

Bloom Filter Trie

Generated by Doxygen 1.8.9.1

Fri Jan 22 2016 15:17:40

Contents

1	Data Structure Index	1
1.1	Data Structures	1
2	File Index	3
2.1	File List	3
3	Data Structure Documentation	5
3.1	BFT_annotation Struct Reference	5
3.1.1	Detailed Description	5
3.2	BFT_kmer Struct Reference	5
3.2.1	Detailed Description	6
3.2.2	Field Documentation	6
3.2.2.1	kmer	6
3.2.2.2	kmer_comp	6
3.2.2.3	res	6
3.3	BFT_Root Struct Reference	6
3.3.1	Detailed Description	7
3.3.2	Field Documentation	7
3.3.2.1	filenames	7
3.3.2.2	k	7
3.3.2.3	nb_genomes	7
3.3.2.4	threshold_compression	7
4	File Documentation	9
4.1	lib/interface.h File Reference	9
4.1.1	Detailed Description	11
4.1.2	Typedef Documentation	11
4.1.2.1	BFT	11
4.1.2.2	BFT_func_ptr	11
4.1.3	Function Documentation	11
4.1.3.1	create_BFT_annotation	11
4.1.3.2	create_cdbg	11

4.1.3.3	create_empty_kmer	12
4.1.3.4	create_kmer	12
4.1.3.5	extract_kmers_to_disk	12
4.1.3.6	free_BFT_annotation	12
4.1.3.7	free_BFT_kmer	12
4.1.3.8	free_BFT_kmer_content	13
4.1.3.9	free_cdbg	13
4.1.3.10	get_annotation	13
4.1.3.11	get_count_id_genomes	13
4.1.3.12	get_flag_kmer	13
4.1.3.13	get_kmer	14
4.1.3.14	get_list_id_genomes	14
4.1.3.15	get_neighbors	14
4.1.3.16	get_predecessors	14
4.1.3.17	get_successors	15
4.1.3.18	insert_genomes	15
4.1.3.19	intersection_annotations	15
4.1.3.20	is_kmer_in_cdbg	15
4.1.3.21	iterate_over_kmers	16
4.1.3.22	presence_genome	16
4.1.3.23	set_flag_kmer	16
4.1.3.24	set_marking	16
4.1.3.25	set_neighbors_traversal	16
4.1.3.26	sym_difference_annotations	18
4.1.3.27	union_annotations	18
4.1.3.28	unset_marking	18
4.1.3.29	unset_neighbors_traversal	18
4.1.3.30	v_iterate_over_kmers	18
4.1.3.31	write_kmer_ascii_to_disk	19
4.1.3.32	write_kmer_comp_to_disk	19
4.2	lib/snippets.h File Reference	19
4.2.1	Detailed Description	20
4.2.2	Macro Definition Documentation	20
4.2.2.1	V_NOT_VISITED	20
4.2.2.2	V_VISITED	20
4.2.3	Function Documentation	21
4.2.3.1	BFS	21
4.2.3.2	BFS_subgraph	21
4.2.3.3	cdbg_traversal	21
4.2.3.4	DFS	21

4.2.3.5	DFS_subgraph	22
4.2.3.6	extract_core_kmers	23
4.2.3.7	extract_core_simple_paths	23
4.2.3.8	extract_dispensable_kmers	23
4.2.3.9	extract_pangenome_kmers_to_disk	23
4.2.3.10	extract_simple_core_paths_to_disk	24
4.2.3.11	extract_simple_paths	24
4.2.3.12	extract_simple_paths_to_disk	24
4.2.3.13	extract_singleton_kmers	24
4.2.3.14	get_nb_connected_component	25
4.2.3.15	is_in_subgraph	25
4.2.3.16	nb_connected_components	25

Index**27**

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

BFT_annotation	Annotation associated with a BFT_kmer	5
BFT_kmer	K-mer stored in a BFT_Root	5
BFT_Root	Root vertex of a BFT	6

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

lib/ interface.h	Interface file containing all functions to manipulate a BFT	9
lib/ Node.h	??
lib/ snippets.h	Code snippets manipulating BFT	19

Chapter 3

Data Structure Documentation

3.1 BFT_annotation Struct Reference

Annotation associated with a [BFT_kmer](#).

Data Fields

- `uint8_t * annot`
- `uint8_t * annot_ext`
- `uint8_t * annot_cplx`
- `int size_annot`
- `int size_annot_cplx`
- `uint8_t from_BFT`

3.1.1 Detailed Description

Annotation associated with a [BFT_kmer](#).

A [BFT_annotation](#) contains the set of colors associated with a k-mer of a [BFT_Root](#).

The documentation for this struct was generated from the following file:

- `lib/interface.h`

3.2 BFT_kmer Struct Reference

K-mer stored in a [BFT_Root](#).

Data Fields

- `char * kmer`
ASCII null-terminated k-mer.
- `uint8_t * kmer_comp`
2 bits encoded form of [BFT_kmer::kmer](#).
- `resultPresence * res`
Contains information about the location of [BFT_kmer::kmer](#) in a [BFT_Root](#).

3.2.1 Detailed Description

K-mer stored in a [BFT_Root](#).

A [BFT_kmer](#) contains the k-mer in its ASCII form and 2 bits encoded form, as well as information about its location in a [BFT_Root](#).

3.2.2 Field Documentation

3.2.2.1 `char* BFT_kmer::kmer`

ASCII null-terminated k-mer.

3.2.2.2 `uint8_t* BFT_kmer::kmer_comp`

2 bits encoded form of [BFT_kmer::kmer](#).

3.2.2.3 `resultPresence* BFT_kmer::res`

Contains information about the location of [BFT_kmer::kmer](#) in a [BFT_Root](#).

The documentation for this struct was generated from the following file:

- `lib/Node.h`

3.3 BFT_Root Struct Reference

Root vertex of a BFT.

Data Fields

- `char ** filenames`
Inserted genome file names.
- `uint64_t * hash_v`
- `uint16_t ** skip_sp`
- `annotation_array_elem * comp_set_colors`
- `info_per_level * info_per_lvl`
- `annotation_inform * ann_inf`
- `resultPresence * res`
- `int k`
Size of k-mers.
- `int r1`
- `int r2`
- `int nb_genomes`
Number of genomes inserted.
- `int length_comp_set_colors`
- `int threshold_compression`
Color compression is triggered every [BFT_Root::threshold_compression](#) genome inserted.
- `uint8_t compressed`
- `uint8_t marked`
- `Node node`

3.3.1 Detailed Description

Root vertex of a BFT.

A [BFT_Root](#) contains the k-mer size as well as the number and name of the inserted genomes. Other contained structures and variables are for internal use only and must not be modified.

3.3.2 Field Documentation

3.3.2.1 `char** BFT_Root::filenames`

Inserted genome file names.

3.3.2.2 `int BFT_Root::k`

Size of k-mers.

3.3.2.3 `int BFT_Root::nb_genomes`

Number of genomes inserted.

It corresponds to the number of files stored in [BFT_Root::filenames](#).

3.3.2.4 `int BFT_Root::treshold_compression`

Color compression is triggered every [BFT_Root::treshold_compression](#) genome inserted.

The documentation for this struct was generated from the following file:

- `lib/Node.h`

Chapter 4

File Documentation

4.1 lib/interface.h File Reference

Interface file containing all functions to manipulate a BFT.

Data Structures

- struct [BFT_annotation](#)
Annotation associated with a [BFT_kmer](#).

Typedefs

- typedef [BFT_Root](#) BFT
Root vertex of a BFT.
- typedef size_t(* [BFT_func_ptr](#)) ([BFT_kmer](#) *bft_kmer, [BFT](#) *bft, va_list args)
Pointer on function used by [iterate_over_kmers\(\)](#) and [v_iterate_over_kmers\(\)](#).

Functions

- uint8_t **intersection_annots** (const uint8_t a, const uint8_t b)
- uint8_t **union_annots** (const uint8_t a, const uint8_t b)
- uint8_t **sym_difference_annots** (const uint8_t a, const uint8_t b)

Graph functions

These functions manipulate a colored de Bruijn graph stored in a BFT.

- [BFT](#) * [create_cdbg](#) (int k, int threshold_compression)
Function creating a colored de Bruijn graph stored in a BFT.
- void [free_cdbg](#) ([BFT](#) *bft)
Free an allocated colored de Bruijn graph stored in a BFT.

Insertion functions

These functions insert genomes in a colored de Bruijn graph stored in a BFT.

- void [insert_genomes](#) (int nb_files, char **paths, [BFT](#) *bft)
Function inserting genomes (k-mer file) in a BFT.

K-mer functions

These functions manipulate k-mers.

- `BFT_kmer * create_kmer` (const char *kmer, int k)
Function creating a `BFT_kmer` object from a k-mer encoded as an ASCII string (char).*
- `BFT_kmer * create_empty_kmer` ()
Function creating an empty `BFT_kmer` object (all its components are NULL).
- void `free_BFT_kmer` (BFT_kmer *bft_kmer, int nb_bft_kmer)
Function freeing allocated `BFT_kmers`.
- void `free_BFT_kmer_content` (BFT_kmer *bft_kmer, int nb_bft_kmer)
Function freeing the content of allocated `BFT_kmers`.
- `BFT_kmer * get_kmer` (const char *kmer, BFT *bft)
Function searching for a k-mer in a BFT.
- bool `is_kmer_in_cdbg` (BFT_kmer *bft_kmer)
Function testing if a k-mer is in a BFT.
- void `extract_kmers_to_disk` (BFT *bft, char *filename_output, bool compressed_output)
Function extracting the k-mers of a BFT in a file.
- void `write_kmer_ascii_to_disk` (BFT_kmer *bft_kmer, BFT *bft, va_list args)
Function writing an ASCII k-mer in a file.
- void `write_kmer_comp_to_disk` (BFT_kmer *bft_kmer, BFT *bft, va_list args)
Function writing an 2 bits encoded k-mer in a file.

Annotation functions

These functions manipulate annotations (color sets).

- `BFT_annotation * create_BFT_annotation` ()
Function creating an empty `BFT_annotation`.
- void `free_BFT_annotation` (BFT_annotation *bft_annot)
Function freeing a `BFT_annotation`.
- `BFT_annotation * get_annotation` (BFT_kmer *bft_kmer)
Function extracting the annotation (set of colors) associated with a k-mer of a BFT.
- bool `presence_genome` (uint32_t id_genome, BFT_annotation *bft_annot, BFT *bft)
Function testing if a k-mer occurred in a genome.
- `BFT_annotation * intersection_annotations` (BFT *bft, uint32_t nb_annotations,...)
Function computing the intersection of a set of annotations.
- `BFT_annotation * union_annotations` (BFT *bft, uint32_t nb_annotations,...)
Function computing the union of a set of annotations.
- `BFT_annotation * sym_difference_annotations` (BFT *bft, uint32_t nb_annotations,...)
Function computing the symmetric difference of a set of annotations.
- uint32_t * `get_list_id_genomes` (BFT_annotation *bft_annot, BFT *bft)
Function extracting a list of genome identifiers from an annotation.
- uint32_t `get_count_id_genomes` (BFT_annotation *bft_annot, BFT *bft)
Function counting the number of genome identifiers in an annotation.

Marking functions

These functions allow to mark k-mers of a colored de Bruijn graph with flags.

- void `set_marking` (BFT *bft)
Function locking and preparing the graph for vertices marking (no insertion can happen before unlocking).
- void `unset_marking` (BFT *bft)
Function unlocking and the graph locked for vertices marking.
- void `set_flag_kmer` (uint8_t flag, BFT_kmer *bft_kmer, BFT *bft)
Function marking a k-mer of a BFT with a flag.
- uint8_t `get_flag_kmer` (BFT_kmer *bft_kmer, BFT *bft)
Function getting a k-mer of a BFT with a flag.

Traversal functions

These functions allow to traverse a colored de Bruijn graph stored as a BFT.

- void `set_neighbors_traversal` (BFT *bft)
Function locking the graph for traversal.

- void [unset_neighbors_traversal](#) (BFT *bft)
Function unlocking a locked graph for traversal.
- BFT_kmer * [get_neighbors](#) (BFT_kmer *bft_kmer, BFT *bft)
Function extracting the neighbors of a k-mer.
- BFT_kmer * [get_predecessors](#) (BFT_kmer *bft_kmer, BFT *bft)
Function extracting the predecessors of a k-mer.
- BFT_kmer * [get_successors](#) (BFT_kmer *bft_kmer, BFT *bft)
Function extracting the successors of a k-mer.

Iteration functions

These functions iterate over the k-mers of a colored de Bruijn graph stored as a BFT.

- void [iterate_over_kmers](#) (BFT *bft, BFT_func_ptr f,...)
Function iterating over the k-mers of a BFT.
- void [v_iterate_over_kmers](#) (BFT *bft, BFT_func_ptr f, va_list args)
Function iterating over the k-mers of a BFT.

4.1.1 Detailed Description

Interface file containing all functions to manipulate a BFT.

Code snippets using this interface are provided in [snippets.h](#).

4.1.2 Typedef Documentation

4.1.2.1 typedef BFT_Root BFT

Root vertex of a BFT.

A [BFT_Root](#) contains the k-mer size as well as the number and name of the inserted genomes. Other contained structures and variables are for internal use only and must not be modified.

4.1.2.2 typedef size_t(* BFT_func_ptr) (BFT_kmer *bft_kmer, BFT *bft, va_list args)

Pointer on function used by [iterate_over_kmers\(\)](#) and [v_iterate_over_kmers\(\)](#).

Such a function (user written) is called on every k-mer of a BFT.

Parameters

<i>bft_kmer</i>	is a k-mer from a BFT.
<i>bft</i>	is the BFT from which bft_kmer is from.
<i>args</i>	contains all additional parameters given to iterate_over_kmers() / v_iterate_over_kmers() .

Returns

a size_t type object. It can be use to return an unsigned integer or a pointer.

4.1.3 Function Documentation

4.1.3.1 BFT_annotation* create_BFT_annotation () [inline]

Function creating an empty [BFT_annotation](#).

4.1.3.2 BFT* create_cdbg (int k, int threshold_compression)

Function creating a colored de Bruijn graph stored in a BFT.

Parameters

<i>k</i>	is the length of k-mers.
<i>threshold_compression</i> ↔	indicates when the color compression should be triggered (every <i>threshold_compression</i> genome inserted).

Returns

a BFT pointer

4.1.3.3 **BFT_kmer*** `create_empty_kmer ()`

Function creating an empty **BFT_kmer** object (all its components are NULL).

Returns

a **BFT_kmer** pointer.

4.1.3.4 **BFT_kmer*** `create_kmer (const char * kmer, int k)`

Function creating a **BFT_kmer** object from a k-mer encoded as an ASCII string (char*).

Parameters

<i>kmer</i>	is an an ASCII encoded k-mer string (char*).
<i>k</i>	is the k-mer length.

Returns

a **BFT_kmer** pointer.

4.1.3.5 `void extract_kmers_to_disk (BFT * bft, char * filename_output, bool compressed_output)`

Function extracting the k-mers of a BFT in a file.

Parameters

<i>bft</i>	is a BFT containing the k-mers to iterate over.
<i>filename_output</i>	is the name of a file to which the k-mers are written. File is overwritten if it already exists.
<i>compressed_output</i> ↔	is a boolean indicating if the k-mers should be written in their 2 bits form (true) or ASCII form (false).

4.1.3.6 `void free_BFT_annotation (BFT_annotation * bft_annot)`

Function freeing a **BFT_annotation**.

Parameters

<i>bft_annot</i>	is a pointer to the BFT_annotation to free.
------------------	--

4.1.3.7 `void free_BFT_kmer (BFT_kmer * bft_kmer, int nb_bft_kmer)`

Function freeing allocated BFT_kmers.

Parameters

<i>bft_kmer</i>	is a pointer to an array of at least one BFT_kmer .
<i>nb_bft_kmer</i>	is the number of BFT_kmer in <i>bft_kmer</i> .

4.1.3.8 void free_BFT_kmer_content (BFT_kmer * *bft_kmer*, int *nb_bft_kmer*)

Function freeing the content of allocated BFT_kmers.

Parameters

<i>bft_kmer</i>	is a pointer to an array of at least one BFT_kmer .
<i>nb_bft_kmer</i>	is the number of BFT_kmer in <i>bft_kmer</i> .

4.1.3.9 void free_cdbg (BFT * *bft*)

Free an allocated colored de Bruijn graph stored in a BFT.

Parameters

<i>bft</i>	is an allocated BFT.
------------	----------------------

4.1.3.10 BFT_annotation* get_annotation (BFT_kmer * *bft_kmer*)

Function extracting the annotation (set of colors) associated with a k-mer of a BFT.

Parameters

<i>bft_kmer</i>	is a k-mer obtained via search or iteration over a BFT (via get_kmer() for example).
-----------------	--

Returns

a [BFT_annotation](#) pointer.

4.1.3.11 uint32_t get_count_id_genomes (BFT_annotation * *bft_annot*, BFT * *bft*)

Function counting the number of genome identifiers in an annotation.

Parameters

<i>bft_annot</i>	is an annotation.
<i>bft</i>	is a BFT from which the annotation was extracted.

Returns

a count of genome identifiers.

4.1.3.12 uint8_t get_flag_kmer (BFT_kmer * *bft_kmer*, BFT * *bft*)

Function getting a k-mer of a BFT with a flag.

Parameters

<i>bft_kmer</i>	is a k-mer obtained via search/iteration over a BFT for which the function returns the flag.
<i>bft</i>	is a BFT locked for vertices marking.

4.1.3.13 **BFT_kmer*** get_kmer (const char * *kmer*, BFT * *bft*)

Function searching for a k-mer in a BFT.

Parameters

<i>kmer</i>	is an an ASCII encoded k-mer string (char*) to search for in the BFT.
<i>bft</i>	is a BFT in which k-mer is searched

Returns

a [BFT_kmer](#) pointer.

4.1.3.14 uint32_t* get_list_id_genomes (BFT_annotation * *bft_annot*, BFT * *bft*)

Function extracting a list of genome identifiers from an annotation.

Parameters

<i>bft_annot</i>	is an annotation from which the ids must be extracted.
<i>bft</i>	is a BFT from which the annotation was extracted.

Returns

a pointer to an array of genome identifiers (uint32_t). The first element of this array (position 0) indicates how many ids there are in this array. Therefore, the length of the array is array[0] + 1.

4.1.3.15 **BFT_kmer*** get_neighbors (BFT_kmer * *bft_kmer*, BFT * *bft*)

Function extracting the neighbors of a k-mer.

Parameters

<i>bft_kmer</i>	is a k-mer obtained via search/iteration over a BFT.
<i>bft</i>	is a BFT from which was extracted bft_kmer

Returns

a pointer to an array of 8 [BFT_kmer](#): positions 0 to 3 are the possible predecessors and 4 to 7 the possible successors.

4.1.3.16 **BFT_kmer*** get_predecessors (BFT_kmer * *bft_kmer*, BFT * *bft*)

Function extracting the predecessors of a k-mer.

Parameters

<i>bft_kmer</i>	is a k-mer obtained via search/iteration over a BFT.
<i>bft</i>	is a BFT from which was extracted <i>bft_kmer</i>

Returns

a pointer to an array of 4 [BFT_kmer](#) that are the possible predecessors.

4.1.3.17 BFT_kmer* get_successors (BFT_kmer * bft_kmer, BFT * bft)

Function extracting the successors of a k-mer.

Parameters

<i>bft_kmer</i>	is a k-mer obtained via search/iteration over a BFT.
<i>bft</i>	is a BFT from which was extracted <i>bft_kmer</i>

Returns

a pointer to an array of 4 [BFT_kmer](#) that are the possible successors.

4.1.3.18 void insert_genomes (int nb_files, char ** paths, BFT * bft)

Function inserting genomes (k-mer file) in a BFT.

Parameters

<i>nb_files</i>	is the number of files to insert.
<i>paths</i>	is an <i>nb_files</i> size array of strings (char*). Each string is the name of a file (+ eventually its path) to insert.
<i>bft</i>	is a BFT where the genomes are inserted.

4.1.3.19 BFT_annotation* intersection_annotations (BFT * bft, uint32_t nb_annotations, ...)

Function computing the intersection of a set of annotations.

Parameters

<i>bft</i>	is a BFT from which the input annotations are originated.
<i>nb_annotations</i>	indicates how many annotations must be included in the intersection.
...	is a list of <i>nb_annotations</i> BFT_annotation pointers of which the intersection is computed.

Returns

a [BFT_annotation](#) pointer to an annotation which is the intersection of the input annotations.

4.1.3.20 bool is_kmer_in_cdbg (BFT_kmer * bft_kmer)

Function testing if a k-mer is in a BFT.

Parameters

<i>bft_kmer</i>	is a k-mer obtained via search or iteration over a BFT (via get_kmer() for example).
-----------------	--

Returns

a boolean indicating the presence (true) or absence (false) of the k-mer in a BFT.

4.1.3.21 void iterate_over_kmers (BFT * bft, BFT_func_ptr f, ...)

Function iterating over the k-mers of a BFT.

Parameters

<i>bft</i>	is a BFT containing the k-mers to iterate over.
<i>f</i>	is a pointer on function that will be called on each k-mer.
...	are the additional arguments that must be transmitted to f. They can be extracted in f via its parameter of type va_list.

4.1.3.22 bool presence_genome (uint32_t id_genome, BFT_annotation * bft_annot, BFT * bft)

Function testing if a k-mer occurred in a genome.

Parameters

<i>id_genome</i>	is the genome identifier.
<i>bft_annot</i>	is the annotation of the k-mer to test the presence in genome.
<i>bft</i>	is a BFT in which the k-mer is stored.

Returns

a boolean indicating the presence (true) or absence (false) of the k-mer in a the genome.

4.1.3.23 void set_flag_kmer (uint8_t flag, BFT_kmer * bft_kmer, BFT * bft)

Function marking a k-mer of a BFT with a flag.

Parameters

<i>flag</i>	is the mark to add to a k-mer. It can have value 0, 1, 2 or 3.
<i>bft_kmer</i>	is a k-mer obtained via search/iteration over a BFT that must be marked.
<i>bft</i>	is a BFT locked for vertices marking.

4.1.3.24 void set_marking (BFT * bft)

Function locking and preparing the graph for vertices marking (no insertion can happen before unlocking).

By default, all k-mers of the graph are initialized with a 0 flag value.

Parameters

<i>bft</i>	is a BFT to lock and prepare for vertices marking.
------------	--

4.1.3.25 void set_neighbors_traversal (BFT * bft)

Function locking the graph for traversal.

It is not necessary to lock the graph for traversal (no insertion can happen during the locking) but traversing a locked graph is faster than traversing an unlocked graph.

Parameters

<i>bft</i>	is a BFT to lock for traversal.
------------	---------------------------------

4.1.3.26 **BFT_annotation*** sym_difference_annotations (**BFT *** *bft*, **uint32_t** *nb_annotations*, ...)

Function computing the symmetric difference of a set of annotations.

Parameters

<i>bft</i>	is a BFT from which the input annotations are originated.
<i>nb_annotations</i>	indicates how many annotations must be included in the symmetric difference.
...	is a list of <i>nb_annotations</i> BFT_annotation pointers of which the symmetric difference is computed.

Returns

a [BFT_annotation](#) pointer to an annotation which is the symmetric difference of the input annotations.

4.1.3.27 **BFT_annotation*** union_annotations (**BFT *** *bft*, **uint32_t** *nb_annotations*, ...)

Function computing the union of a set of annotations.

Parameters

<i>bft</i>	is a BFT from which the input annotations are originated.
<i>nb_annotations</i>	indicates how many annotations must be included in the union.
...	is a list of <i>nb_annotations</i> BFT_annotation pointers of which the union is computed.

Returns

a [BFT_annotation](#) pointer to an annotation which is the union of the input annotations.

4.1.3.28 **void** unset_marking (**BFT *** *bft*)

Function unlocking and the graph locked for vertices marking.

Parameters

<i>bft</i>	is a BFT locked for vertices marking.
------------	---------------------------------------

4.1.3.29 **void** unset_neighbors_traversal (**BFT *** *bft*)

Function unlocking a locked graph for traversal.

Parameters

<i>bft</i>	is a locked BFT for traversal that must be unlocked.
------------	--

4.1.3.30 **void** v_iterate_over_kmers (**BFT *** *bft*, **BFT_func_ptr** *f*, **va_list** *args*)

Function iterating over the k-mers of a BFT.

This function should be used only when called from a function with a variable number of arguments. If not, you must use [iterate_over_kmers\(\)](#).

Parameters

<i>bft</i>	is a BFT containing the k-mers to iterate over.
<i>f</i>	is a pointer on function that will be called on each k-mer.
<i>args</i>	should contain all additional arguments to pass to <i>f</i> . They can be extracted in <i>f</i> via its parameter of type <i>va_list</i> .

4.1.3.31 void write_kmer_ascii_to_disk (BFT_kmer * bft_kmer, BFT * bft, va_list args)

Function writing an ASCII k-mer in a file.

This function is of type *BFT_func_ptr* and is intended to be a parameter of [iterate_over_kmers\(\)](#) or [v_iterate_over_kmers\(\)](#).

Parameters

<i>bft_kmer</i>	is a k-mer to write to disk.
<i>bft</i>	is a BFT from which <i>bft_kmer</i> was extracted.
<i>args</i>	is a variable list of arguments. It contains a pointer to a file where to write <i>bft_kmer</i> .

4.1.3.32 void write_kmer_comp_to_disk (BFT_kmer * bft_kmer, BFT * bft, va_list args)

Function writing an 2 bits encoded k-mer in a file.

This function is of type *BFT_func_ptr* and is intended to be a parameter of [iterate_over_kmers\(\)](#) or [v_iterate_over_kmers\(\)](#).

Parameters

<i>bft_kmer</i>	is a k-mer to write to disk.
<i>bft</i>	is a BFT from which <i>bft_kmer</i> was extracted.
<i>args</i>	is a variable list of arguments. It contains a pointer to a file where to write <i>bft_kmer</i> .

4.2 lib/snippets.h File Reference

Code snippets manipulating BFT.

Macros

- `#define V_NOT_VISITED 0`
Flag for marked vertices indicating the vertex has not been visited before.
- `#define V_VISITED 1`
Flag for marked vertices indicating the vertex has been visited before.

Functions

K-mer extraction functions

These functions extract k-mers stored in a BFT to disk.

*The last argument of [extract_pangenome_kmers_to_disk\(\)](#) is of type *BFT_func_ptr*, you can use [extract_core_kmers\(\)](#), [extract_dispensable_kmers\(\)](#) or [extract_singleton_kmers\(\)](#) to extract core, dispensable or singleton k-mers.*

- `size_t extract_core_kmers (BFT_kmer *kmer, BFT *graph, va_list args)`
*Function of type *BFT_func_ptr* extracting a core k-mer to disk.*

- `size_t extract_dispensable_kmers (BFT_kmer *kmer, BFT *graph, va_list args)`
Function of type `BFT_func_ptr` extracting a dispensable k-mer to disk.
- `size_t extract_singleton_kmers (BFT_kmer *kmer, BFT *graph, va_list args)`
Function of type `BFT_func_ptr` extracting a singleton k-mer to disk.
- `void extract_pangenome_kmers_to_disk (BFT *graph, char *filename_output, BFT_func_ptr f)`
Function extracting pan-genome (core/dispensable/singleton) k-mers from a BFT to disk.

Path extraction functions

These functions extract simple (non branching) paths of k-mers stored in a BFT to disk.

- `size_t extract_simple_paths (BFT_kmer *kmer, BFT *graph, va_list args)`
Function of type `BFT_func_ptr` extracting from a non-branching k-mer the simple (non branching) path in which it is.
- `void extract_simple_paths_to_disk (BFT *graph, char *filename_output)`
Function extracting from a colored de Bruijn graph stored as a BFT all simple (non branching) paths.
- `size_t extract_core_simple_paths (BFT_kmer *kmer, BFT *graph, va_list args)`
Function of type `BFT_func_ptr` extracting from a non-branching core k-mer the simple (non branching) path in which it is.
- `void extract_simple_core_paths_to_disk (BFT *graph, double core_ratio, char *filename_output)`
Function extracting from a colored de Bruijn graph stored as a BFT all simple (non branching) core paths.

Graph traversal functions

These functions iterate over a colored de Bruijn graph stored as a BFT.

- `size_t BFS (BFT_kmer *kmer, BFT *graph, va_list args)`
Function of type `BFT_func_ptr` starting a Breadth-First Search traversal from a k-mer.
- `size_t BFS_subgraph (BFT_kmer *kmer, BFT *graph, va_list args)`
Function of type `BFT_func_ptr` starting a Breadth-First Search traversal from a k-mer that is part of a subgraph.
- `size_t DFS (BFT_kmer *kmer, BFT *graph, va_list args)`
Function of type `BFT_func_ptr` starting a Depth-First Search traversal from a k-mer.
- `size_t DFS_subgraph (BFT_kmer *kmer, BFT *graph, va_list args)`
Function of type `BFT_func_ptr` starting a Depth-First Search traversal from a k-mer that is part of a subgraph.
- `bool is_in_subgraph (BFT_kmer *kmer, BFT *graph, int nb_id_genomes, const va_list args)`
Function computing if a k-mer is part of a subgraph.
- `void cdbg_traversal (BFT *graph, BFT_func_ptr f,...)`
Function traversing a colored de Bruijn graph stored as a BFT.
- `size_t nb_connected_components (BFT_kmer *kmer, BFT *graph, va_list args)`
Function of type `BFT_func_ptr` calling a traversal method (DFS/BFS, DFS_subgraph/BFS_subgraph) on a k-mer to determine if it is in a new connected component.
- `void get_nb_connected_component (BFT *graph,...)`
Compute the number of connected components in a colored de-Bruijn graph.

4.2.1 Detailed Description

Code snippets manipulating BFT.

The purpose of this file is to give examples of how to use the functions of the BFT API ([interface.h](#)).

4.2.2 Macro Definition Documentation

4.2.2.1 #define V_NOT_VISITED 0

Flag for marked vertices indicating the vertex has not been visited before.

4.2.2.2 #define V_VISITED 1

Flag for marked vertices indicating the vertex has been visited before.

4.2.3 Function Documentation

4.2.3.1 `size_t BFS (BFT_kmer * kmer, BFT * graph, va_list args)`

Function of type `BFT_func_ptr` starting a Breadth-First Search traversal from a k-mer.

Parameters

<i>kmer</i>	is a k-mer from the BFT graph.
<i>graph</i>	is the BFT from which <i>kmer</i> is from.
<i>args</i>	contains additional parameters given to the calling function (here none).

Returns

true if it is a new connected component, else false.

4.2.3.2 `size_t BFS_subgraph (BFT_kmer * kmer, BFT * graph, va_list args)`

Function of type `BFT_func_ptr` starting a Breadth-First Search traversal from a k-mer that is part of a subgraph.

Parameters

<i>kmer</i>	is a k-mer from the BFT graph.
<i>graph</i>	is the BFT from which <i>kmer</i> is from.
<i>args</i>	contains additional parameters given to the calling function. Here, it contains a number of genome ids, followed by the genome ids, that a k-mer must contain to be considered part of the subgraph.

Returns

true if it is a new connected component, else false.

4.2.3.3 `void cdbg_traversal (BFT * graph, BFT_func_ptr f, ...)`

Function traversing a colored de Bruijn graph stored as a BFT.

Parameters

<i>graph</i>	is a BFT representing a colored de Bruijn graph.
<i>f</i>	is the traversal function (DFS/BFS, DFS_subgraph/BFS_subgraph) to use.
...	is the additional arguments to transfer to <i>f</i> () (if there are some).

4.2.3.4 `size_t DFS (BFT_kmer * kmer, BFT * graph, va_list args)`

Function of type `BFT_func_ptr` starting a Depth-First Search traversal from a k-mer.

Parameters

<i>kmer</i>	is a k-mer from the BFT graph.
<i>graph</i>	is the BFT from which <i>kmer</i> is from.
<i>args</i>	contains additional parameters given to the calling function (here none).

Returns

true if it is a new connected component, else false.

4.2.3.5 `size_t DFS_subgraph (BFT_kmer * kmer, BFT * graph, va_list args)`

Function of type `BFT_func_ptr` starting a Depth-First Search traversal from a k-mer that is part of a subgraph.

Parameters

<i>kmer</i>	is a k-mer from the BFT graph.
<i>graph</i>	is the BFT from which kmer is from.
<i>args</i>	contains additional parameters given to the calling function. Here, it contains a number of genome ids, followed by the genome ids, that a k-mer must contain to be considered part of the subgraph.

Returns

true if it is a new connected component, else false.

4.2.3.6 `size_t extract_core_kmers (BFT_kmer * kmer, BFT * graph, va_list args)`

Function of type BFT_func_ptr extracting a core k-mer to disk.

A core k-mer contains in its annotation all genome ids inserted in the graph.

Parameters

<i>kmer</i>	is a k-mer from the BFT graph.
<i>graph</i>	is the BFT from which kmer is from.
<i>args</i>	contains all additional parameters given to extract_pangenome_kmers_to_disk() : A pointer to a file where to write the k-mer and a pointer to the current number of k-mers written.

4.2.3.7 `size_t extract_core_simple_paths (BFT_kmer * kmer, BFT * graph, va_list args)`

Function of type BFT_func_ptr extracting from a non-branching core k-mer the simple (non branching) path in which it is.

args contains as argument a core_ratio float (between 0 to 1) indicating the ratio of genome ids (compared to the total number of genome ids) that a k-mer annotation must contain to be considered being part of a core simple path.

Parameters

<i>kmer</i>	is a k-mer from the BFT graph.
<i>graph</i>	is the BFT from which kmer is from.
<i>args</i>	contains all additional parameters given to extract_simple_core_paths_to_disk() : A pointer to a file where to write paths and a pointer to the current max. size of a path written in the file.

4.2.3.8 `size_t extract_dispensable_kmers (BFT_kmer * kmer, BFT * graph, va_list args)`

Function of type BFT_func_ptr extracting a dispensable k-mer to disk.

A dispensable k-mer contains in its annotation less than all genome ids inserted in the graph.

Parameters

<i>kmer</i>	is a k-mer from the BFT graph.
<i>graph</i>	is the BFT from which kmer is from.
<i>args</i>	contains all additional parameters given to extract_pangenome_kmers_to_disk() : A pointer to a file where to write the k-mer and a pointer to the current number of k-mers written.

4.2.3.9 `void extract_pangenome_kmers_to_disk (BFT * graph, char * filename_output, BFT_func_ptr f)`

Function extracting pan-genome (core/dispensable/singleton) k-mers from a BFT to disk.

A core k-mer contains in its annotation all genome ids inserted in the graph. A dispensable k-mer contains in its annotation less than all genome ids inserted in the graph. A singleton k-mer contains in its annotation one genome id inserted in the graph.

Parameters

<i>graph</i>	is a BFT from which k-mers must be extracted.
<i>filename_output</i>	is the name of the file where to write the k-mers.
<i>f</i>	is a function of type BFT_func_ptr (like extract_core_kmers() , extract_dispensable_kmers() , and extract_singleton_kmers()) which write a pan-genome k-mer to disk.

4.2.3.10 void extract_simple_core_paths_to_disk (BFT * *graph*, double *core_ratio*, char * *filename_output*)

Function extracting from a colored de Bruijn graph stored as a BFT all simple (non branching) core paths.

Parameters

<i>graph</i>	is a colored de Bruijn graph stored as a BFT.
<i>core_ratio</i>	is a float (between 0 to 1) indicating the ratio of genome ids (compared to the total number of genome ids inserted in graph) that a k-mer annotation must contain to be considered being part of a core simple path.
<i>filename_output</i>	is the name of the file where to write the simple core paths.

4.2.3.11 size_t extract_simple_paths (BFT_kmer * *kmer*, BFT * *graph*, va_list *args*)

Function of type BFT_func_ptr extracting from a non-branching k-mer the simple (non branching) path in which it is.

Parameters

<i>kmer</i>	is a k-mer from the BFT graph.
<i>graph</i>	is the BFT from which kmer is from.
<i>args</i>	contains all additional parameters given to extract_simple_paths_to_disk() : A pointer to a file where to write paths and a pointer to the current max. size of a path written in the file.

4.2.3.12 void extract_simple_paths_to_disk (BFT * *graph*, char * *filename_output*)

Function extracting from a colored de Bruijn graph stored as a BFT all simple (non branching) paths.

Parameters

<i>graph</i>	is a colored de Bruijn graph stored as a BFT.
<i>filename_output</i>	is the name of the file where to write the simple paths.

4.2.3.13 size_t extract_singleton_kmers (BFT_kmer * *kmer*, BFT * *graph*, va_list *args*)

Function of type BFT_func_ptr extracting a singleton k-mer to disk.

Singleton k-mer contains in its annotation one genome id inserted in the graph.

Parameters

<i>kmer</i>	is a k-mer from the BFT graph.
<i>graph</i>	is the BFT from which kmer is from.

<i>args</i>	contains all additional parameters given to extract_pangenome_kmers_to_disk() : A pointer to a file where to write the k-mer and a pointer to the current number of k-mers written.
-------------	---

4.2.3.14 void get_nb_connected_component (BFT * graph, ...)

Compute the number of connected components in a colored de-Bruijn graph.

Parameters

<i>graph</i>	is a BFT representing a colored de Bruijn graph.
...	is the additional arguments to transfer to nb_connected_components() (traversal method and additional arguments if there are some).

4.2.3.15 bool is_in_subgraph (BFT_kmer * kmer, BFT * graph, int nb_id_genomes, const va_list args)

Function computing if a k-mer is part of a subgraph.

A subgraph is determined by k-mers having in their annotation specific genome ids.

Parameters

<i>kmer</i>	is a k-mer from the BFT graph.
<i>graph</i>	is the BFT from which kmer is from.
<i>nb_id_genomes</i>	is the number of genome ids that a k-mer must contain to be considered part of the subgraph.
<i>args</i>	contains additional parameters given to the calling function. Here, it contains the genome ids that a k-mer must contain to be considered part of the subgraph.

Returns

true if it is a new connected component, else false.

4.2.3.16 size_t nb_connected_components (BFT_kmer * kmer, BFT * graph, va_list args)

Function of type BFT_func_ptr calling a traversal method (DFS/BFS, DFS_subgraph/BFS_subgraph) on a k-mer to determine if it is in a new connected component.

Parameters

<i>kmer</i>	is a k-mer from the BFT graph.
<i>graph</i>	is the BFT from which kmer is from.
<i>args</i>	contains additional parameters given to the calling function. Here, it contains the traversal method to call and its additional arguments.

Index

BFS
 snippets.h, [21](#)

BFS_subgraph
 snippets.h, [21](#)

BFT
 interface.h, [11](#)

BFT_Root, [6](#)
 filenames, [7](#)
 k, [7](#)
 nb_genomes, [7](#)
 threshold_compression, [7](#)

BFT_annotation, [5](#)

BFT_func_ptr
 interface.h, [11](#)

BFT_kmer, [5](#)
 kmer, [6](#)
 kmer_comp, [6](#)
 res, [6](#)

cdbg_traversal
 snippets.h, [21](#)

create_BFT_annotation
 interface.h, [11](#)

create_cdbg
 interface.h, [11](#)

create_empty_kmer
 interface.h, [12](#)

create_kmer
 interface.h, [12](#)

DFS
 snippets.h, [21](#)

DFS_subgraph
 snippets.h, [21](#)

extract_core_kmers
 snippets.h, [23](#)

extract_core_simple_paths
 snippets.h, [23](#)

extract_dispensable_kmers
 snippets.h, [23](#)

extract_kmers_to_disk
 interface.h, [12](#)

extract_pangenome_kmers_to_disk
 snippets.h, [23](#)

extract_simple_core_paths_to_disk
 snippets.h, [24](#)

extract_simple_paths
 snippets.h, [24](#)

extract_simple_paths_to_disk
 snippets.h, [24](#)

extract_singleton_kmers
 snippets.h, [24](#)

filenames
 BFT_Root, [7](#)

free_BFT_annotation
 interface.h, [12](#)

free_BFT_kmer
 interface.h, [12](#)

free_BFT_kmer_content
 interface.h, [13](#)

free_cdbg
 interface.h, [13](#)

get_annotation
 interface.h, [13](#)

get_count_id_genomes
 interface.h, [13](#)

get_flag_kmer
 interface.h, [13](#)

get_kmer
 interface.h, [14](#)

get_list_id_genomes
 interface.h, [14](#)

get_nb_connected_component
 snippets.h, [25](#)

get_neighbors
 interface.h, [14](#)

get_predecessors
 interface.h, [14](#)

get_successors
 interface.h, [15](#)

insert_genomes
 interface.h, [15](#)

interface.h
 BFT, [11](#)
 BFT_func_ptr, [11](#)
 create_BFT_annotation, [11](#)
 create_cdbg, [11](#)
 create_empty_kmer, [12](#)
 create_kmer, [12](#)
 extract_kmers_to_disk, [12](#)
 free_BFT_annotation, [12](#)
 free_BFT_kmer, [12](#)
 free_BFT_kmer_content, [13](#)
 free_cdbg, [13](#)
 get_annotation, [13](#)
 get_count_id_genomes, [13](#)

- get_flag_kmer, 13
- get_kmer, 14
- get_list_id_genomes, 14
- get_neighbors, 14
- get_predecessors, 14
- get_successors, 15
- insert_genomes, 15
- intersection_annotations, 15
- is_kmer_in_cdbg, 15
- iterate_over_kmers, 16
- presence_genome, 16
- set_flag_kmer, 16
- set_marking, 16
- set_neighbors_traversal, 16
- sym_difference_annotations, 18
- union_annotations, 18
- unset_marking, 18
- unset_neighbors_traversal, 18
- v_iterate_over_kmers, 18
- write_kmer_ascii_to_disk, 19
- write_kmer_comp_to_disk, 19
- intersection_annotations
 - interface.h, 15
- is_in_subgraph
 - snippets.h, 25
- is_kmer_in_cdbg
 - interface.h, 15
- iterate_over_kmers
 - interface.h, 16
- k
 - BFT_Root, 7
- kmer
 - BFT_kmer, 6
- kmer_comp
 - BFT_kmer, 6
- lib/interface.h, 9
- lib/snippets.h, 19
- nb_connected_components
 - snippets.h, 25
- nb_genomes
 - BFT_Root, 7
- presence_genome
 - interface.h, 16
- res
 - BFT_kmer, 6
- set_flag_kmer
 - interface.h, 16
- set_marking
 - interface.h, 16
- set_neighbors_traversal
 - interface.h, 16
- snippets.h
 - BFS, 21
 - BFS_subgraph, 21
 - cdbg_traversal, 21
 - DFS, 21
 - DFS_subgraph, 21
 - extract_core_kmers, 23
 - extract_core_simple_paths, 23
 - extract_dispensable_kmers, 23
 - extract_pangenome_kmers_to_disk, 23
 - extract_simple_core_paths_to_disk, 24
 - extract_simple_paths, 24
 - extract_simple_paths_to_disk, 24
 - extract_singleton_kmers, 24
 - get_nb_connected_component, 25
 - is_in_subgraph, 25
 - nb_connected_components, 25
 - V_NOT_VISITED, 20
 - V_VISITED, 20
 - sym_difference_annotations
 - interface.h, 18
- treshold_compression
 - BFT_Root, 7
- union_annotations
 - interface.h, 18
- unset_marking
 - interface.h, 18
- unset_neighbors_traversal
 - interface.h, 18
- V_NOT_VISITED
 - snippets.h, 20
- V_VISITED
 - snippets.h, 20
- v_iterate_over_kmers
 - interface.h, 18
- write_kmer_ascii_to_disk
 - interface.h, 19
- write_kmer_comp_to_disk
 - interface.h, 19