**Introduction of rucgraph**

[Folder: build\_in\_progress 3](#_Toc122195757)

[Folder: data\_structures 3](#_Toc122195758)

[PairingHeapYS.h 3](#_Toc122195759)

[PairingHeapYS\_with\_offset.h 3](#_Toc122195760)

[Union\_Find.h 3](#_Toc122195761)

[Folder: graph\_hash\_of\_mixed\_weighted (INCOMPLETE) 3](#_Toc122195762)

[Folder: graph\_hash\_of\_vectors\_unweighted 3](#_Toc122195763)

[graph\_hash\_of\_vectors\_unweighted.h 3](#_Toc122195764)

[Folder: graph\_hash\_of\_vectors\_weighted 3](#_Toc122195765)

[graph\_hash\_of\_vectors\_weighted.h 3](#_Toc122195766)

[Folder: graph\_v\_of\_v\_idealID 4](#_Toc122195767)

[graph\_v\_of\_v\_idealID.h 4](#_Toc122195768)

[graph\_v\_of\_v\_idealID\_change\_new\_vertexIDs.h 4](#_Toc122195769)

[Folder: common\_algorithms 4](#_Toc122195770)

[graph\_v\_of\_v\_idealID\_connected\_components.h 4](#_Toc122195771)

[graph\_v\_of\_v\_idealID\_shortest\_paths.h 4](#_Toc122195772)

[Folder: extract\_subgraph 4](#_Toc122195773)

[graph\_v\_of\_v\_idealID\_breadth\_first\_search\_a\_set\_of\_vertices.h 4](#_Toc122195774)

[graph\_v\_of\_v\_idealID\_extract\_subgraph.h 4](#_Toc122195775)

[Folder: random\_graph 4](#_Toc122195776)

[graph\_v\_of\_v\_idealID\_generate\_random\_connected\_graph.h 4](#_Toc122195777)

[Folder: read\_save 4](#_Toc122195778)

[graph\_v\_of\_v\_idealID\_read\_for\_GSTP.h 4](#_Toc122195779)

[graph\_v\_of\_v\_idealID\_save\_for\_GSTP.h 4](#_Toc122195780)

[Folder: text\_mining 5](#_Toc122195781)

[binary\_save\_read\_vector 5](#_Toc122195782)

[binary\_save\_read\_vector\_of\_vectors.h 5](#_Toc122195783)

[convert\_number\_to\_array\_of\_binary.h 5](#_Toc122195784)

[latitude\_and\_longitude\_distance.h 5](#_Toc122195785)

[list\_all\_files\_in\_a\_directory.h 5](#_Toc122195786)

[parse\_string.h 5](#_Toc122195787)

[parse\_substring\_between\_pairs\_of\_delimiters.h 5](#_Toc122195788)

[parse\_substring\_between\_two\_unique\_delimiters.h 5](#_Toc122195789)

[print\_items.h 5](#_Toc122195790)

[read\_csv.h 5](#_Toc122195791)

[read\_file\_line\_by\_line.h 5](#_Toc122195792)

[read\_file\_total\_line\_number.h 5](#_Toc122195793)

[replace\_chars\_in\_string.h 5](#_Toc122195794)

[string\_contains\_number.h 6](#_Toc122195795)

[string\_is\_number.h 6](#_Toc122195796)

[StringCompare\_caseInSensitive.h 6](#_Toc122195797)

[utc\_time\_to\_local\_time.h 6](#_Toc122195798)

[Folder: tool\_functions 6](#_Toc122195799)

[Combinations\_Permutations.h 6](#_Toc122195800)

[Current\_Memory\_Consumption\_of\_This\_Process.h 6](#_Toc122195801)

[ThreadPool.h 6](#_Toc122195802)

# Folder: build\_in\_progress

This folder contains informal codes.

# Folder: data\_structures

This folder contains some special data structures.

## PairingHeapYS.h

This file contains a pairing heap.

## PairingHeapYS\_with\_offset.h

This file contains an augmented pairing heap. In this heap, there is an offset value for every inside node. Using these values, we can change the key values of all inside nodes in O(1) time!

## Union\_Find.h

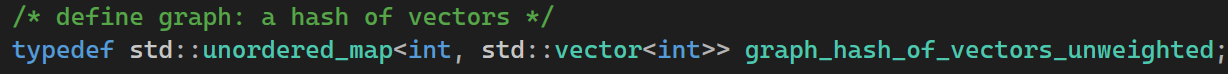
This file contains the Union Find data structure.

# Folder: graph\_hash\_of\_mixed\_weighted (INCOMPLETE)

# Folder: graph\_hash\_of\_vectors\_unweighted

## graph\_hash\_of\_vectors\_unweighted.h

This is an adjacency list build using a hash of vectors:

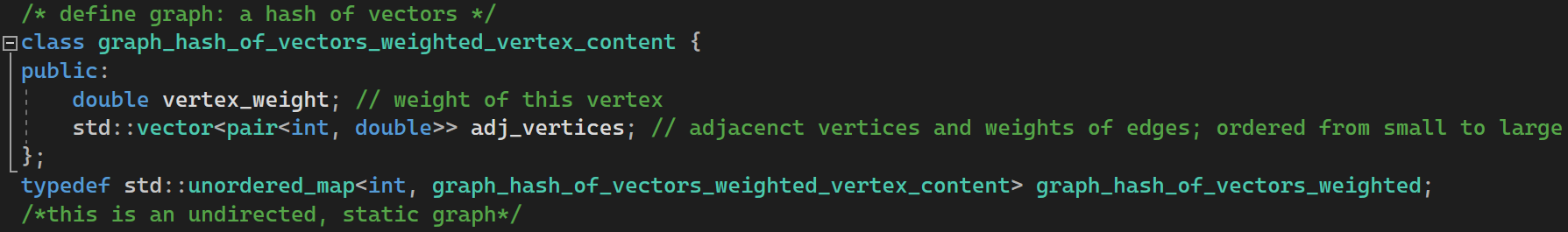


This adjacency list does not contain vertex or edge weights.

# Folder: graph\_hash\_of\_vectors\_weighted

## graph\_hash\_of\_vectors\_weighted.h

This is an adjacency list build using a hash of vectors:



This adjacency list contains vertex or edge weights.

# Folder: graph\_v\_of\_v\_idealID

## graph\_v\_of\_v\_idealID.h

This is an adjacency list with the structure of vectors of vectors.

## graph\_v\_of\_v\_idealID\_change\_new\_vertexIDs.h

This is to change vertex IDs in a graph built using graph\_v\_of\_v\_idealID.

## Folder: common\_algorithms

### graph\_v\_of\_v\_idealID\_connected\_components.h

This to find the maximum connected components of a graph built using graph\_v\_of\_v\_idealID. It returns the maximum connected components with the structure of list of lists of vertices.

### graph\_v\_of\_v\_idealID\_shortest\_paths.h

This is to find shortest paths in a graph built using graph\_v\_of\_v\_idealID.

## Folder: extract\_subgraph

### graph\_v\_of\_v\_idealID\_breadth\_first\_search\_a\_set\_of\_vertices.h

This is to breadth\_first\_search a\_set\_of\_vertices from a root vertex in a graph built using graph\_v\_of\_v\_idealID.

### graph\_v\_of\_v\_idealID\_extract\_subgraph.h

This is to extract a subgraph based on a given set of vertices.

## Folder: random\_graph

### graph\_v\_of\_v\_idealID\_generate\_random\_connected\_graph.h

This is to generate a random connected graph built using graph\_v\_of\_v\_idealID.

## Folder: read\_save

### graph\_v\_of\_v\_idealID\_read\_for\_GSTP.h

This is to read two graphs and a group of vertex IDs, for testing codes of solving group Steiner trees. The read files are generated by the following codes.

### graph\_v\_of\_v\_idealID\_save\_for\_GSTP.h

This is to save two graphs and a group of vertex IDs, for testing codes of solving group Steiner trees.

# Folder: text\_mining

## binary\_save\_read\_vector

This is to save and read vectors in binary format. Notably, the elements in vectors should have fixed sizes.

## binary\_save\_read\_vector\_of\_vectors.h

This is similar to the above file, for saving and reading vectors of vectors.

## convert\_number\_to\_array\_of\_binary.h

This is to concert a number to an array of binary values, e.g., from 3 to 11.

## latitude\_and\_longitude\_distance.h

This is to compute the distance between two points using latitude\_and\_longitude.

## list\_all\_files\_in\_a\_directory.h

This is to list all file names in a path.

## parse\_string.h

This is to parse a string based on a delimiter.

## parse\_substring\_between\_pairs\_of\_delimiters.h

This is get substrings between a pair of different delimiters.

## parse\_substring\_between\_two\_unique\_delimiters.h

This is to get the substring between two\_unique\_delimiters.

## print\_items.h

This is used to print items.

## read\_csv.h

This is to read a csv file into a vector of vectors of strings.

## read\_file\_line\_by\_line.h

This is to print a file line by line.

## read\_file\_total\_line\_number.h

This is to print the total line number of a file.

## replace\_chars\_in\_string.h

This function replace all chars "from" in a string to "to".

## string\_contains\_number.h

This is to check whether a string contains a number char.

## string\_is\_number.h

This is to check whether a string is a number.

## StringCompare\_caseInSensitive.h

This is an insensitive comparison of two strings, e.g., A==a.

## utc\_time\_to\_local\_time.h

This is to convert utc\_time\_to\_local\_time.

# Folder: tool\_functions

This folder contains some tool functions.

## Combinations\_Permutations.h

This file contains codes to enumerate every possible permutation of a set of elements.

## Current\_Memory\_Consumption\_of\_This\_Process.h

This file is to check how many RAM has been allocated by the OS to the current process.

## ThreadPool.h

This is a widely adopted ThreadPool.h implementation.