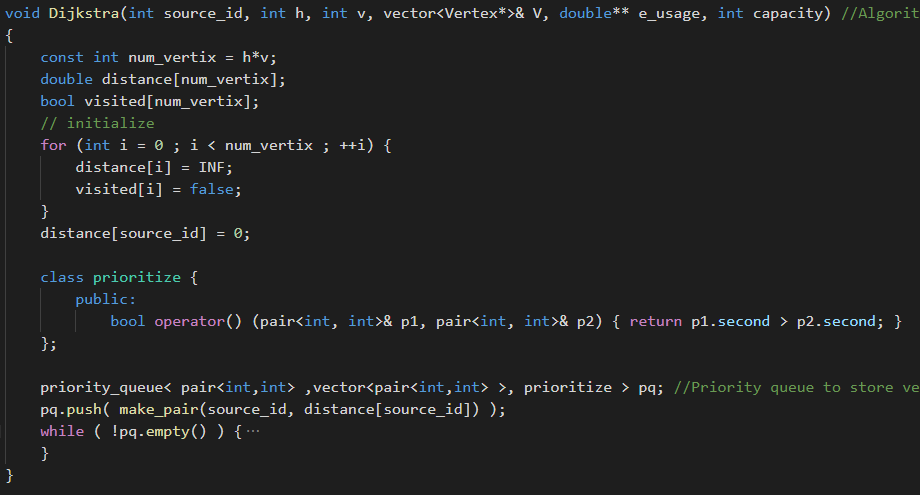
**Programming Assignment #3:**

**Global Routing Report**

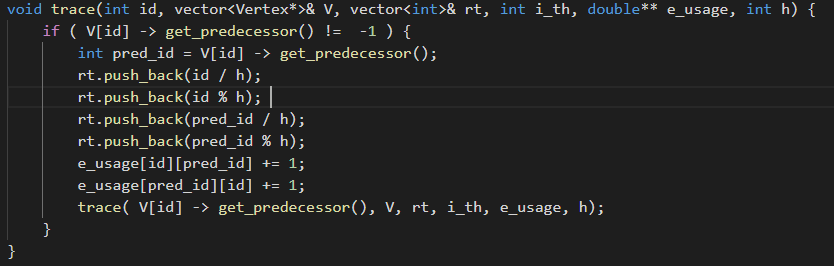
B05901182 電機三 潘彥銘

1. **演算法 (Algorithm)**
2. 每次從input檔parse一條路線時，從起點(某個grid)開始，使用**Dijkstra演算**

**法**，如下圖(一)，我用**priority queue(pq)**處理每個edge，每個edge用**pair<int,int>**表示，並額外寫個**class prioritize**讓pq知道這些pair的優先順序(距離小的在上面)，edge cost我用的是，並算出從起點到各個grid的最短距離。再從終點(某個grid) trace back，每個grid我都有記predecessor，等於-1時表示沒有predecessor(代表起點)，因此透過第116到127行的**trace**函式，透過**recursive**的方式，從終點一路往回找到起點，程式碼如下圖(二)。



圖(一)



圖(二)

1. 從起點開始找路的過程中，我並沒有一開始把每個grid的鄰居記下來，因為只要回報ID，我就知道有哪些grids是他的鄰居。
2. 每parse完一條路線，路線上的每個edge的demand就會加1，讓下條路線在路線上選擇時可以參考。

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | 1 | 2 | 3 |
| 4 | 5 | 6 | 7 |
| 8 | 9 | 10 | 11 |
| 12 | 13 | 14 | 15 |

**二、資料結構 (Data Structure)**

圖(三)

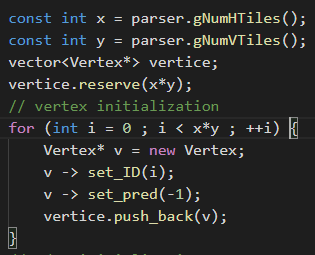
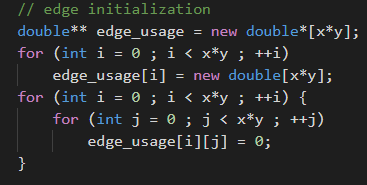
1. 一開始每個grid視為一個vertex，一個vertex是一個object，這些vertices各有

**ID**和**predecessor**，initialize **ID**為-1，predecessor也是-1(代表無predecessor)。

2. 從**parser.gNumHTiles()和parser.gNumVTiles()**就可以得知vertex的數目，以

gr4x4.in為例，可以知道vertex的數目為16，我把這些vertices一一給予

ID(0~15)，如上圖(三)，並用指標追蹤這些vertices，程式碼如下圖(四)。

圖(四) 圖(五)

3. 用二維動態陣列edge\_usage存取每條edge的demand(usage)，edge[i][j]和

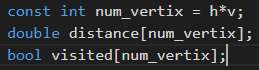
edge[j][i]代表ID為i的vertex和ID為j的vertex之間的邊的demand(usage)，程式

碼如上圖(五)。

4. 在Dijkstra演算法中，我用兩個一維陣列，第一個是distance[]來記每個vertex

與source vertex的最短距離，例如，distance[i]代表ID為i的vertex與source

vertex的最短距離; 第二個是visited[]來記這個vertex是否在先前被訪問過，

true代表有，false代表沒有，程式碼如下圖(六)。

圖(六)

5. 在Dijkstra演算法中，用priority queue來存一些pair<int, int>，這些pair的第一

個int代表vertex的ID，第二個int代表與source vertex的最短距離。

6. 在Dijkstra演算法中，前面提到我並沒有記每個vertex鄰居的資訊，而是當場

計算，透過vertex的ID，我可以得知鄰居是誰。以第一個row為例，在第一

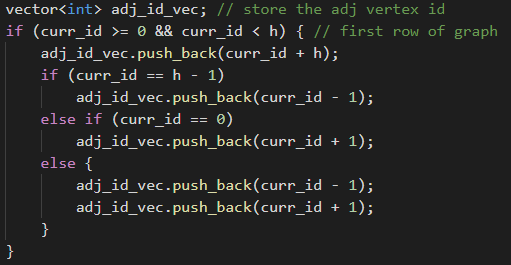
個row的ID一定在0和parser.gNumHTiles()之間，這些vertices的共通點是他們

下面都有鄰居(可參考圖(三))。接下來要分case討論，ID為0的vertex在最左

上角，因此右邊還有一個鄰居;ID為parser.gNumHTiles()的vertex在最右上

角，因此左邊還有一個鄰居;而夾在中間的vertices左右兩邊都各有一個鄰

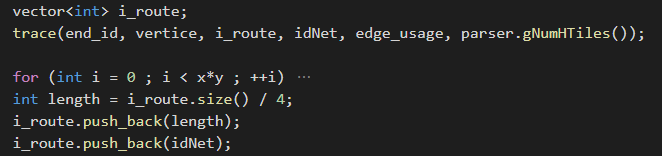
居。找到這些鄰居的ID之後，存入一個vecter<int> adj\_id\_vec裡，以方便之後的討論。程式碼如下圖(七)。



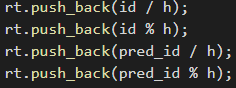
圖(七)

7. 在trace back的過程中，會新增一個vector<int> i\_route來存路線上每個vertex

的x座標、y座標、路線長度和第幾條路線，以方便之後輸出檔的處理。程式碼如下圖(八)、(九)



圖(八)

 圖(九)

**三、問題與討論 (Discussion)**

1. Edge cost討論:

(a)當edge cost = **- 1**時，三個較為代表性的case verify後的結果為下圖

gr10x10.in gr20x20.in gr60x60.in

(b)當edge cost = 時，三個較為代表性的case verify後的結果為下圖

gr10x10.in gr20x20.in gr60x60.in

從(a)(b)兩個case可以發現沒有減一的(b)表現比有減一好，在case越大時差距

越明顯，以gr60x60.in為例，overflow的數目少了5366，wirelength少了

13712。

2. 但至於上述為何case(b)的performance會好於case(a)，目前仍在思考中。