Maze Generator

Generated by Doxygen 1.9.8

1 Class Index 1

1.1 Class List . . . 1

2 File Index 3

2.1 File List . . . 3

3 Class Documentation 5

3.1 AdjList Struct Reference . . . 5

3.1.1 Detailed Description . . . 5

3.1.2 Member Data Documentation . . . 6

3.1.2.1 headNode . . . 6

3.2 FileLoaded Struct Reference . . . 6

3.2.1 Detailed Description . . . 7

3.2.2 Member Data Documentation . . . 7

3.2.2.1 graph . . . 7

3.2.2.2 height . . . 7

3.2.2.3 width . . . 7

3.3 Graph Struct Reference . . . 7

3.3.1 Detailed Description . . . 8

3.3.2 Member Data Documentation . . . 8

3.3.2.1 adjList . . . 8

3.3.2.2 numNodes . . . 8

3.4 Maze Struct Reference . . . 8

3.4.1 Detailed Description . . . 8

3.4.2 Member Data Documentation . . . 9

3.4.2.1 height . . . 9

3.4.2.2 matrix . . . 9

3.4.2.3 width . . . 9

3.5 MinHeap Struct Reference . . . 9

3.5.1 Detailed Description . . . 10

3.5.2 Member Data Documentation . . . 10

3.5.2.1 capacity . . . 10

3.5.2.2 nodes . . . 10

3.5.2.3 positions . . . 10

3.5.2.4 size . . . 10

3.6 MinHeapNode Struct Reference . . . 10

3.6.1 Detailed Description . . . 11

3.6.2 Member Data Documentation . . . 11

3.6.2.1 cost . . . 11

3.6.2.2 idNode . . . 11

3.7 Node Struct Reference . . . 11

3.7.1 Detailed Description . . . 12

3.7.2 Member Data Documentation . . . 12

3.7.2.1 cost . . . 12
3.7.2.2 idDestinationNode . . . 12
3.7.2.3 nextNode . . . 12
3.8 PrimResult Struct Reference . . . 12
3.8.1 Detailed Description . . . 13
3.8.2 Member Data Documentation . . . 13
3.8.2.1 predecessors . . . 13
3.8.2.2 totalCost . . . 13

4 File Documentation **15**

4.1 include/AppGenerateMaze.h File Reference . . . 15
4.1.1 Detailed Description . . . 16
4.1.2 Function Documentation . . . 16
4.1.2.1 chooseAlgo() . . . 16
4.1.2.2 doYouSaveFile() . . . 16
4.1.2.3 generateGraph() . . . 16
4.1.2.4 saveFile() . . . 17
4.1.2.5 uploadGraph() . . . 17
4.1.2.6 useMazeFile() . . . 17
4.2 AppGenerateMaze.h . . . 17
4.3 include/Graph.h File Reference . . . 18
4.3.1 Detailed Description . . . 19
4.3.2 Function Documentation . . . 19
4.3.2.1 addEdge() . . . 19
4.3.2.2 createGraph() . . . 20
4.3.2.3 createNode() . . . 20
4.3.2.4 destroyGraph() . . . 20
4.3.2.5 printGraph() . . . 20
4.4 Graph.h . . . 21
4.5 include/Heap.h File Reference . . . 21
4.5.1 Detailed Description . . . 23
4.5.2 Function Documentation . . . 23
4.5.2.1 createMinHeap() . . . 23
4.5.2.2 createMinHeapNode() . . . 23
4.5.2.3 decreaseMinHeapNodeCost() . . . 23
4.5.2.4 extractMinHeapNode() . . . 24
4.5.2.5 isInMinHeap() . . . 24
4.5.2.6 isMinHeapEmpty() . . . 24
4.5.2.7 minHeapify() . . . 25
4.5.2.8 swapMinHeapNode() . . . 25
4.6 Heap.h . . . 25
4.7 include/Load.h File Reference . . . 26

4.7.1 Detailed Description . . . 27
4.7.2 Function Documentation . . . 27
4.7.2.1 loadGraph() . . . 27
4.8 Load.h . . . 28
4.9 include/Maze.h File Reference . . . 28
4.9.1 Detailed Description . . . 29
4.9.2 Function Documentation . . . 29
4.9.2.1 createMaze() . . . 29
4.9.2.2 printMaze() . . . 30
4.9.2.3 printPPM() . . . 30
4.10 Maze.h . . . 30
4.11 include/Prim.h File Reference . . . 31
4.11.1 Detailed Description . . . 32
4.11.2 Function Documentation . . . 33
4.11.2.1 minCostIndex() . . . 33
4.11.2.2 prim() . . . 33
4.11.2.3 primHeap() . . . 33
4.11.2.4 printMST() . . . 34
4.12 Prim.h . . . 34

Index **35**

Chapter 1

Class Index

1.1 Class List

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

AdjList
: Represents the adjacency list for a node in the graph 5

FileLoaded
: Represents the loaded file data 6

Graph
: Represents the graph 7

Maze
: Structure representing a maze 8

MinHeap
: Represents a MinHeap data structure 9

MinHeapNode
: Represents a node in the MinHeap 10

Node
: Represents a node in the graph 11

PrimResult
: Structure to store the result of Prim's algorithm 12

Chapter 2

File Index

2.1 File List

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

include/AppGenerateMaze.h
This file contains the declarations of the functions used to generate a maze 15
include/Graph.h
This file contains the declarations of the Graph data structure and related functions 18
include/Heap.h
This file contains the declarations of the MinHeap data structure and related functions 21
include/Load.h
This file contains the declarations of the functions used to load a graph from a file 26
include/Maze.h
This file contains the declarations of the functions used to create and print a maze 28
include/Prim.h
Header file for Prim's algorithm implementation 31

Chapter 3

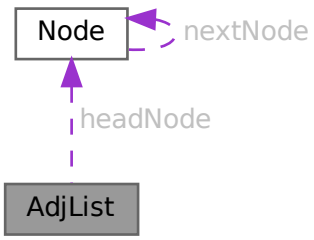
Class Documentation

3.1 AdjList Struct Reference

Represents the adjacency list for a node in the graph.

`#include <Graph.h>`

Collaboration diagram for AdjList:



Public Attributes

- Node * headNode

3.1.1 Detailed Description

Represents the adjacency list for a node in the graph.

3.1.2 Member Data Documentation

3.1.2.1 headNode

```
Node* AdjList::headNode
```

Pointer to the head node of the adjacency list.

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

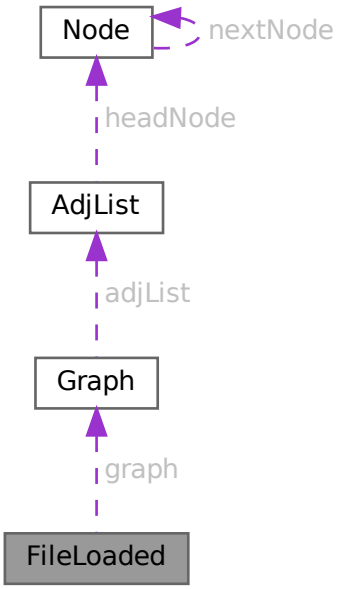
- include/Graph.h

3.2 FileLoaded Struct Reference

Represents the loaded file data.

```
#include <Load.h>
```

Collaboration diagram for FileLoaded:



Public Attributes

- int width
- int height
- Graph * graph

3.2.1 Detailed Description

Represents the loaded file data.

3.2.2 Member Data Documentation

3.2.2.1 graph

```
Graph* FileLoaded::graph
```

The graph loaded from the file.

3.2.2.2 height

```
int FileLoaded::height
```

The height of the loaded graph.

3.2.2.3 width

```
int FileLoaded::width
```

The width of the loaded graph.

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

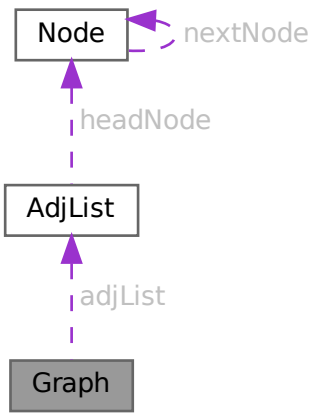
- include/Load.h

3.3 Graph Struct Reference

Represents the graph.

```
#include <Graph.h>
```

Collaboration diagram for Graph:



Public Attributes

- int numNodes
- AdjList * adjList

3.3.1 Detailed Description

Represents the graph.

3.3.2 Member Data Documentation

3.3.2.1 adjList

```
AdjList* Graph::adjList
```

Pointer to the array of adjacency lists.

3.3.2.2 numNodes

```
int Graph::numNodes
```

The number of nodes in the graph.

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

- include/Graph.h

3.4 Maze Struct Reference

Structure representing a maze.

```
#include <Maze.h>
```

Public Attributes

- int width
- int height
- bool ** matrix

3.4.1 Detailed Description

Structure representing a maze.

3.4.2 Member Data Documentation

3.4.2.1 height

```
int Maze::height
```

Height of the maze.

3.4.2.2 matrix

```
bool** Maze::matrix
```

2D matrix representing the maze.

3.4.2.3 width

```
int Maze::width
```

Width of the maze.

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

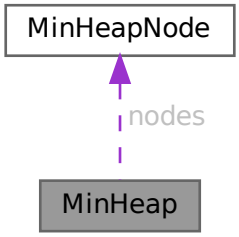
- include/Maze.h

3.5 MinHeap Struct Reference

Represents a MinHeap data structure.

```
#include <Heap.h>
```

Collaboration diagram for MinHeap:



Public Attributes

- int size
- int capacity
- int * positions
- MinHeapNode ** nodes

3.5.1 Detailed Description

Represents a MinHeap data structure.

3.5.2 Member Data Documentation

3.5.2.1 capacity

```
int MinHeap::capacity
```

The maximum capacity of the MinHeap.

3.5.2.2 nodes

```
MinHeapNode** MinHeap::nodes
```

An array of MinHeapNode pointers.

3.5.2.3 positions

```
int* MinHeap::positions
```

An array to store the positions of nodes in the MinHeap.

3.5.2.4 size

```
int MinHeap::size
```

The current size of the MinHeap.

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

- include/Heap.h

3.6 MinHeapNode Struct Reference

Represents a node in the MinHeap.

```
#include <Heap.h>
```

Public Attributes

- int idNode
- int cost

3.6.1 Detailed Description

Represents a node in the MinHeap.

3.6.2 Member Data Documentation

3.6.2.1 cost

```
int MinHeapNode::cost
```

The cost associated with the node.

3.6.2.2 idNode

```
int MinHeapNode::idNode
```

The ID of the node.

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

- include/Heap.h

3.7 Node Struct Reference

Represents a node in the graph.

```
#include <Graph.h>
```

Collaboration diagram for Node:

Public Attributes

- int idDestinationNode
- int cost
- struct Node * nextNode

3.7.1 Detailed Description

Represents a node in the graph.

3.7.2 Member Data Documentation

3.7.2.1 cost

```
int Node::cost
```

The cost of the edge to the destination node.

3.7.2.2 idDestinationNode

```
int Node::idDestinationNode
```

The ID of the destination node.

3.7.2.3 nextNode

```
struct Node* Node::nextNode
```

Pointer to the next node in the adjacency list.

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

- include/Graph.h

3.8 PrimResult Struct Reference

Structure to store the result of Prim's algorithm.

```
#include <Prim.h>
```

Public Attributes

- int * predecessors
- int totalCost

3.8.1 Detailed Description

Structure to store the result of Prim's algorithm.

This structure contains the predecessors array and the total cost of the MST.

3.8.2 Member Data Documentation

3.8.2.1 predecessors

```
int* PrimResult::predecessors
```

Array of predecessors for each node in the MST.

3.8.2.2 totalCost

```
int PrimResult::totalCost
```

Total cost of the minimum spanning tree.

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

- include/Prim.h

Chapter 4

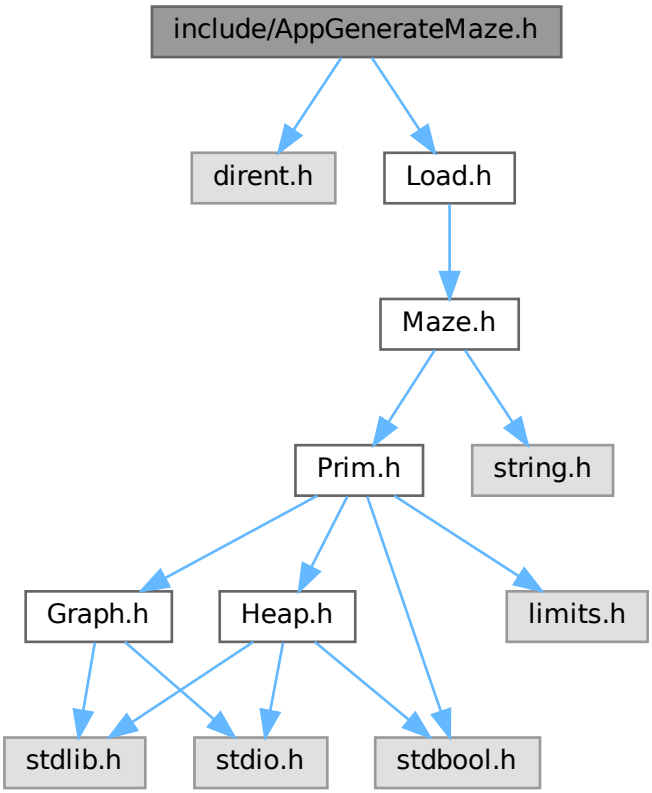
File Documentation

4.1 include/AppGenerateMaze.h File Reference

This file contains the declarations of the functions used to generate a maze.

```
#include <dirent.h>
#include "Load.h"
```

Include dependency graph for AppGenerateMaze.h:



Functions

- bool useMazeFile ()
- void uploadGraph (int ∗width, int ∗height, Graph ∗graph)
- void generateGraph (int ∗width, int ∗height, Graph ∗graph)
- int chooseAlgo ()
- bool doYouSaveFile ()
- void saveFile (Maze ∗maze)

4.1.1 Detailed Description

This file contains the declarations of the functions used to generate a maze.

4.1.2 Function Documentation

4.1.2.1 chooseAlgo()

```
int chooseAlgo ( )
```

Prompts the user to choose an algorithm for maze generation.

Returns

The chosen algorithm.

4.1.2.2 doYouSaveFile()

```
bool doYouSaveFile ( )
```

Asks the user if they want to save the generated maze to a file.

Returns

true if the user wants to save the maze to a file, false otherwise.

4.1.2.3 generateGraph()

```
void generateGraph (
            int * width,
            int * height,
            Graph * graph )
```

Generates a graph for the maze.

Parameters

width	The width of the maze.
height	The height of the maze.
graph	The graph to generate the maze on.

4.1.2.4 saveFile()

```
void saveFile (
            Maze * maze )
```

Saves the generated maze to a file.

Parameters

maze	The maze to save.

4.1.2.5 uploadGraph()

```
void uploadGraph (
            int * width,
            int * height,
            Graph * graph )
```

Uploads the graph from a maze file.

Parameters

width	The width of the maze.
height	The height of the maze.
graph	The graph to upload the maze to.

4.1.2.6 useMazeFile()

```
bool useMazeFile ( )
```

Checks if the user wants to use a maze file.

Returns

true if the user wants to use a maze file, false otherwise.

4.2 AppGenerateMaze.h

Go to the documentation of this file.

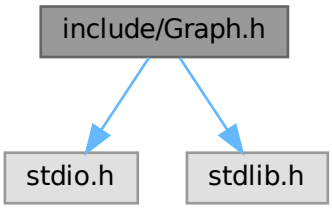
```
00001
00006 #ifndef APP_GENERATE_MAZE_H
00007 #define APP_GENERATE_MAZE_H
00008
00009 #include <dirent.h>
00010 #include "Load.h"
00011
00017 bool useMazeFile();
00018
00026 void uploadGraph(int *width, int *height, Graph *graph);
00027
00035 void generateGraph(int *width, int *height, Graph *graph);
00036
00042 int chooseAlgo();
00043
00049 bool doYouSaveFile();
00050
00056 void saveFile(Maze *maze);
00057
00058 #endif // APP_GENERATE_MAZE_H
```

4.3 include/Graph.h File Reference

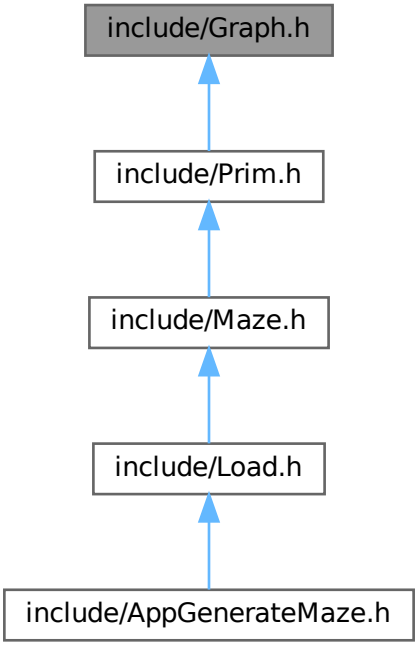
This file contains the declarations of the Graph data structure and related functions.

```
#include <stdio.h>
#include <stdlib.h>
```

Include dependency graph for Graph.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct Node

Represents a node in the graph.

- struct AdjList

  Represents the adjacency list for a node in the graph.

- struct Graph

  Represents the graph.

Typedefs

- typedef struct Node **Node**

Functions

- Node * createNode (int idDestinationNode, int cost)

  Creates a new node with the given destination node ID and cost.

- Graph * createGraph (int numNodes)

  Creates a new graph with the given number of nodes.

- void destroyGraph (Graph *graph)

  Destroys the given graph and frees the memory.

- void addEdge (Graph *graph, int idSourceNode, int idDestinationNode, int cost)

  Adds an edge between the source node and the destination node with the given cost.

- void printGraph (Graph *graph)

  Prints the graph.

4.3.1 Detailed Description

This file contains the declarations of the Graph data structure and related functions.

4.3.2 Function Documentation

4.3.2.1 addEdge()

```
void addEdge (
            Graph * graph,
            int idSourceNode,
            int idDestinationNode,
            int cost )
```

Adds an edge between the source node and the destination node with the given cost.

Parameters

graph	Pointer to the graph.
idSourceNode	The ID of the source node.
idDestinationNode	The ID of the destination node.
cost	The cost of the edge.

4.3.2.2 createGraph()

```
Graph * createGraph (
            int numNodes )
```

Creates a new graph with the given number of nodes.

Parameters

numNodes	The number of nodes in the graph.

Returns

Pointer to the newly created graph.

4.3.2.3 createNode()

```
Node * createNode (
            int idDestinationNode,
            int cost )
```

Creates a new node with the given destination node ID and cost.

Parameters

idDestinationNode	The ID of the destination node.
cost	The cost of the edge to the destination node.

Returns

Pointer to the newly created node.

4.3.2.4 destroyGraph()

```
void destroyGraph (
            Graph * graph )
```

Destroys the given graph and frees the memory.

Parameters

graph	Pointer to the graph to be destroyed.

4.3.2.5 printGraph()

```
void printGraph (
            Graph * graph )
```

Prints the graph.

Parameters

graph	Pointer to the graph to be printed.

4.4 Graph.h

Go to the documentation of this file.

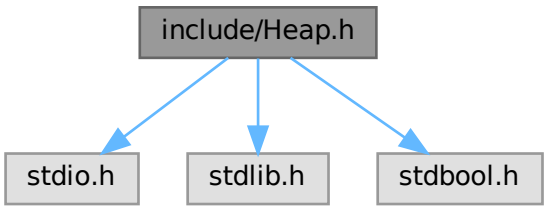
```
00001
00006 #ifndef GRAPH_H
00007 #define GRAPH_H
00008
00009 #include <stdio.h>
00010 #include <stdlib.h>
00011
00016 typedef struct Node {
00017     int idDestinationNode;
00018     int cost;
00019     struct Node* nextNode;
00020 } Node;
00021
00026 typedef struct {
00027     Node *headNode;
00028 } AdjList;
00029
00034 typedef struct {
00035     int numNodes;
00036     AdjList* adjList;
00037 } Graph;
00038
00045 Node* createNode(int idDestinationNode, int cost);
00046
00052 Graph* createGraph(int numNodes);
00053
00058 void destroyGraph(Graph* graph);
00059
00067 void addEdge(Graph* graph, int idSourceNode, int idDestinationNode, int cost);
00068
00073 void printGraph(Graph* graph);
00074
00075 #endif // GRAPH_H
```

4.5 include/Heap.h File Reference

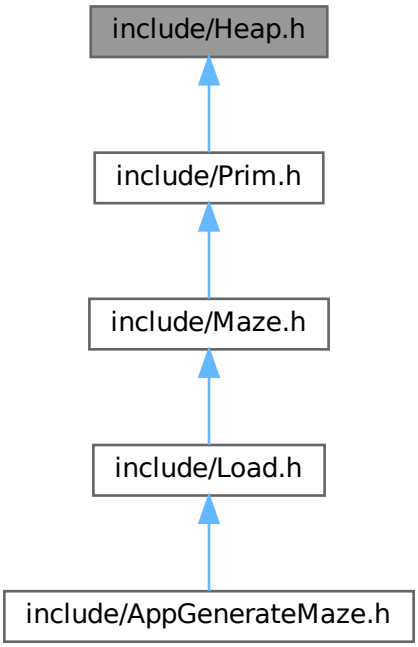
This file contains the declarations of the MinHeap data structure and related functions.

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
```

Include dependency graph for Heap.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct MinHeapNode

  Represents a node in the MinHeap.
- struct MinHeap

  Represents a MinHeap data structure.

Functions

- MinHeapNode * createMinHeapNode (int idNode, int cost)

  Creates a new MinHeapNode with the given ID and cost.
- MinHeap * createMinHeap (int capacity)

  Creates a new MinHeap with the given capacity.
- void swapMinHeapNode (MinHeapNode **a, MinHeapNode **b)

  Swaps two MinHeapNode pointers.
- void minHeapify (MinHeap *minHeap, int idNode)

  Restores the MinHeap property starting from the given node index.
- bool isMinHeapEmpty (MinHeap *minHeap)

  Checks if the MinHeap is empty.
- MinHeapNode * extractMinHeapNode (MinHeap *minHeap)

  Extracts the node with the minimum cost from the MinHeap.
- void decreaseMinHeapNodeCost (MinHeap *minHeap, int idNode, int newCost)

  Decreases the cost of the node with the given ID in the MinHeap.
- bool isInMinHeap (MinHeap *minHeap, int idNode)

  Checks if the node with the given ID is present in the MinHeap.

4.5.1 Detailed Description

This file contains the declarations of the MinHeap data structure and related functions.

4.5.2 Function Documentation

4.5.2.1 createMinHeap()

```
MinHeap * createMinHeap (
            int capacity )
```

Creates a new MinHeap with the given capacity.

Parameters

capacity	The maximum capacity of the MinHeap.

Returns

A pointer to the newly created MinHeap.

4.5.2.2 createMinHeapNode()

```
MinHeapNode * createMinHeapNode (
            int idNode,
            int cost )
```

Creates a new MinHeapNode with the given ID and cost.

Parameters

idNode	The ID of the node.
cost	The cost associated with the node.

Returns

A pointer to the newly created MinHeapNode.

4.5.2.3 decreaseMinHeapNodeCost()

```
void decreaseMinHeapNodeCost (
            MinHeap * minHeap,
            int idNode,
            int newCost )
```

Decreases the cost of the node with the given ID in the MinHeap.

Parameters

minHeap	Pointer to the MinHeap.
idNode	The ID of the node.
newCost	The new cost for the node.

4.5.2.4 extractMinHeapNode()

```
MinHeapNode * extractMinHeapNode (
            MinHeap * minHeap )
```

Extracts the node with the minimum cost from the MinHeap.

Parameters

minHeap	Pointer to the MinHeap.

Returns

A pointer to the MinHeapNode with the minimum cost.

4.5.2.5 isInMinHeap()

```
bool isInMinHeap (
            MinHeap * minHeap,
            int idNode )
```

Checks if the node with the given ID is present in the MinHeap.

Parameters

minHeap	Pointer to the MinHeap.
idNode	The ID of the node.

Returns

True if the node is present in the MinHeap, false otherwise.

4.5.2.6 isMinHeapEmpty()

```
bool isMinHeapEmpty (
            MinHeap * minHeap )
```

Checks if the MinHeap is empty.

Parameters

minHeap	Pointer to the MinHeap.

Returns

True if the MinHeap is empty, false otherwise.

4.5.2.7 minHeapify()

```
void minHeapify (
            MinHeap * minHeap,
            int idNode )
```

Restores the MinHeap property starting from the given node index.

Parameters

minHeap	Pointer to the MinHeap.
idNode	The index of the node to start heapify from.

4.5.2.8 swapMinHeapNode()

```
void swapMinHeapNode (
            MinHeapNode ** a,
            MinHeapNode ** b )
```

Swaps two MinHeapNode pointers.

Parameters

a	Pointer to the first MinHeapNode pointer.
b	Pointer to the second MinHeapNode pointer.

4.6 Heap.h

Go to the documentation of this file.

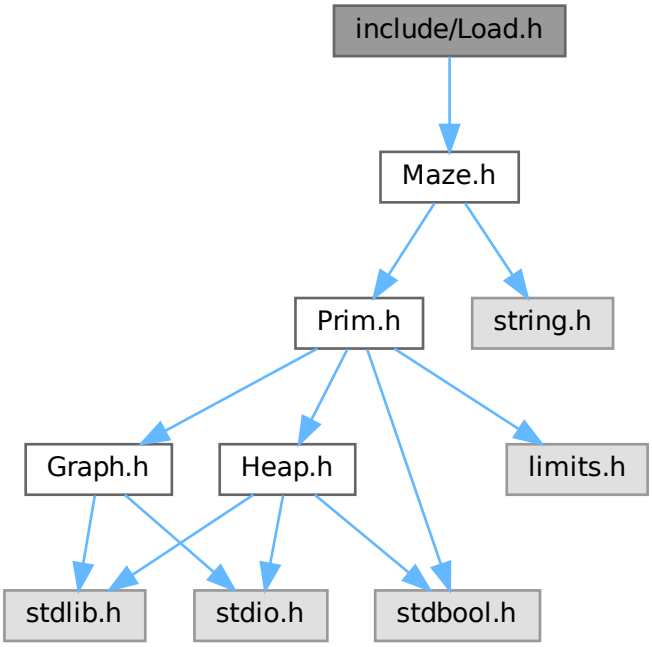
```
00001
00006 #ifndef HEAP_H
00007 #define HEAP_H
00008
00009 #include <stdio.h>
00010 #include <stdlib.h>
00011 #include <stdbool.h>
00012
00017 typedef struct {
00018     int idNode;
00019     int cost;
00020 } MinHeapNode;
00021
00026 typedef struct {
```

```
00027     int size;
00028     int capacity;
00029     int *positions;
00030     MinHeapNode **nodes;
00031 } MinHeap;
00032
00039 MinHeapNode *createMinHeapNode(int idNode, int cost);
00040
00046 MinHeap *createMinHeap(int capacity);
00047
00053 void swapMinHeapNode(MinHeapNode **a, MinHeapNode **b);
00054
00060 void minHeapify(MinHeap *minHeap, int idNode);
00061
00067 bool isMinHeapEmpty(MinHeap *minHeap);
00068
00074 MinHeapNode *extractMinHeapNode(MinHeap *minHeap);
00075
00082 void decreaseMinHeapNodeCost(MinHeap *minHeap, int idNode, int newCost);
00083
00090 bool isInMinHeap(MinHeap *minHeap, int idNode);
00091
00092 #endif // HEAP_H
```

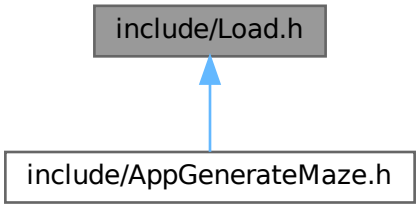
4.7 include/Load.h File Reference

This file contains the declarations of the functions used to load a graph from a file.

`#include "Maze.h"`
Include dependency graph for Load.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct FileLoaded

  Represents the loaded file data.

Functions

- FileLoaded * loadGraph (char *filename)

  Loads a graph from a file.

4.7.1 Detailed Description

This file contains the declarations of the functions used to load a graph from a file.

4.7.2 Function Documentation

4.7.2.1 loadGraph()

```
FileLoaded * loadGraph (
            char * filename )
```

Loads a graph from a file.

Parameters

filename	The name of the file to load.

Returns

A pointer to the loaded file data.

4.8 Load.h

Go to the documentation of this file.

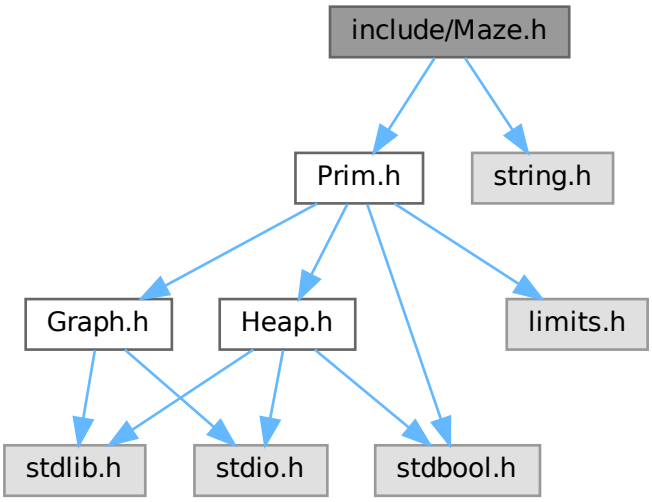
```
00001
00006 #ifndef LOAD_H
00007 #define LOAD_H
00008
00009 #include "Maze.h"
00010
00015 typedef struct {
00016     int width;
00017     int height;
00018     Graph *graph;
00019 } FileLoaded;
00020
00027 FileLoaded *loadGraph(char *filename);
00028
00029 #endif // LOAD_H
```

4.9 include/Maze.h File Reference

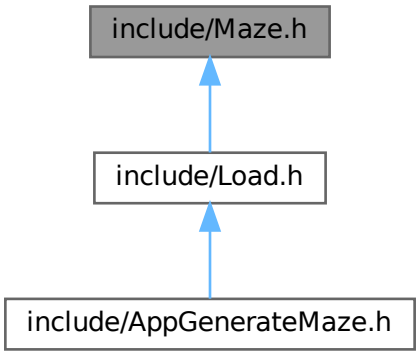
This file contains the declarations of the functions used to create and print a maze.

```
#include "Prim.h"
#include <string.h>
```

Include dependency graph for Maze.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct Maze

  Structure representing a maze.

Functions

- Maze ∗ createMaze (Graph ∗graph, int width, int height, char ∗typePrim)

  Creates a maze based on a given graph and dimensions.
- void printPPM (Maze ∗maze, char ∗filename)

  Prints the maze as a PPM image file.
- void printMaze (Maze ∗maze)

  Prints the maze to the console.

4.9.1 Detailed Description

This file contains the declarations of the functions used to create and print a maze.

4.9.2 Function Documentation

4.9.2.1 createMaze()

```
Maze * createMaze (
            Graph * graph,
            int width,
            int height,
            char * typePrim )
```

Creates a maze based on a given graph and dimensions.

Parameters

graph	The graph used to generate the maze.
width	The width of the maze.
height	The height of the maze.
typePrim	The type of Prim's algorithm to use.

Returns

A pointer to the created maze.

4.9.2.2 printMaze()

```
void printMaze (
            Maze * maze )
```

Prints the maze to the console.

Parameters

maze	The maze to print.

4.9.2.3 printPPM()

```
void printPPM (
            Maze * maze,
            char * filename )
```

Prints the maze as a PPM image file.

Parameters

maze	The maze to print.
filename	The name of the PPM image file.

4.10 Maze.h

Go to the documentation of this file.

```
00001
00006 #ifndef MAZE_H
00007 #define MAZE_H
00008
00009 #include "Prim.h"
00010 #include <string.h>
00011
00015 typedef struct {
00016     int width;
00017     int height;
00018     bool **matrix;
00019 } Maze;
```

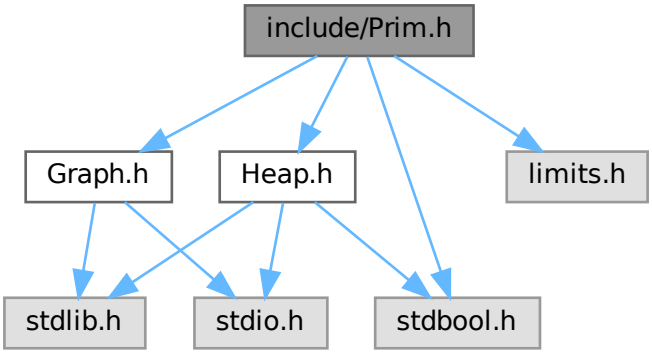
```
00020
00030 Maze *createMaze(Graph *graph, int width, int height, char *typePrim);
00031
00038 void printPPM(Maze *maze, char *filename);
00039
00045 void printMaze(Maze *maze);
00046
00047 #endif // MAZE_H
```

4.11 include/Prim.h File Reference

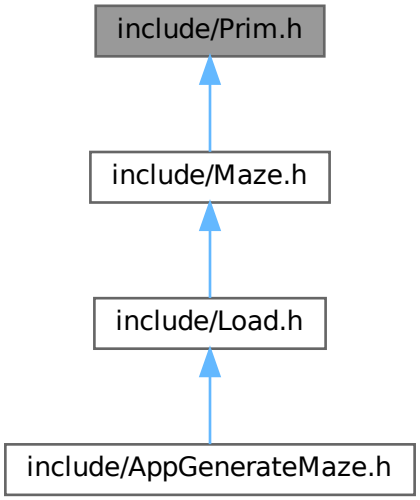
Header file for Prim's algorithm implementation.

```
#include "Graph.h"
#include "Heap.h"
#include <limits.h>
#include <stdbool.h>
```

Include dependency graph for Prim.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct PrimResult

  Structure to store the result of Prim's algorithm.

Functions

- int minCostIndex (int ∗cost, bool ∗mstSet, int numNodes)

  Find the index of the node with the minimum cost that is not yet included in the MST.
- PrimResult ∗ prim (Graph ∗graph)

  Find the minimum spanning tree (MST) of a graph using Prim's algorithm.
- PrimResult ∗ primHeap (Graph ∗graph)

  Find the minimum spanning tree (MST) of a graph using Prim's algorithm with a heap.
- void printMST (Graph ∗graph, int ∗predecessors, int ∗costs)

  Print the minimum spanning tree (MST) of a graph.

4.11.1 Detailed Description

Header file for Prim's algorithm implementation.

This file contains the declarations of functions and structures related to Prim's algorithm. Prim's algorithm is used to find the minimum spanning tree (MST) of a graph.

4.11.2 Function Documentation

4.11.2.1 minCostIndex()

```
int minCostIndex (
            int * cost,
            bool * mstSet,
            int numNodes )
```

Find the index of the node with the minimum cost that is not yet included in the MST.

This function takes an array of costs and a boolean array indicating whether a node is already included in the MST. It returns the index of the node with the minimum cost that is not yet included in the MST.

Parameters

cost	The array of costs for each node.
mstSet	The boolean array indicating whether a node is already included in the MST.
numNodes	The number of nodes in the graph.

Returns

The index of the node with the minimum cost that is not yet included in the MST.

4.11.2.2 prim()

```
PrimResult * prim (
            Graph * graph )
```

Find the minimum spanning tree (MST) of a graph using Prim's algorithm.

This function takes a graph and returns the minimum spanning tree (MST) using Prim's algorithm. It uses an adjacency matrix representation of the graph.

Parameters

graph	The graph for which to find the MST.

Returns

A pointer to the PrimResult structure containing the MST.

4.11.2.3 primHeap()

```
PrimResult * primHeap (
            Graph * graph )
```

Find the minimum spanning tree (MST) of a graph using Prim's algorithm with a heap.

This function takes a graph and returns the minimum spanning tree (MST) using Prim's algorithm. It uses a binary heap to efficiently find the node with the minimum cost.

Parameters

graph	The graph for which to find the MST.

Returns

A pointer to the PrimResult structure containing the MST.

4.11.2.4 printMST()

```
void printMST (
            Graph * graph,
            int * predecessors,
            int * costs )
```

Print the minimum spanning tree (MST) of a graph.

This function takes a graph, an array of predecessors, and an array of costs, and prints the minimum spanning tree (MST) of the graph.

Parameters

graph	The graph for which to print the MST.
predecessors	The array of predecessors for each node in the MST.
costs	The array of costs for each node in the MST.

4.12 Prim.h

Go to the documentation of this file.

```
00001
00009 #ifndef PRIM_H
00010 #define PRIM_H
00011
00012 #include "Graph.h"
00013 #include "Heap.h"
00014 #include <limits.h>
00015 #include <stdbool.h>
00016
00023 typedef struct {
00024     int *predecessors;
00025     int totalCost;
00026 } PrimResult;
00027
00040 int minCostIndex(int *cost, bool *mstSet, int numNodes);
00041
00051 PrimResult *prim(Graph* graph);
00052
00062 PrimResult *primHeap(Graph* graph);
00063
00074 void printMST(Graph* graph, int *predecessors, int *costs);
00075
00076 #endif // PRIM_H
```

Index

addEdge
    Graph.h, 19
AdjList, 5
    headNode, 6
adjList
    Graph, 8
AppGenerateMaze.h
    chooseAlgo, 16
    doYouSaveFile, 16
    generateGraph, 16
    saveFile, 17
    uploadGraph, 17
    useMazeFile, 17

capacity
    MinHeap, 10
chooseAlgo
    AppGenerateMaze.h, 16
cost
    MinHeapNode, 11
    Node, 12
createGraph
    Graph.h, 19
createMaze
    Maze.h, 29
createMinHeap
    Heap.h, 23
createMinHeapNode
    Heap.h, 23
createNode
    Graph.h, 20

decreaseMinHeapNodeCost
    Heap.h, 23
destroyGraph
    Graph.h, 20
doYouSaveFile
    AppGenerateMaze.h, 16

extractMinHeapNode
    Heap.h, 24

FileLoaded, 6
    graph, 7
    height, 7
    width, 7

generateGraph
    AppGenerateMaze.h, 16
Graph, 7
    adjList, 8
    numNodes, 8
graph
    FileLoaded, 7
Graph.h
    addEdge, 19
    createGraph, 19
    createNode, 20
    destroyGraph, 20
    printGraph, 20

headNode
    AdjList, 6
Heap.h
    createMinHeap, 23
    createMinHeapNode, 23
    decreaseMinHeapNodeCost, 23
    extractMinHeapNode, 24
    isInMinHeap, 24
    isMinHeapEmpty, 24
    minHeapify, 25
    swapMinHeapNode, 25
height
    FileLoaded, 7
    Maze, 9

idDestinationNode
    Node, 12
idNode
    MinHeapNode, 11
include/AppGenerateMaze.h, 15, 17
include/Graph.h, 18, 21
include/Heap.h, 21, 25
include/Load.h, 26, 28
include/Maze.h, 28, 30
include/Prim.h, 31, 34
isInMinHeap
    Heap.h, 24
isMinHeapEmpty
    Heap.h, 24

Load.h
    loadGraph, 27
loadGraph
    Load.h, 27

matrix
    Maze, 9
Maze, 8
    height, 9
    matrix, 9

width, 9
Maze.h
createMaze, 29
printMaze, 30
printPPM, 30
minCostIndex
Prim.h, 33
MinHeap, 9
capacity, 10
nodes, 10
positions, 10
size, 10
minHeapify
Heap.h, 25
MinHeapNode, 10
cost, 11
idNode, 11

nextNode
Node, 12
Node, 11
cost, 12
idDestinationNode, 12
nextNode, 12
nodes
MinHeap, 10
numNodes
Graph, 8

positions
MinHeap, 10
predecessors
PrimResult, 13
prim
Prim.h, 33
Prim.h
minCostIndex, 33
prim, 33
primHeap, 33
printMST, 34
primHeap
Prim.h, 33
PrimResult, 12
predecessors, 13
totalCost, 13
printGraph
Graph.h, 20
printMaze
Maze.h, 30
printMST
Prim.h, 34
printPPM
Maze.h, 30

saveFile
AppGenerateMaze.h, 17
size
MinHeap, 10
swapMinHeapNode
Heap.h, 25

totalCost
PrimResult, 13

uploadGraph
AppGenerateMaze.h, 17
useMazeFile
AppGenerateMaze.h, 17

width
FileLoaded, 7
Maze, 9