping can be leveraged to invert the transform "IPSSM$PISSII" directly (ie. without reconstructing the matrix). What is the time complexity of the algorithm you used?*

▶ **Question 16.** *(💻) Write a method* `get_string` *that retrieve the initial sequence from the Burrows-Wheeler transform, by leveraging the LF-mapping property. You can assume that the fields* `FM_index.bwt`*,* `FM_index.fm_count` *and* `FM_index.fm_rank` *are initialized.*

(2.3) ◆ Pattern matching

▶ **Question 17.** *(💻) Write a private method* `__compute_nsl` *that initialize the field* `FM_index.next_smallest_letter` *with the map that link any letter to its lexicographic successor (or* `None` *if no such letter exists). You can assume that the field* `FM_index.count` *is initialized.*

▶ **Question 18.** *(✎) Recall the definition of the map of arrays* Ranks*. What is it for the sequence "ABRACADABRA"?*

▶ **Question 19.** *(💻) Write a method* `__compute_ranks` *that initialize the field* `FM_index.fm_ranks` *with the map of (multiple) rank arrays (*Ranks*) of the FM-index. You can assume that the fields* `FM_index.bwt` *and* `FM_index.rank` *are initialized.*

▶ **Question 20.** *(💻) Write a private method* `__getPatternInterval` *that retrieve F-interval in which start all the occurences of the pattern given as input.*

▶ **Question 21.** *(💻) Relying on* `__getPatternInterval`*, write three public pattern-matching methods* `membership`*,* `count` *and* `locate`*.*