Advanced Practical Programming for Scientists

Generated by Doxygen 1.8.8

Sun Jul 16 2017 21:26:09

Contents

Chapter 1

Exercise 5: Graphs and shortest paths

Write a program that:

- reads in an undirected graph from a file given in .gph format (see here for examples) with the filename provided as a command line argument. Note that the graphs have positive edge weights (that are always below 2000000000).
- computes a longest (with respect to the edge weights) shortest path from any vertex to the vertex with index 1. In case of ties the vertex with smallest index should be chosen.
- gives an output with the following syntax:

  RESULT VERTEX 42

  RESULT DIST 5553

You may use graph libraries such as boost (for C++) or graph-tool (for Python), but you are not allowed to copy the entire program.

Deadline: June 8, 18:00

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

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

Chapter 4

Class Documentation

4.1 graph Struct Reference

Public Attributes

- int number_of_nodes
- int number_of_edges
- int * number_of_neighbours
- int * index_of_first_neighbour
- int * sorted_heads
- int * sorted_weights
- int * tail
- int * head
- int * edge_weight
- int * predecessor
- int * distance

4.1.1 Detailed Description

the graphs attributes

4.1.2 Member Data Documentation

4.1.2.1 int* graph::distance

the distance of each vertex from source

4.1.2.2 int* graph::edge_weight

the weight corresponding to each edge

4.1.2.3 int* graph::head

the head corresponding to each edge

4.1.2.4 int* graph::index_of_first_neighbour

index of first neighbour for each vertex

4.1.2.5 int graph::number_of_edges

number of edges in the graph

4.1.2.6 int* graph::number_of_neighbours

number of neighbours for each vertex

4.1.2.7 int graph::number_of_nodes

number of vertices in the graph

4.1.2.8 int* graph::predecessor

the predecessor of each vertex in shortest path tree

4.1.2.9 int* graph::sorted_heads

heads of each edge sorded by vertex

4.1.2.10 int* graph::sorted_weights

weights of each edge sorded by vertex

4.1.2.11 int* graph::tail

the tail corresponding to each edge

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

- ex5.c

Chapter 5

File Documentation

5.1 ex5.c File Reference

A program to calculate the longest shortest path of a graph using Dijkstra's algorithm.

```
#include <assert.h>
#include <ctype.h>
#include <limits.h>
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/time.h>
#include <time.h>
```

Include dependency graph for ex5.c:

Classes

- struct graph

Macros

- #define **EXT_SIZE** (1024∗1024)
- #define **MAX_LINE_LEN** 512
- #define **error_exit**(msg) error_exit_fun(msg, __FILE__, __LINE__)

Functions

- void error_exit_fun (const char ∗const msg, const char ∗const file, const int lineno)
- int sift_up (int ∗heap, int ∗distance, int ∗index, int current)
- int sift_down (int ∗heap, int ∗distance, int ∗index, int current, const int size)
- int dijkstra (struct graph ∗G, int source)
- int main (int argc, const char ∗const ∗const argv)

5.1.1 Detailed Description

A program to calculate the longest shortest path of a graph using Dijkstra's algorithm.

Author

Tri-Peter Shrive

5.1.2 Function Documentation

5.1.2.1 int dijkstra (struct graph * *G*, int *source*)

implementation of the dijkstra algorithm

Parameters

G	graph attributes
source	source node

5.1.2.2 void error_exit_fun (const char *const *msg*, const char *const *file*, const int *lineno*)

utility function that simplifies error handling

Parameters

msg	message to be displayed
file	file name
lineno	line number

5.1.2.3 int main (int *argc*, const char *const *const *argv*)

reads data from file storing nodes and weights in graph structure. then calls dijkstra's algorithm and assesses longest shortes path

5.1.2.4 int sift_down (int * *heap*, int * *distance*, int * *index*, int *current*, const int *size*)

sifts an entry down through binary heap

Parameters

heap	nodes in heap
distance	distance of nodes in heap
index	index of nodes in heap
current	current position of node in heap
size	size of heap

5.1.2.5 int sift_up (int * *heap*, int * *distance*, int * *index*, int *current*)

sifts an entry up through binary heap

Parameters

heap	nodes in heap
distance	distance of nodes in heap
index	index of nodes in heap
current	current position of node in heap