RNARedPrint

Generated by Doxygen 1.8.13

Contents

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 2

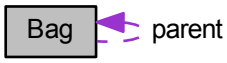
Class Documentation

2.1 Bag Class Reference

The Bag class represents a set of indices, ie positions in the sequences that must be jointly considered in the context of a given tree-decomposition.

`#include <TreeDecomposition.hpp>`

Collaboration diagram for Bag:



Public Member Functions

- Bag (int i)

  Bag Constructs a bag with given identifier.
- void addIndex (int i)

  addIndex Adds an index/position to this bag
- vector< int > getIndices ()

  getIndices Returns the set of indices/positions for this bag
- void orderIndices ()

  orderIndices Reorders the indices such that the proper index is put at the last position in the index list.
- void addChild (Bag *b)

  addChild Adds a child to the bag
- void replaceChild (Bag *prev, Bag *next)

  replaceChild Replaces any occurrence of a bag in the list of children bags with another bag
- void setParent (Bag *b)

  setParent Sets the parent of this bag to a given bag
- int numProper ()

numProper Returns the number of proper indices for this bag

- vector< int > getProperIndices ()

  getProperIndices Returns the list of proper indices for this bag

- vector< Bag * > getChildren ()

  getChildren Returns the list of children bags for this bag

- vector< int > getProperParentIndices ()

  getProperParentIndices Returns the list of indices that are proper to the parent, ie indices that are in the parent list but not in this child list

- int numProperParentIndices ()

  numProperParentIndices Returns the number of indices that are proper to the parent, ie indices that are in the parent list but not in this child list

- const vector< BasePair * > & getProperBasePairs ()

  getProperBasePairs Returns the set of base pairs that are proper to this bag

- int width ()

  width Returns the width of this bag

- int getId ()

  getId Returns the unique identifier for this bag

- void addBasePair (BasePair *bp)

  addBasePair Add a base pair to this bag

- void topologicalSort (vector< Bag *> &result)

  topologicalSort Sorts and returns the bags in the tree initiated at this node

Public Attributes

- vector< Bag * > **children**
- Bag * **parent**

2.1.1 Detailed Description

The Bag class represents a set of indices, ie positions in the sequences that must be jointly considered in the context of a given tree-decomposition.

2.1.2 Constructor & Destructor Documentation

2.1.2.1 Bag()

```
Bag::Bag (
            int i )
```

Bag Constructs a bag with given identifier.

Parameters

i	Unique identifier for the Bag

2.1.3 Member Function Documentation

2.1.3.1 addBasePair()

```
void Bag::addBasePair (
            BasePair * bp )
```

addBasePair Add a base pair to this bag

Parameters

bp	Pointer to base pair to be added

2.1.3.2 addChild()

```
void Bag::addChild (
            Bag * b )
```

addChild Adds a child to the bag

Parameters

b	Pointer to a bag to add as a child to this bag

Adds a child to the bag, considered as an internal node in the tree-decomposition. The parent of the child bag is set to the current bag.

2.1.3.3 addIndex()

```
void Bag::addIndex (
            int i )
```

addIndex Adds an index/position to this bag

Parameters

i	Index/position to add to the bag

2.1.3.4 getChildren()

```
vector< Bag * > Bag::getChildren ( )
```

getChildren Returns the list of children bags for this bag

Returns

List of children bags for this bag

2.1.3.5 getId()

```
int Bag::getId ( )
```

getId Returns the unique identifier for this bag

Returns

Unique identifier for this bag

2.1.3.6 getIndices()

```
vector< int > Bag::getIndices ( )
```

getIndices Returns the set of indices/positions for this bag

Returns

2.1.3.7 getProperBasePairs()

```
const vector< BasePair * > & Bag::getProperBasePairs ( )
```

getProperBasePairs Returns the set of base pairs that are proper to this bag

Returns

Set of base pairs that are proper to this bag

The proper base pairs for a bag are the base pairs such that: a) Both 5' and 3' end of the base pair are in the list of indices; b) Either the 5' or the 3' end of the base pair is proper to the current bag. Note that this definition uniquely defines to which bag a base pair must be attributed in a given tree decomposition (assuming that the base-pair is materialized by some edge in the graph used for the TD).

2.1.3.8 getProperIndices()

```
vector< int > Bag::getProperIndices ( )
```

getProperIndices Returns the list of proper indices for this bag

Returns

List of proper indices for this bag, ie indices that are in this bag but not in the parent bag

2.1.3.9 getProperParentIndices()

```
vector< int > Bag::getProperParentIndices ( )
```

getProperParentIndices Returns the list of indices that are proper to the parent, ie indices that are in the parent list but not in this child list

Returns

list of indices that are proper to the parent, ie indices that are in the parent list but not in this child list

2.1.3.10 numProper()

```
int Bag::numProper ( )
```

numProper Returns the number of proper indices for this bag

Returns

Number of proper indices, ie indices that are in this bag but not in the parent bag

2.1.3.11 numProperParentIndices()

```
int Bag::numProperParentIndices ( )
```

numProperParentIndices Returns the number of indices that are proper to the parent, ie indices that are in the parent list but not in this child list

Returns

Number of indices that are proper to the parent, ie indices that are in the parent list but not in this child list

2.1.3.12 orderIndices()

```
void Bag::orderIndices ( )
```

orderIndices Reorders the indices such that the proper index is put at the last position in the index list.

Checks that there is exactly one proper index per bag. A proper index is an index which is found in the child bag, but not in the parent bag. Puts it at the end of the index/position list for conveniency of future accesses.

2.1.3.13 replaceChild()

```
void Bag::replaceChild (
            Bag * prev,
            Bag * next )
```

replaceChild Replaces any occurrence of a bag in the list of children bags with another bag

Parameters

prev	Pointer to current child bag to be replaced
next	Pointer to replacement child bag

2.1.3.14 setParent()

```
void Bag::setParent (
            Bag * b )
```

setParent Sets the parent of this bag to a given bag

Parameters

b	Pointer to bag, set as parent to this bag

2.1.3.15 topologicalSort()

```
void Bag::topologicalSort (
            vector< Bag *> & result )
```

topologicalSort Sorts and returns the bags in the tree initiated at this node

Parameters

result	List of bags such that the leaves can be found first and, more generally, children can be found before the parents.

2.1.3.16 width()

```
int Bag::width ( )
```

width Returns the width of this bag

Returns

Width, ie number of indices, of this bag

The documentation for this class was generated from the following files:

- src/TreeDecomposition.hpp
- src/TreeDecomposition.cpp

2.2 BasePair Class Reference

The BasePair class encodes a basic base pair.

```
#include <RNAStructure.hpp>
```

Public Member Functions

- BasePair (int a, int b)
  BasePair Creates a base pair.

Public Attributes

- int **i**
- int **j**

2.2.1 Detailed Description

The BasePair class encodes a basic base pair.

2.2.2 Constructor & Destructor Documentation

2.2.2.1 BasePair()

```
BasePair::BasePair (
            int a,
            int b )
```

BasePair Creates a base pair.

Parameters

a	5' position of base pair
b	3' position of base pair

The documentation for this class was generated from the following files:

- src/RNAStructure.hpp
- src/RNAStructure.cpp

2.3 LabeledBasePair Class Reference

The LabeledBasePair class represents base pairs that are labeled (eg to indicate which secondary structure they come from)

`#include <RNAStructure.hpp>`

Public Member Functions

- LabeledBasePair (int a, int b, int x)

  LabeledBasePair Constructs a labeled base pair.

Public Attributes

- int **i**
- int **j**
- int **id**

2.3.1 Detailed Description

The LabeledBasePair class represents base pairs that are labeled (eg to indicate which secondary structure they come from)

2.3.2 Constructor & Destructor Documentation

2.3.2.1 LabeledBasePair()

```
LabeledBasePair::LabeledBasePair (
            int a,
            int b,
            int x )
```

LabeledBasePair Constructs a labeled base pair.

Parameters

a	5' position of the base pair
b	3' position of the base pair
x	Integer-valued labeled

The documentation for this class was generated from the following files:

- src/RNAStructure.hpp
- src/RNAStructure.cpp

2.4 Loop Class Reference

The Loop class represents a loop in the Turner model, ie a set of indices.

```
#include <RNAStructure.hpp>
```

Public Member Functions

- Loop (int a, int b, vector< int > ss)

  Loop Constructs a loop.

Public Attributes

- int **i**
- int **j**
- vector< BasePair * > **basePairs**

2.4.1 Detailed Description

The Loop class represents a loop in the Turner model, ie a set of indices.

2.4.2 Constructor & Destructor Documentation

2.4.2.1 Loop()

```
Loop::Loop (
            int a,
            int b,
            vector< int > ss )
```

Loop Constructs a loop.

Parameters

a	5' position of the enclosing base pair (-1 if exterior face)
b	3' position of the enclosing base pair (-1 if exterior face)
ss	Set of positions involved in the loop

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

- src/RNAStructure.hpp

2.5 SecondaryStructure Class Reference

The SecondaryStructure class represents an RNA secondary structure, possibly with pseudoknots.

```
#include <RNAStructure.hpp>
```

Public Member Functions

- SecondaryStructure (string s, int idd)

  SecondaryStructure Constructs a secondary structure having given id from a dot-bracket representation.
- vector< int > getSS ()

  getSS Returns a representation of the secondary structure as an array of base pairing partners
- vector< LabeledBasePair * > getLabelledBPs ()

  getLabelledBPs Returns a set of base pairs decorated by the structure identifier
- int getLength ()

  getLength Returns the number of nucleotides involved in the secondary structure
- vector< BasePair * > getPartners (int i)

  getPartners Returns the set of base pairs in which a given position is involved

2.5.1 Detailed Description

The SecondaryStructure class represents an RNA secondary structure, possibly with pseudoknots.

2.5.2 Constructor & Destructor Documentation

2.5.2.1 SecondaryStructure()

```
SecondaryStructure::SecondaryStructure (
            string s,
            int idd )
```

SecondaryStructure Constructs a secondary structure having given id from a dot-bracket representation.

Parameters

s	Dot-bracket representation of secondary structure
idd	Uniquer identifier for the secondary structure

2.5.3 Member Function Documentation

2.5.3.1 getLabelledBPs()

```
vector< LabeledBasePair * > SecondaryStructure::getLabelledBPs ( )
```

getLabelledBPs Returns a set of base pairs decorated by the structure identifier

Returns

Set of base pairs decorated by the structure identifier

2.5.3.2 getLength()

```
int SecondaryStructure::getLength ( )
```

getLength Returns the number of nucleotides involved in the secondary structure

Returns

Number of nucleotides involved in the secondary structure

2.5.3.3 getPartners()

```
vector< BasePair * > SecondaryStructure::getPartners (
            int i )
```

getPartners Returns the set of base pairs in which a given position is involved

Parameters

i	Position

Returns

Set of base pairs associated with a given position

2.5.3.4 getSS()

```
vector< int > SecondaryStructure::getSS ( )
```

getSS Returns a representation of the secondary structure as an array of base pairing partners

Returns

Representation of the secondary structure as an array of base pairing partners

Remark: Should only be called for non-pseudoknotted structures

The documentation for this class was generated from the following files:

- src/RNAStructure.hpp
- src/RNAStructure.cpp

2.6 TreeDecomposition Class Reference

Public Member Functions

- TreeDecomposition ()
  TreeDecomposition Constructs an empty tree decomposition.
- vector< Bag * > topologicalSort ()
  topologicalSort Builds a topological ordering of bags
- void loadFromFile (string path)
  loadFromFile Loads the content of a tree-decomposition from a file
- void addStructure (SecondaryStructure *ss)
  addStructure Adds the structural elements of a secondary structure to the suitable bags in the tree-decomposition
- vector< Bag * > getBags ()
  getBags Returns the set of bags for this tree decomposition
- void show (int depth=0)
  show Prints this tree decomposition recursively to standard error

Public Attributes

- vector< Bag * > **bags**
- vector< int > **roots**

2.6.1 Member Function Documentation

2.6.1.1 addStructure()

```
void TreeDecomposition::addStructure (
            SecondaryStructure * ss )
```

addStructure Adds the structural elements of a secondary structure to the suitable bags in the tree-decomposition

Parameters

ss	Secondary structure to be added

2.6.1.2 getBags()

```
vector< Bag * > TreeDecomposition::getBags ( )
```

getBags Returns the set of bags for this tree decomposition

Returns

List of pointers to bags in this tree decomposition

2.6.1.3 loadFromFile()

```
void TreeDecomposition::loadFromFile (
            string path )
```

loadFromFile Loads the content of a tree-decomposition from a file

Parameters

path	Path to a file describing a tree-decomposition

2.6.1.4 show()

```
void TreeDecomposition::show (
            int depth = 0 )
```

show Prints this tree decomposition recursively to standard error

Parameters

depth	Depth in tree decomposition of current bag

2.6.1.5 topologicalSort()

```
vector< Bag * > TreeDecomposition::topologicalSort ( )
```

topologicalSort Builds a topological ordering of bags

Returns

List of topologically-sorted (leaves before internal, children before parent) bags

The documentation for this class was generated from the following files:

- src/TreeDecomposition.hpp
- src/TreeDecomposition.cpp

Index