Bloom Filter Trie

Generated by Doxygen 1.8.9.1

Fri Jan 22 2016 15:17:40

Contents

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 2

File Index

2.1 File List

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

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 treshold_compression
  Color compression is triggered every BFT_Root::treshold_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 treshold_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 occured 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

bft_kmer	is a k-mer from a BFT.
bft	is the BFT from which bft_kmer is from.
args	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 *treshold_compression*)

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

Parameters

k	is the length of k-mers.
treshold_compression	indicates when the color compression should be triggered (every treshold_compression 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

kmer	is an an ASCII encoded k-mer string (char*).
k	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

bft	is a BFT containing the k-mers to iterate over.
filename_output	is the name of a file to which the k-mers are written. File is overwritten if it already exists.
compressed_output	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

bft_annot	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

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

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

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

4.1.3.9 void free_cdbg (BFT * *bft*)

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

Parameters

bft	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

bft_kmer	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

bft_annot	is an annotation.
bft	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

bft_kmer	is a k-mer obtained via search/iteration over a BFT for which the function returns the flag.
bft	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

kmer	is an an ASCII encoded k-mer string (char*) to search for in the BFT.
bft	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

bft_annot	is an annotation from which the ids must be extracted.
bft	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

bft_kmer	is a k-mer obtained via search/iteration over a BFT.
bft	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

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

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

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

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

nb_files	is the number of files to insert.
paths	is an nb_files size array of strings (char*). Each string is the name of a file (+ eventually its path) to insert.
bft	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

bft	is a BFT from which the input annotations are originated.
nb_annotations	indicates how many annotations must be included in the intersection.
...	is a list of nb_annotations 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

bft_kmer	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

bft	is a BFT containing the k-mers to iterate over.
f	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 occured in a genome.

Parameters

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

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

bft	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

bft	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

bft	is a BFT from which the input annotations are originated.
nb_annotations	indicates how many annotations must be included in the symmetric difference.
...	is a list of nb_annotations 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

bft	is a BFT from which the input annotations are originated.
nb_annotations	indicates how many annotations must be included in the union.
...	is a list of nb_annotations 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

bft	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

bft	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

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

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

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

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

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

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

kmer	is a k-mer from the BFT graph.
graph	is the BFT from which kmer is from.
args	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

kmer	is a k-mer from the BFT graph.
graph	is the BFT from which kmer is from.
args	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

graph	is a BFT representing a colored de Bruijn graph.
f	is the traversal function (DFS/BFS, DFS_subgraph/BFS_subgraph) to use.
...	is the additional arguments to transfer to f() (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

kmer	is a k-mer from the BFT graph.
graph	is the BFT from which kmer is from.
args	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

kmer	is a k-mer from the BFT graph.
graph	is the BFT from which kmer is from.
args	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

kmer	is a k-mer from the BFT graph.
graph	is the BFT from which kmer is from.
args	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

kmer	is a k-mer from the BFT graph.
graph	is the BFT from which kmer is from.
args	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

kmer	is a k-mer from the BFT graph.
graph	is the BFT from which kmer is from.
args	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

graph	is a BFT from which k-mers must be extracted.
filename_output	is the name of the file where to write the k-mers.
f	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

graph	is a colored de Bruijn graph stored as a BFT.
core_ratio	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.
filename_output	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

kmer	is a k-mer from the BFT graph.
graph	is the BFT from which kmer is from.
args	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

graph	is a colored de Bruijn graph stored as a BFT.
filename_output	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

kmer	is a k-mer from the BFT graph.
graph	is the BFT from which kmer is from.

args	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

graph	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

kmer	is a k-mer from the BFT graph.
graph	is the BFT from which kmer is from.
nb_id_genomes	is the number of genome ids that a k-mer must contain to be considered part of the subgraph.
args	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

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

Index