**Introduction of rucgraph**

[Folder: build\_in\_progress 4](#_Toc123036184)

[Folder: data\_structures 4](#_Toc123036185)

[PairingHeapYS.h 4](#_Toc123036186)

[PairingHeapYS\_with\_offset.h 4](#_Toc123036187)

[Union\_Find.h 4](#_Toc123036188)

[Folder: dgraph\_v\_of\_v 4](#_Toc123036189)

[dgraph\_v\_of\_v.h 4](#_Toc123036190)

[Folder: graph\_hash\_of\_mixed\_weighted 4](#_Toc123036191)

[graph\_hash\_of\_mixed\_weighted.h 4](#_Toc123036192)

[graph\_hash\_of\_mixed\_weighted\_binary\_operations.h 4](#_Toc123036193)

[graph\_hash\_of\_mixed\_weighted\_update\_vertexIDs.h 5](#_Toc123036194)

[Folder: common\_algorithms 5](#_Toc123036195)

[graph\_hash\_of\_mixed\_weighted\_connected\_components.h 5](#_Toc123036196)

[graph\_hash\_of\_mixed\_weighted\_minimum\_spanning\_tree.h 5](#_Toc123036197)

[graph\_hash\_of\_mixed\_weighted\_random\_spanning\_tree.h 5](#_Toc123036198)

[graph\_hash\_of\_mixed\_weighted\_shortest\_paths.h 5](#_Toc123036199)

[Folder: extract\_subgraph 5](#_Toc123036200)

[graph\_hash\_of\_mixed\_weighted\_breadth\_first\_search\_a\_fixed\_depth\_of\_vertices.h 5](#_Toc123036201)

[graph\_hash\_of\_mixed\_weighted\_breadth\_first\_search\_a\_fixed\_number\_of\_vertices.h 5](#_Toc123036202)

[graph\_hash\_of\_mixed\_weighted\_breadth\_first\_search\_a\_fixed\_number\_of\_vertices\_in\_unconnected\_graphs.h 5](#_Toc123036203)

[graph\_hash\_of\_mixed\_weighted\_breadth\_first\_search\_a\_fixed\_number\_of\_vertices\_in\_unconnected\_graphs\_start\_from\_maxcpn.h 5](#_Toc123036204)

[graph\_hash\_of\_mixed\_weighted\_breadth\_first\_search\_a\_set\_of\_vertices.h 5](#_Toc123036205)

[graph\_hash\_of\_mixed\_weighted\_breadth\_first\_search\_a\_tree.h 5](#_Toc123036206)

[graph\_hash\_of\_mixed\_weighted\_breadth\_first\_search\_a\_tree\_of\_edges.h 5](#_Toc123036207)

[graph\_hash\_of\_mixed\_weighted\_extract\_subgraph.h 6](#_Toc123036208)

[graph\_hash\_of\_mixed\_weighted\_random\_walk\_a\_fixed\_number\_of\_vertices\_in\_unconnected\_graphs\_start\_from\_maxcpn.h 6](#_Toc123036209)

[Folder: random\_graph 6](#_Toc123036210)

[graph\_hash\_of\_mixed\_weighted\_generate\_random\_connected\_graph.h 6](#_Toc123036211)

[graph\_hash\_of\_mixed\_weighted\_generate\_random\_graph.h 6](#_Toc123036212)

[graph\_hash\_of\_mixed\_weighted\_generate\_random\_graph\_for\_Cytoscape.cpp 6](#_Toc123036213)

[Folder: read\_save 6](#_Toc123036214)

[graph\_hash\_of\_mixed\_weighted\_binary\_save\_read.h 6](#_Toc123036215)

[graph\_hash\_of\_mixed\_weighted\_read\_for\_GSTP.h 6](#_Toc123036216)

[graph\_hash\_of\_mixed\_weighted\_read\_graph\_with\_weight.h 6](#_Toc123036217)

[graph\_hash\_of\_mixed\_weighted\_save\_for\_GSTP.h 6](#_Toc123036218)

[graph\_hash\_of\_mixed\_weighted\_save\_graph\_with\_weight.h 6](#_Toc123036219)

[Folder: two\_graphs\_operations 6](#_Toc123036220)

[graph\_hash\_of\_mixed\_weighted\_copy\_graph\_to\_another\_graph.h 6](#_Toc123036221)

[graph\_hash\_of\_mixed\_weighted\_copy\_weights\_of\_graph1\_to\_graph2.h 6](#_Toc123036222)

[graph\_hash\_of\_mixed\_weighted\_graph1\_is\_graph2.h 7](#_Toc123036223)

[graph\_hash\_of\_mixed\_weighted\_graph1\_is\_in\_graph2.h 7](#_Toc123036224)

[graph\_hash\_of\_mixed\_weighted\_merge\_graph\_hash\_of\_mixed\_weighted.h 7](#_Toc123036225)

[graph\_hash\_of\_mixed\_weighted\_to\_graph\_v\_of\_v\_idealID.h 7](#_Toc123036226)

[graph\_hash\_of\_mixed\_weighted\_to\_graph\_v\_of\_v\_idealID\_2.h 7](#_Toc123036227)

[Folder: weight\_operations 7](#_Toc123036228)

[graph\_hash\_of\_mixed\_weighted\_ec\_update\_pairwise\_jaccard\_distance.h 7](#_Toc123036229)

[graph\_hash\_of\_mixed\_weighted\_nw\_ec\_normalization.h 7](#_Toc123036230)

[graph\_hash\_of\_mixed\_weighted\_sum\_of\_nw\_ec.h 7](#_Toc123036231)

[Folder: graph\_hash\_of\_vectors\_unweighted 7](#_Toc123036232)

[graph\_hash\_of\_vectors\_unweighted.h 7](#_Toc123036233)

[Folder: graph\_hash\_of\_vectors\_weighted 7](#_Toc123036234)

[graph\_hash\_of\_vectors\_weighted.h 7](#_Toc123036235)

[Folder: graph\_v\_of\_v\_idealID 8](#_Toc123036236)

[graph\_v\_of\_v\_idealID.h 8](#_Toc123036237)

[graph\_v\_of\_v\_idealID\_change\_new\_vertexIDs.h 8](#_Toc123036238)

[Folder: common\_algorithms 8](#_Toc123036239)

[graph\_v\_of\_v\_idealID\_connected\_components.h 8](#_Toc123036240)

[graph\_v\_of\_v\_idealID\_shortest\_paths.h 8](#_Toc123036241)

[Folder: extract\_subgraph 8](#_Toc123036242)

[graph\_v\_of\_v\_idealID\_breadth\_first\_search\_a\_set\_of\_vertices.h 8](#_Toc123036243)

[graph\_v\_of\_v\_idealID\_extract\_subgraph.h 8](#_Toc123036244)

[Folder: random\_graph 8](#_Toc123036245)

[graph\_v\_of\_v\_idealID\_generate\_random\_connected\_graph.h 8](#_Toc123036246)

[Folder: read\_save 8](#_Toc123036247)

[graph\_v\_of\_v\_idealID\_read\_for\_GSTP.h 8](#_Toc123036248)

[graph\_v\_of\_v\_idealID\_save\_for\_GSTP.h 8](#_Toc123036249)

[Folder: text\_mining 9](#_Toc123036250)

[binary\_save\_read\_vector 9](#_Toc123036251)

[binary\_save\_read\_vector\_of\_vectors.h 9](#_Toc123036252)

[convert\_number\_to\_array\_of\_binary.h 9](#_Toc123036253)

[latitude\_and\_longitude\_distance.h 9](#_Toc123036254)

[list\_all\_files\_in\_a\_directory.h 9](#_Toc123036255)

[parse\_string.h 9](#_Toc123036256)

[parse\_substring\_between\_pairs\_of\_delimiters.h 9](#_Toc123036257)

[parse\_substring\_between\_two\_unique\_delimiters.h 9](#_Toc123036258)

[print\_items.h 9](#_Toc123036259)

[read\_csv.h 9](#_Toc123036260)

[read\_file\_line\_by\_line.h 9](#_Toc123036261)

[read\_file\_total\_line\_number.h 9](#_Toc123036262)

[replace\_chars\_in\_string.h 9](#_Toc123036263)

[string\_contains\_number.h 10](#_Toc123036264)

[string\_is\_number.h 10](#_Toc123036265)

[StringCompare\_caseInSensitive.h 10](#_Toc123036266)

[utc\_time\_to\_local\_time.h 10](#_Toc123036267)

[Folder: tool\_functions 10](#_Toc123036268)

[Combinations\_Permutations.h 10](#_Toc123036269)

[Current\_Memory\_Consumption\_of\_This\_Process.h 10](#_Toc123036270)

[ThreadPool.h 10](#_Toc123036271)

[ThreadPool2.h 10](#_Toc123036272)

# 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: dgraph\_v\_of\_v

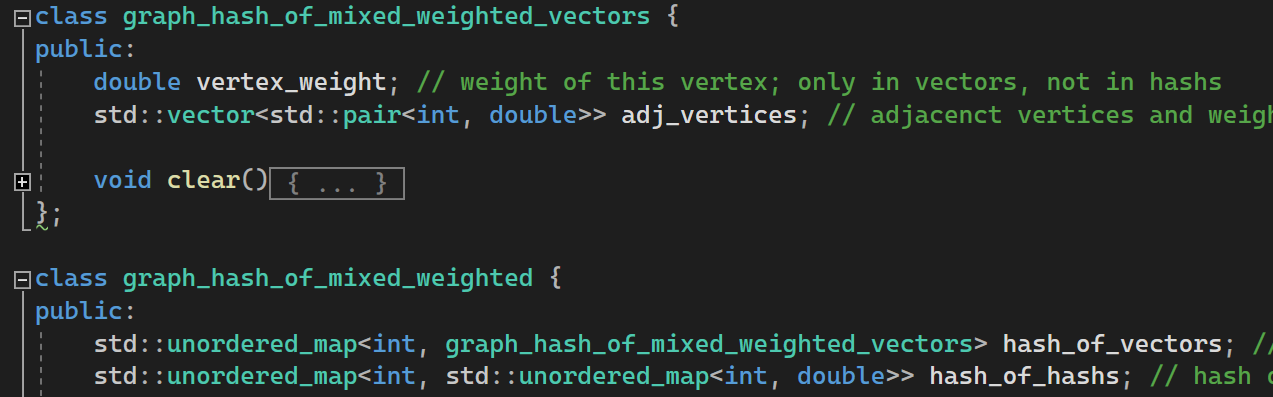
## dgraph\_v\_of\_v.h

This is an adjacency list built using vectors, for representing a directed graph with edge weights.

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

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

This is a complex adjacency list. Its structure is kind of between hash of vectors and hash of hashes. If a vertex has a large degree, then its adjacency edges and edges weights are stored in a sorted vector, otherwise stored in a hash. This complex nature is to balance the space and search time for achieving an ideal time complexity.



## graph\_hash\_of\_mixed\_weighted\_binary\_operations.h

This is to binary operate a sorted vector of edges, for achieving a high efficiency.

## graph\_hash\_of\_mixed\_weighted\_update\_vertexIDs.h

This is change vertex IDs in a graph built using graph\_hash\_of\_mixed\_weighted.

## Folder: common\_algorithms

### graph\_hash\_of\_mixed\_weighted\_connected\_components.h

This is to find the maximum connected components of a graph. It returns the maximum connected components with the structure of list of lists (or vector of vectors) of vertices.

### graph\_hash\_of\_mixed\_weighted\_minimum\_spanning\_tree.h

This is to find a minimum spanning tree of a graph.

### graph\_hash\_of\_mixed\_weighted\_random\_spanning\_tree.h

This is to find a random spanning tree of a graph.

### graph\_hash\_of\_mixed\_weighted\_shortest\_paths.h

This is to find shortest paths.

## Folder: extract\_subgraph

### graph\_hash\_of\_mixed\_weighted\_breadth\_first\_search\_a\_fixed\_depth\_of\_vertices.h

This is to breadth\_first\_search a\_fixed\_depth\_of\_vertices, usually used in experiments.

### graph\_hash\_of\_mixed\_weighted\_breadth\_first\_search\_a\_fixed\_number\_of\_vertices.h

This is to breadth\_first\_search a\_fixed number of\_vertices.

Note that, we assume that a fixed number of vertices can be searched, otherwise it returns as many vertices as possible.

### graph\_hash\_of\_mixed\_weighted\_breadth\_first\_search\_a\_fixed\_number\_of\_vertices\_in\_unconnected\_graphs.h

This is to breadth\_first\_search a\_fixed number of\_vertices in disconnected graphs. It may randomly search multiple components.

### graph\_hash\_of\_mixed\_weighted\_breadth\_first\_search\_a\_fixed\_number\_of\_vertices\_in\_unconnected\_graphs\_start\_from\_maxcpn.h

This is to breadth\_first\_search a\_fixed number of\_vertices in disconnected graphs. It search components in a decreasing order of the sizes of components. This is frequently used in experiments.

### graph\_hash\_of\_mixed\_weighted\_breadth\_first\_search\_a\_set\_of\_vertices.h

This is to breadth\_first\_search a set of\_vertices.

### graph\_hash\_of\_mixed\_weighted\_breadth\_first\_search\_a\_tree.h

This is to breadth\_first\_search a tree.

### graph\_hash\_of\_mixed\_weighted\_breadth\_first\_search\_a\_tree\_of\_edges.h

This is to breadth\_first\_search a tree of edges.

### graph\_hash\_of\_mixed\_weighted\_extract\_subgraph.h

This is to extract a subgraph for a given set of vertices.

### graph\_hash\_of\_mixed\_weighted\_random\_walk\_a\_fixed\_number\_of\_vertices\_in\_unconnected\_graphs\_start\_from\_maxcpn.h

This is to random walk a\_fixed number of\_vertices in disconnected graphs. It search components in a decreasing order of the sizes of components. This is frequently used in experiments.

## Folder: random\_graph

### graph\_hash\_of\_mixed\_weighted\_generate\_random\_connected\_graph.h

This is to generate a random connected graph.

### graph\_hash\_of\_mixed\_weighted\_generate\_random\_graph.h

This is to generate a random (may not be connected) graph.

### graph\_hash\_of\_mixed\_weighted\_generate\_random\_graph\_for\_Cytoscape.cpp

This is an example cpp file for generate and save a random graph that can be draw on Cytoscape.

## Folder: read\_save

### graph\_hash\_of\_mixed\_weighted\_binary\_save\_read.h

This is to save and read a graph\_hash\_of\_mixed\_weighted in binary format.

### graph\_hash\_of\_mixed\_weighted\_read\_for\_GSTP.h

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

### graph\_hash\_of\_mixed\_weighted\_read\_graph\_with\_weight.h

This is to read a graph that is saved in readable format.

### graph\_hash\_of\_mixed\_weighted\_save\_for\_GSTP.h

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

### graph\_hash\_of\_mixed\_weighted\_save\_graph\_with\_weight.h

This is to save a graph in readable format.

## Folder: two\_graphs\_operations

### graph\_hash\_of\_mixed\_weighted\_copy\_graph\_to\_another\_graph.h

This is to merge a graph into another graph, both in the format of graph\_hash\_of\_mixed\_weighted.

### graph\_hash\_of\_mixed\_weighted\_copy\_weights\_of\_graph1\_to\_graph2.h

This is to copy vertex and edge weights in a graph into another graph, both in the format of graph\_hash\_of\_mixed\_weighted.

### graph\_hash\_of\_mixed\_weighted\_graph1\_is\_graph2.h

This is to check whether two graphs are the same.

### graph\_hash\_of\_mixed\_weighted\_graph1\_is\_in\_graph2.h

This is to check whether the vertices and edges in a graph are in another graph or not. It does not check weights.

### graph\_hash\_of\_mixed\_weighted\_merge\_graph\_hash\_of\_mixed\_weighted.h

This is to merge a graph into another graph, both in the format of graph\_hash\_of\_mixed\_weighted.

### graph\_hash\_of\_mixed\_weighted\_to\_graph\_v\_of\_v\_idealID.h

This is to change a graph\_hash\_of\_mixed\_weighted to a graph\_v\_of\_v\_idealID. This function inputs a vertex mapping function.

### graph\_hash\_of\_mixed\_weighted\_to\_graph\_v\_of\_v\_idealID\_2.h

This is to change a graph\_hash\_of\_mixed\_weighted to a graph\_v\_of\_v\_idealID. This function does not input a vertex mapping function, but inputs a max\_ID value.

## Folder: weight\_operations

### graph\_hash\_of\_mixed\_weighted\_ec\_update\_pairwise\_jaccard\_distance.h

This is to update edge weights to pairwise\_jaccard\_distance.

### graph\_hash\_of\_mixed\_weighted\_nw\_ec\_normalization.h

This is to normalize vertex and edge weights.

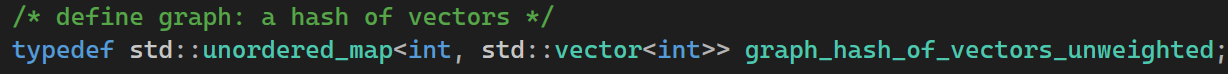
### graph\_hash\_of\_mixed\_weighted\_sum\_of\_nw\_ec.h

This is to compute the sum of vertex and/or edge weights.

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

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

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

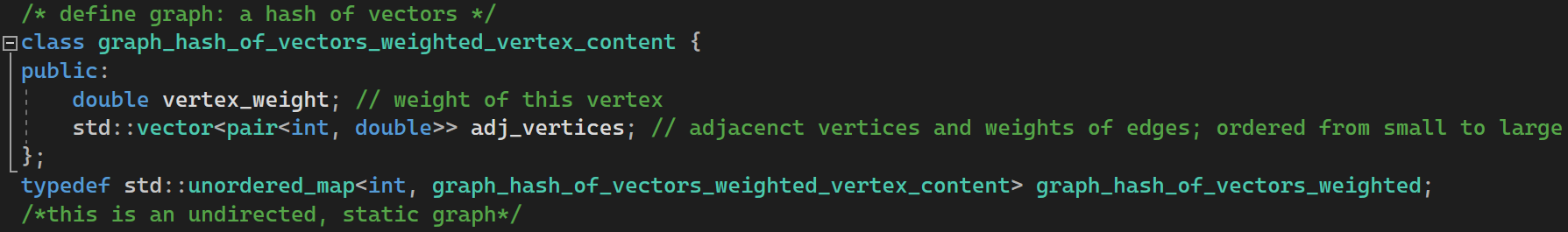


This adjacency list does not contain vertex or edge weights.

# 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.

## ThreadPool2.h

ThreadPool uses mutex, while ThreadPool2 uses shared\_mutex.

ThreadPool is faster than ThreadPool2 on Linux, but is slower than ThreadPool2 on Windows.