**Programming Assignment Report**

**電機四**

**B04901056**

**張承洋**

1. **演算法流程 (Algorithm Flow) <解釋程式的運作>**

**首先，為了程式撰寫方便，我建立 id\_to\_cooridate 和 cooridate\_to\_id 兩個函數，方便我在二維座標和格子點編號 (ID) 之間做轉換（格子點編號為一維資料，數量為 number of tiles）。針對這個題目，我將每個格子點視為vertices，每個點與其上下左右的點有edge連接，以**

**base ^ (demand/capacity) - 1 作為每個邊的權重，讓 router 懂得衡量每個邊的擁擠程度，接著針對每一個 net執行課堂中提到的 Dijkstra’s shortest path algorithm ，並在每個一 net 繞線完畢時更新每個邊的權重，最後利用 backtrace的方式輸出答案。**

1. **資料結構 (Data Structure) <解釋”特別的”資料結構>**

**(a) predecessor : 為一維 number of tiles 的陣列，在執行**

**Dijkstra’s algorithm 時更新每個點的 predecessor**

**(b) distance : 為一維 number of tiles 的陣列，在執行**

**Dijkstra’s algorithm 時更新每個點到 source 的最短距離**

**，在每個 iteration 當中要確保從 heap pop 出來的資料**

**是最新版本**

**(c) heap : 儲存 pair <distance , tile ID> ， 以 distance 作為**

**key ，在每個 iteration 當中針對 heap.top() 相鄰點進行**

**relaxation ，同時更新 distance 與 predecessor 並將相**

**鄰點 push 到 heap ，再進行下一個 iteration 直到 heap**

**為空。**

1. **問題與討論 (Discussion) <討論實作中遇到的問題及疑問>**

**在實做的過程中，我曾經把 distance 的型態誤設為 int，導致 Dijkstra’s algorithm 中在判斷 relaxation 是否成功的時候精度不夠，雖然可以繞線成功，但在 overflow 數量和 wire length 的方面表現較差。**

**我所使用的 Dijkstra’s algorithm 時間複雜度為 O(E\*lgV\*n) = O(N\*lg N\*n) 其中 N = number of tiles，n = Net 數量，原因是我們需要執行 n 次 Dijkstra’s algorithm，每次需要 O(E) 次 relaxation，每次 relaxation 需要O(lgV) 的時間維護 heap properties**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test case | Number of Nets | Time(sec) | Overflow | Wire length |
| gr4x4.in | **3** | **0** | **0** | **13** |
| gr5x5.in | **10** | **0** | **0** | **42** |
| gr10x10.in | **40** | **0** | **1** | **326** |
| gr20x20..in | **1500** | **0.6** | **0** | **20542** |
| generated test case(30x30) | **3000** | **3.04** | **2110** | **62127** |
| generated test case(40x40) | **5500** | **10.24** | **29903** | **151142** |
| generated test case(50x50) | **8000** | **24.44** | **77555** | **276137** |
| gr60x60.in | **10500** | **45.6** | **53357** | **326578** |
| generated test case(90x90) | **20000** | **223.82** | **579572** | **1271200** |

