物件導向程式作業說明文件

00357027 陳威廷

檔案說明：

main.cpp：主程式區段

main.h：常數定義

fileIO.cpp/.h：處理檔案輸入

adjMatrix.cpp/.h：Adjacency Matrix物件

heap.cpp/.h：Heap物件

primsMST.cpp/.h：演算法核心

流程說明：

1. 宣告一個adjMatrix物件並初始化（使用structAdjMatrixInitialize()函式）
2. 進行檔案讀取，從使用者鍵盤輸入檔案名稱，根據檔案內容建立Adjacency Matrix（使用Link List）
3. 進行Prim’s Algorithm計算出Minimal Spanning Tree並輸出最小擴張成本
4. 記憶體釋放

完成目標：

* 實作與宣告的分檔
* 宣告常數
* Adjacency Matrix抽象化
* Heap抽象化
* 使用assert()驗證程式執行正確

未完成目標：

未使用memory\_leak檢查

測試結果：

測資輸入data1.dat

Adjacency Matrix:

Node: 6, Edge: 9

[0] -> (1, 4) -> (2, 2) -> (4, 3)

[1] -> (0, 4) -> (3, 5)

[2] -> (0, 2) -> (3, 1) -> (4, 6) -> (5, 3)

[3] -> (1, 5) -> (2, 1) -> (5, 6)

[4] -> (0, 3) -> (2, 6) -> (5, 2)

[5] -> (2, 3) -> (3, 6) -> (4, 2)

Minimal Spanning Tree:

(0, 2) - 2

(2, 3) - 1

(0, 4) - 3

(4, 5) - 2

(0, 1) - 4

Minimal Spanning Cost: 12

測資輸入data2.dat

Adjacency Matrix:

Node: 15, Edge: 36

[0] -> (4, 25) -> (10, 18)

[1] -> (6, 19) -> (7, 16)

[2] -> (4, 8) -> (7, 28) -> (9, 11) -> (10, 16) -> (11, 23)

[3] -> (10, 30) -> (11, 15) -> (13, 21) -> (14, 29)

[4] -> (0, 25) -> (2, 8) -> (6, 3) -> (8, 24) -> (9, 6) -> (13, 22)

[5] -> (7, 26) -> (11, 9) -> (12, 5) -> (13, 6)

[6] -> (1, 19) -> (4, 3) -> (7, 3) -> (9, 28) -> (12, 20)

[7] -> (1, 16) -> (2, 28) -> (5, 26) -> (6, 3) -> (8, 17) -> (9, 18) -> (11, 6) -> (12, 19)

[8] -> (4, 24) -> (7, 17) -> (11, 26) -> (13, 30) -> (14, 30)

[9] -> (2, 11) -> (4, 6) -> (6, 28) -> (7, 18) -> (10, 5) -> (11, 2) -> (14, 4)

[10] -> (0, 18) -> (2, 16) -> (3, 30) -> (9, 5) -> (12, 24) -> (13, 15)

[11] -> (2, 23) -> (3, 15) -> (5, 9) -> (7, 6) -> (8, 26) -> (9, 2)

[12] -> (5, 5) -> (6, 20) -> (7, 19) -> (10, 24)

[13] -> (3, 21) -> (4, 22) -> (5, 6) -> (8, 30) -> (10, 15)

[14] -> (3, 29) -> (8, 30) -> (9, 4)

Minimal Spanning Tree:

(0, 10) - 18

(10, 9) - 5

(9, 11) - 2

(9, 14) - 4

(9, 4) - 6

(4, 6) - 3

(6, 7) - 3

(4, 2) - 8

(11, 5) - 9

(5, 12) - 5

(5, 13) - 6

(11, 3) - 15

(7, 1) - 16

(7, 8) - 17

Minimal Spanning Cost: 117