|  |
| --- |
|  |
| **Robot Path-Planning Algorithms for Fully Dynamic Shortest Path Problems Report** |
| **159.740 Intelligent System** |
|  |
| **Team Members** |
|  |
| **Ziyi Wang 18042783**  **Nan Liu 17238744**  **Yonggang Li 18043158** |

|  |
| --- |
|  |

## **Abstract**

D start Lite, LPA Start and IDA Start both are using in path-planning algorithms to find shortest path. The purpose of this experiment is to understanding the advantage and disadvantage of three different algorithms in shortest path find. And the influences of g-value, rhs-value, km-vlue and heuristic value in algorithm to find shortest path.

**Contents**

[Abstract 1](#_Toc524898256)

[Related Work 2](#_Toc524898257)

[1.1 Data structures of D\* Lite LPA\* and IDA 2](#_Toc524898258)

[Main.cpp 2](#_Toc524898259)

[Gridworld.h 3](#_Toc524898260)

[globalVariables.h 4](#_Toc524898261)

[C++ code skeleton 5](#_Toc524898262)

[Main.cpp 5](#_Toc524898263)

[Gridword.cpp 6](#_Toc524898264)

[Dstart Lite.cpp 7](#_Toc524898265)

[DStarLite.h 12](#_Toc524898266)

[LPAstart Lite.cpp 12](#_Toc524898267)

[LPAstarLite.h 15](#_Toc524898268)

[IdaStar.h 16](#_Toc524898269)

[IdaStar.cpp 16](#_Toc524898270)

[Detail of D\* Lite Algorithm, LPA\* and IDA 18](#_Toc524898271)

[1.2 Result Table 22](#_Toc524898272)

[1.3 Experiments summary 22](#_Toc524898273)

[User’s Guide 23](#_Toc524898274)

## **Related Work**

### **1.1 Data structures of D\* Lite LPA\* and IDA**

#### **Main.cpp**

DStarLite\* g\_dsl = nullptr;

LpaStar\* g\_lpas = nullptr;

bool findPath();

void updateData(bool fromMazeToMap);

// modify copyMazeToDisplayMap &copyDisplayMapToMaze, if fromMazeToMap is ture, copy date from maze to //map,flase, copy data from map to maze.



void updateH(); // call D\*Lite updateH()

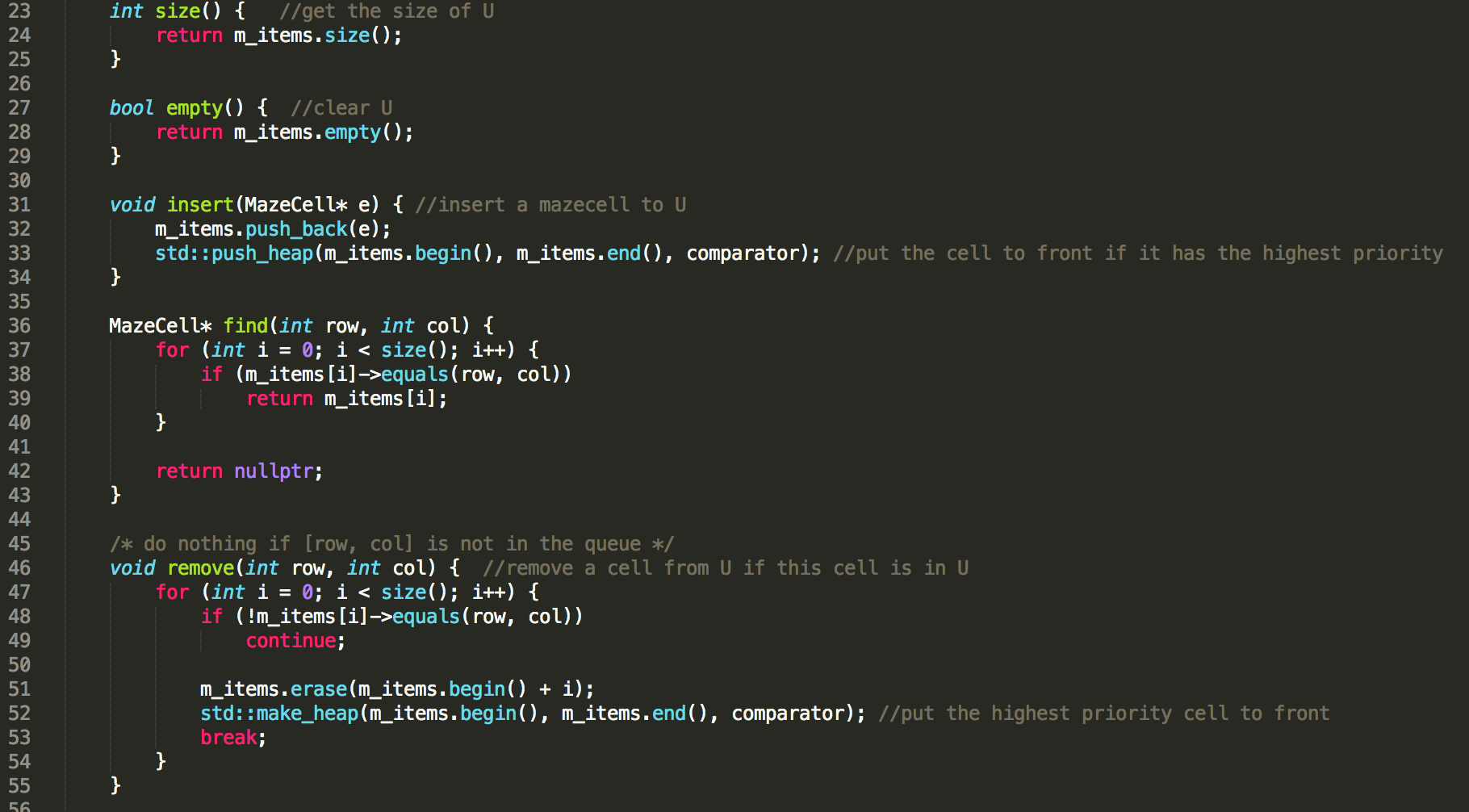
void updateKey();//calle D\*Lite updateKey()

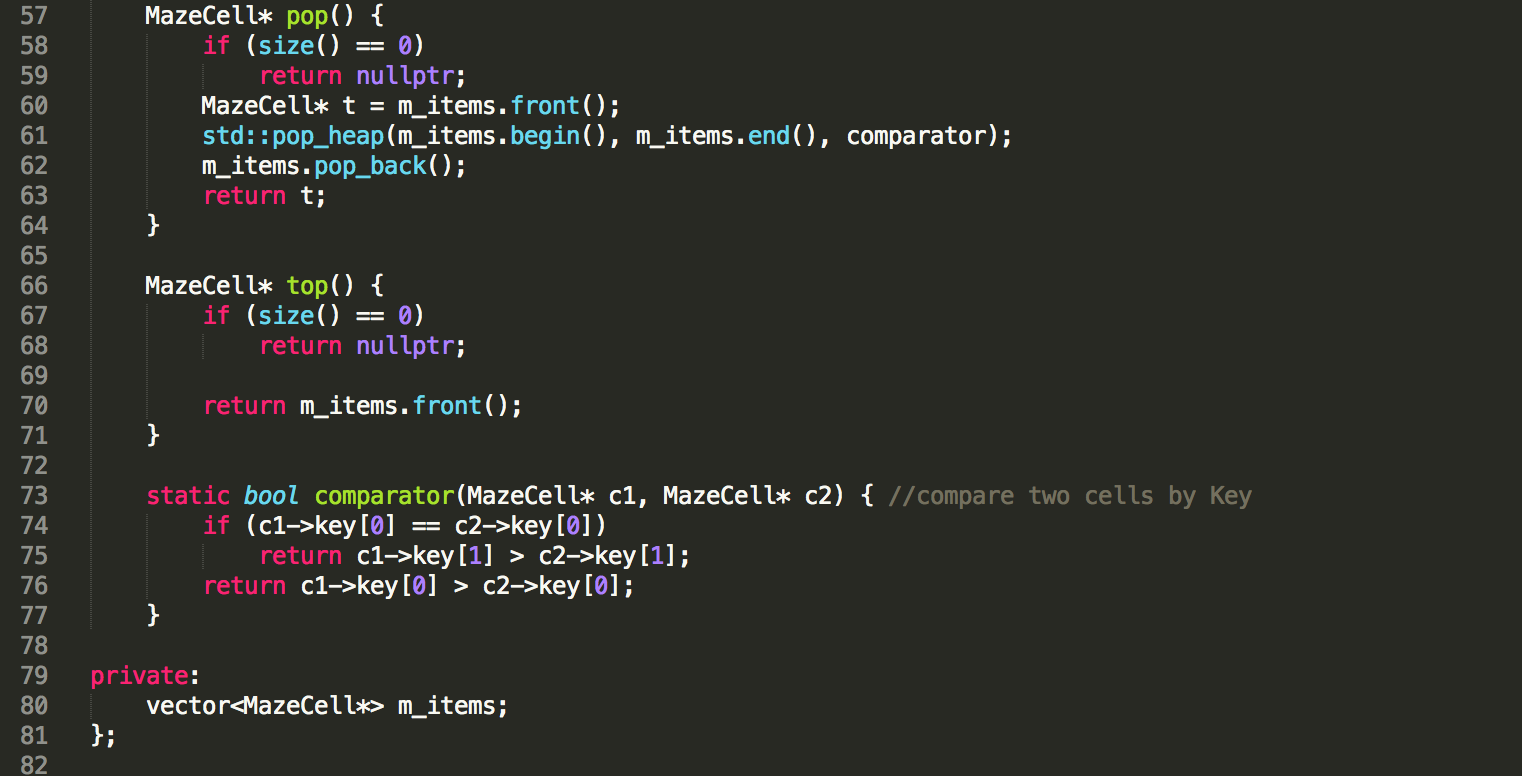
#### **Gridworld.h**

char m\_searchStatus; //-1 not found, 0 reset, 1 found the path show

void setSearchStatus(char status); // set m\_searchStatus

**PriorityQueue**



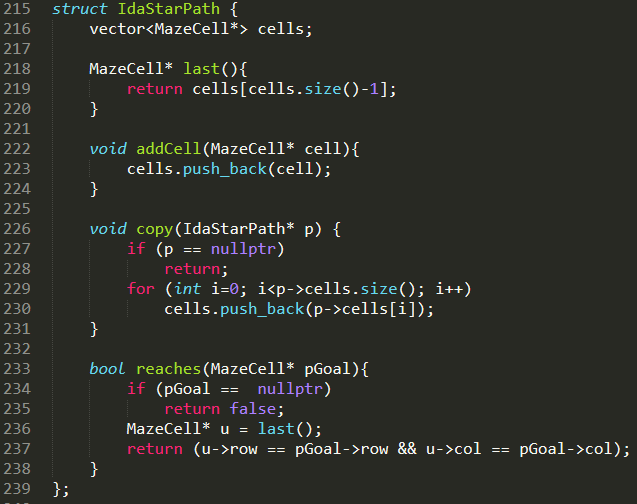


#### **globalVariables.h**

We create a struct MazeCell in globalVariables for all of algorithms:



Struct IdaStarPath is a path for IDA\*



## C++ code skeleton

#### **Main.cpp**

We modified some parts of Main.cpp to call our algorithm.

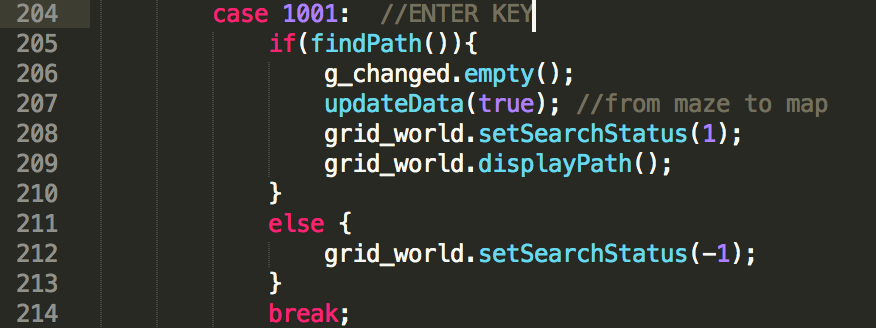
Changes in runsimulation():

case 1001: //press Enter key to calculate the shortest path.

If (findPath()=1) show the path;

// we define a new member m\_searchStatus(-1,0,1) to define the finding result status.

Else show “the goal can not be reached”; //



Case 108:

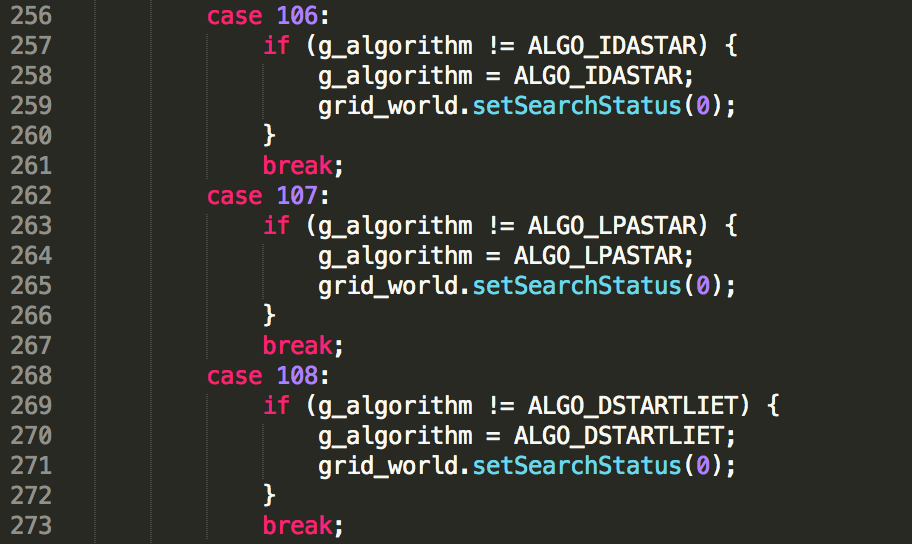
// call D\*Lite final algorithm, set m\_searchStatus to 0, be ready to calculate the shortest path.

Case 106:

//call IDA\*

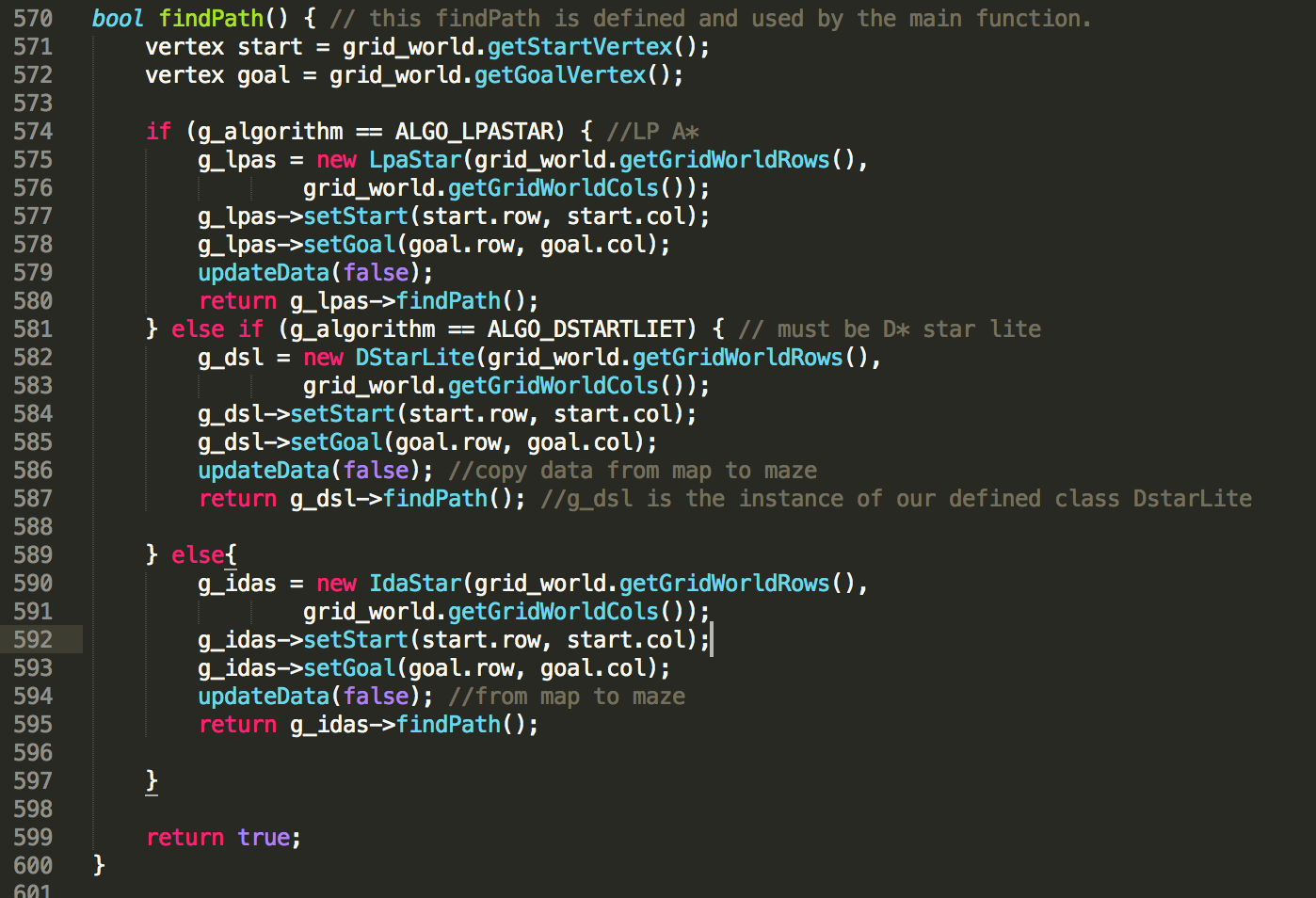
Case 107:

//call LPA\*



Add a new function

bool findPath(): this findPath() is defined and used by the main function to tell if the g\_dsl->findPath() has successfully executed. If g\_dsl->findPath() found the shortest path, it returns ture. The same as LAP\*, IDA\*.



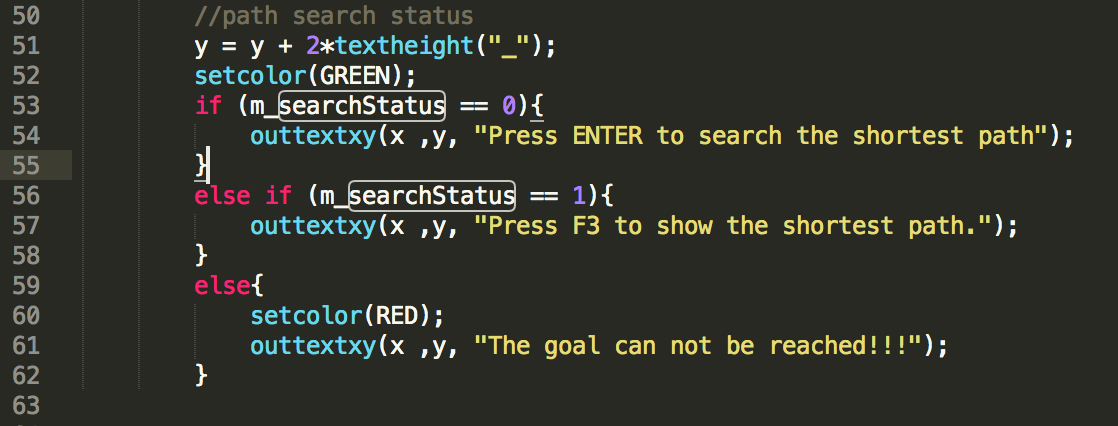
#### **Gridword.cpp**

In displayHeader(): add m\_searchStatus,

m\_searchStatus=0,reset and press enter to run;

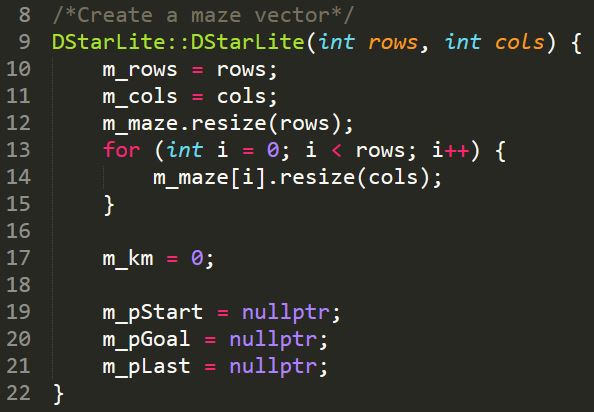
m\_searchStatus=1, show the path;

m\_searchStatus=-1, can not find the path.

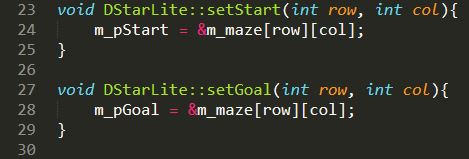


#### **Dstart Lite.cpp**

Step1 Create grildword maze vector:



Step2 set start and set goal:

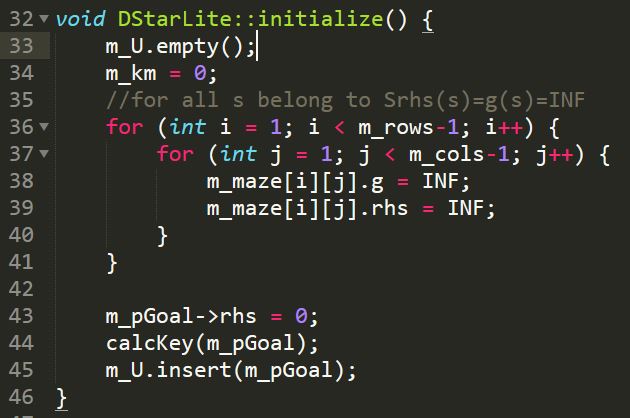


Step3 create initialize function:

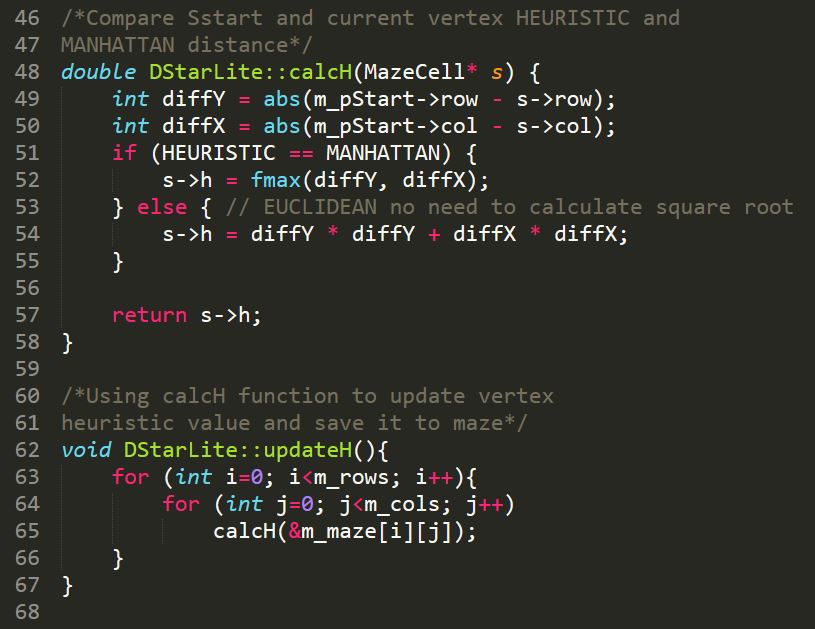
m\_U.empty(); // Empty priority queue

m\_km = 0;// Set km value equal to zero

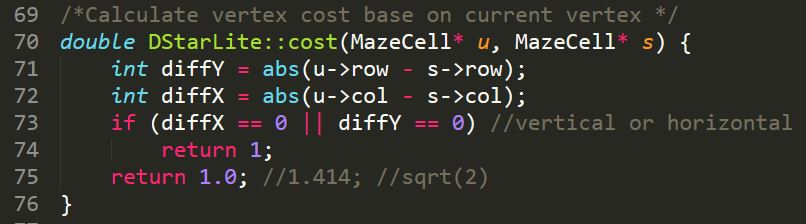
m\_pGoal->rhs = 0;// rhs(Sgoal)=0;



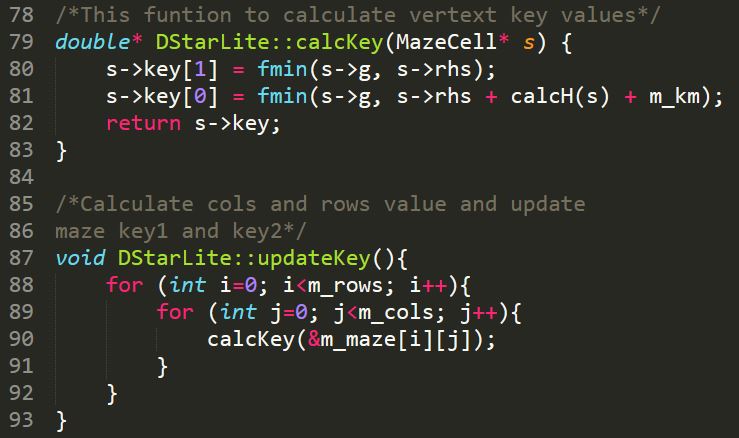
Step4 calculate heuristic value and update heuristic value:



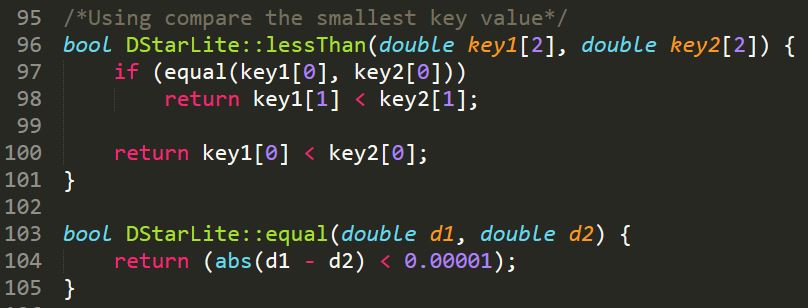
Step5 calculate c(u,s’) value:



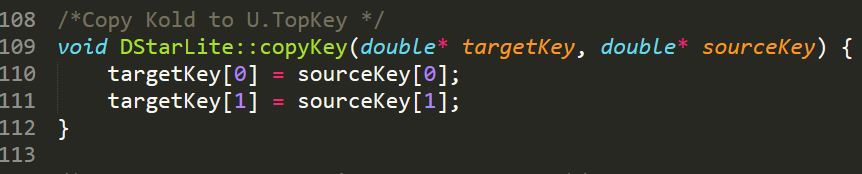
Step6 calculate Key value and update Key value:



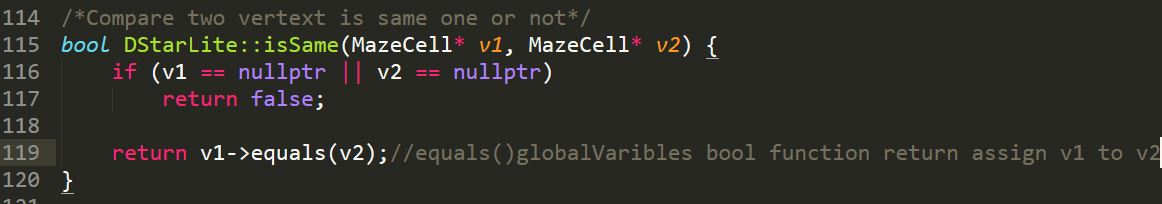
Step7 function for compare the smallest key value:



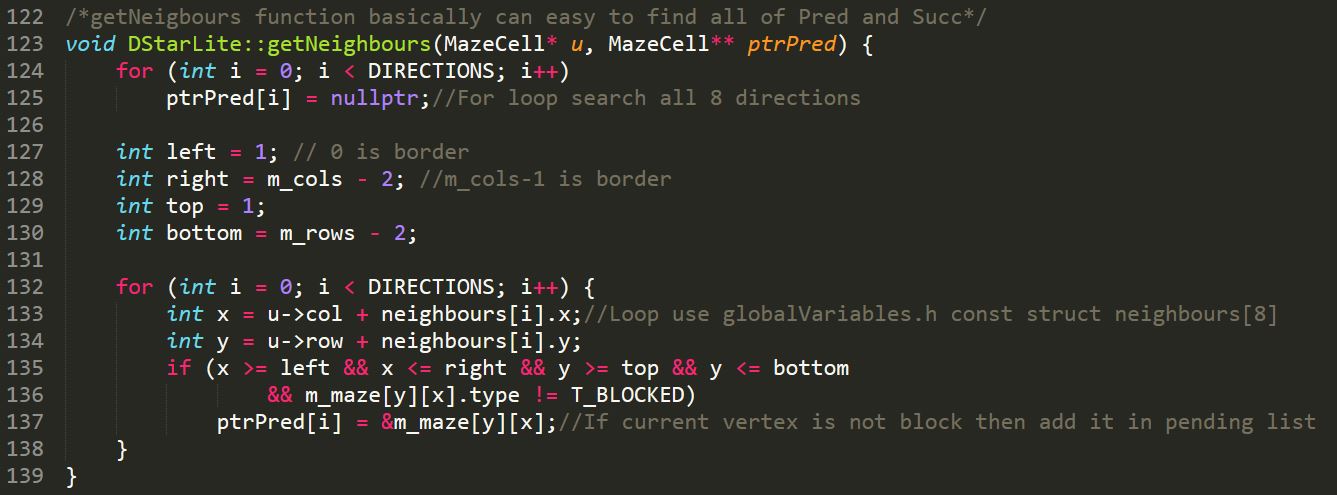
Step8 copy Kold to U.TopKey:



Step9 compare two vertext is same one or not:



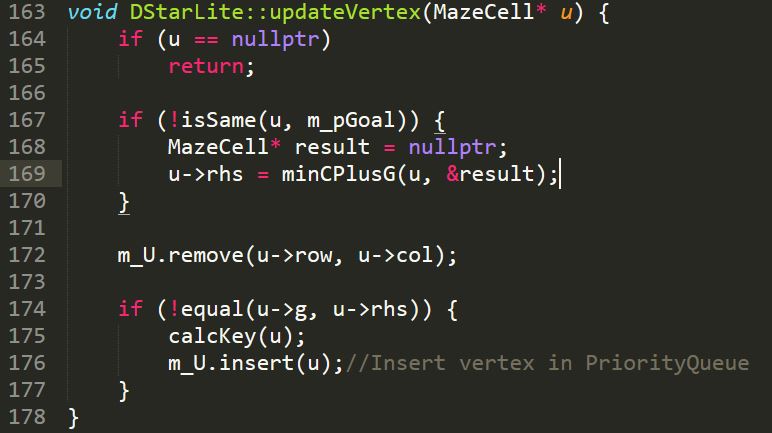
Step10 find all neighbours of current vertex:



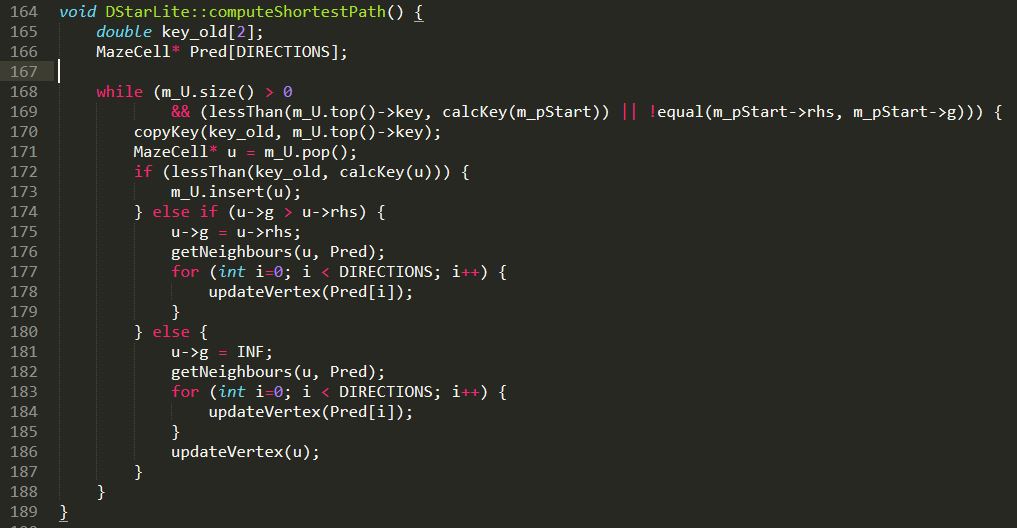
Step11 calculate minin cost pluse g-value:



Step12 create function for update vertex:



Step13 create function for compute shortest path:



Step 14 create findPath function to replaced Main function in pseudocode:



#### **DStarLite.h**



#### **LPAstart Lite.cpp**

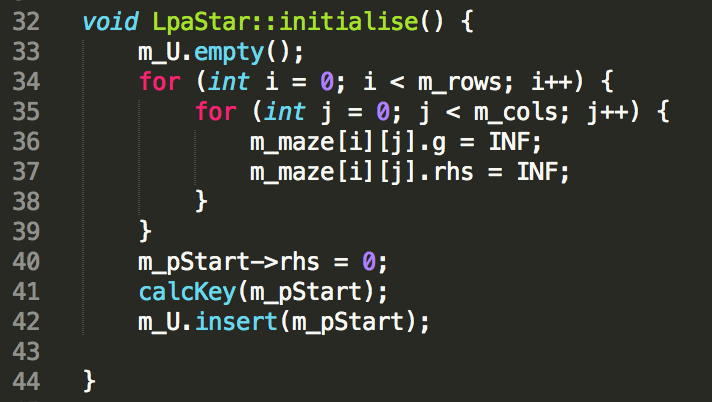
Step1 Create grildword maze vector:

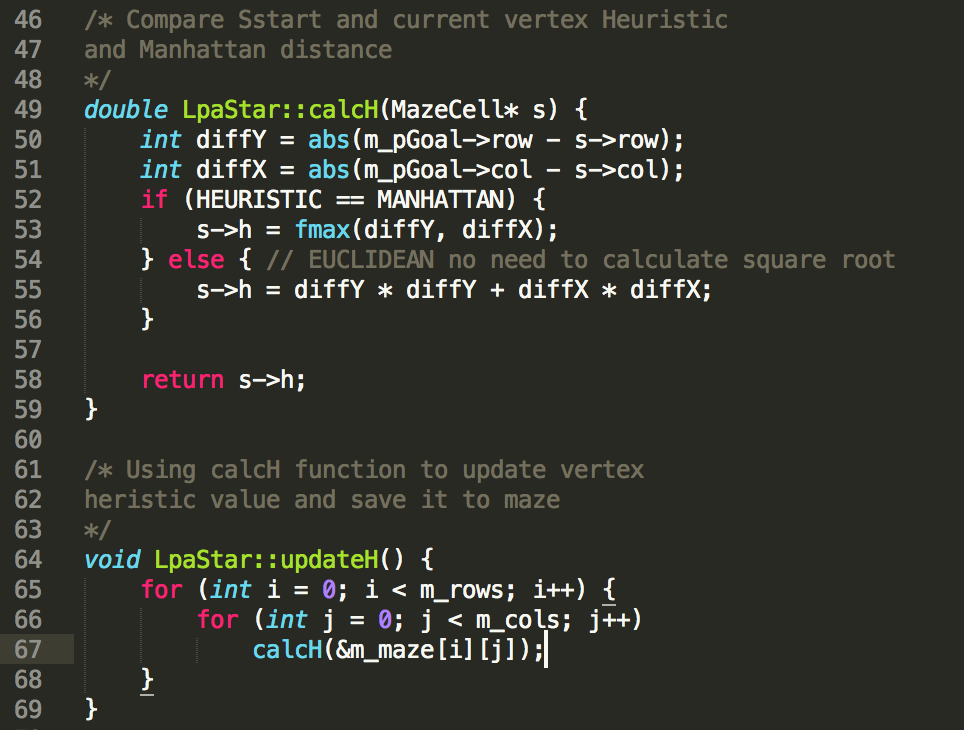
Step2 set start and set goal:

Step3 create initialize function:

m\_U.empty(); // Empty priority queue

m\_pStart->rhs = 0; // rhs(Sstart)=0;

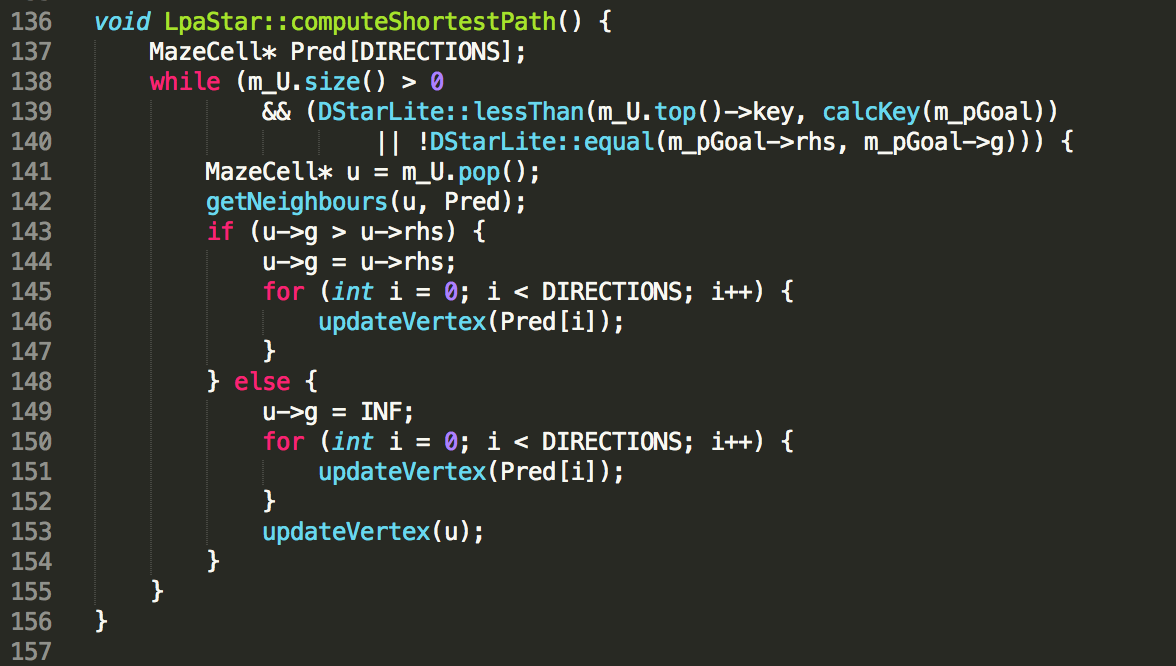


Step4 calculate heuristic value and update heuristic value: 

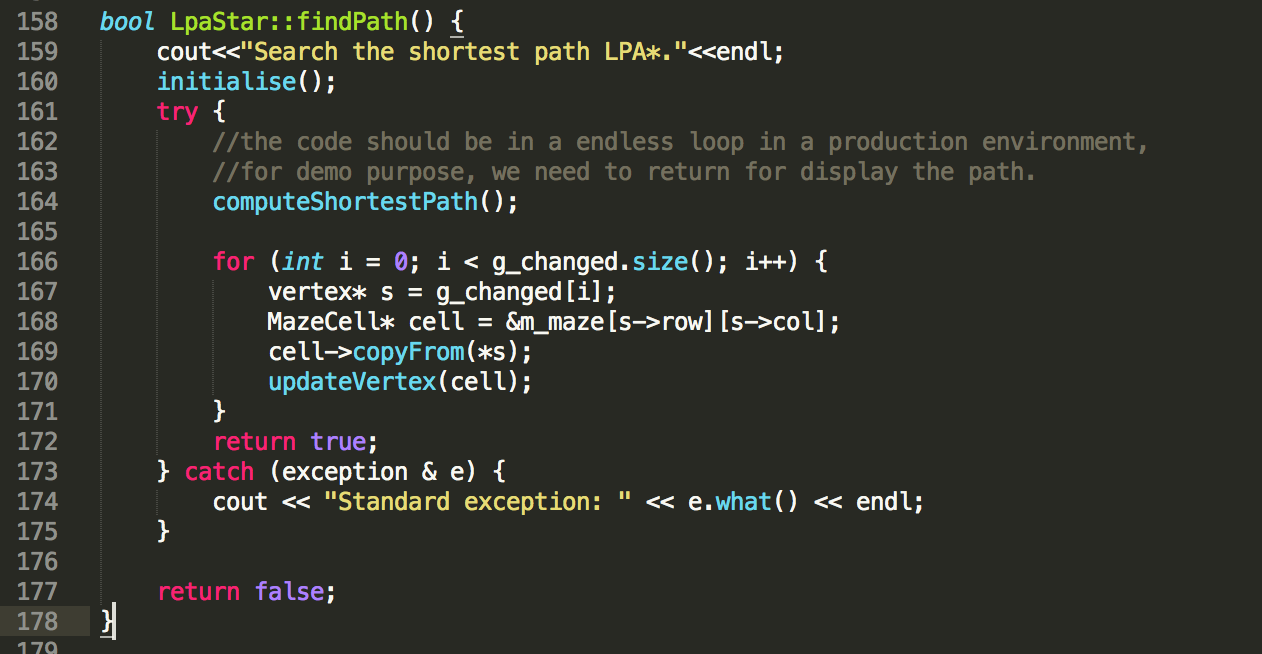
Step5 calculate Key value and update Key value:

*-ComputeShortestPath*

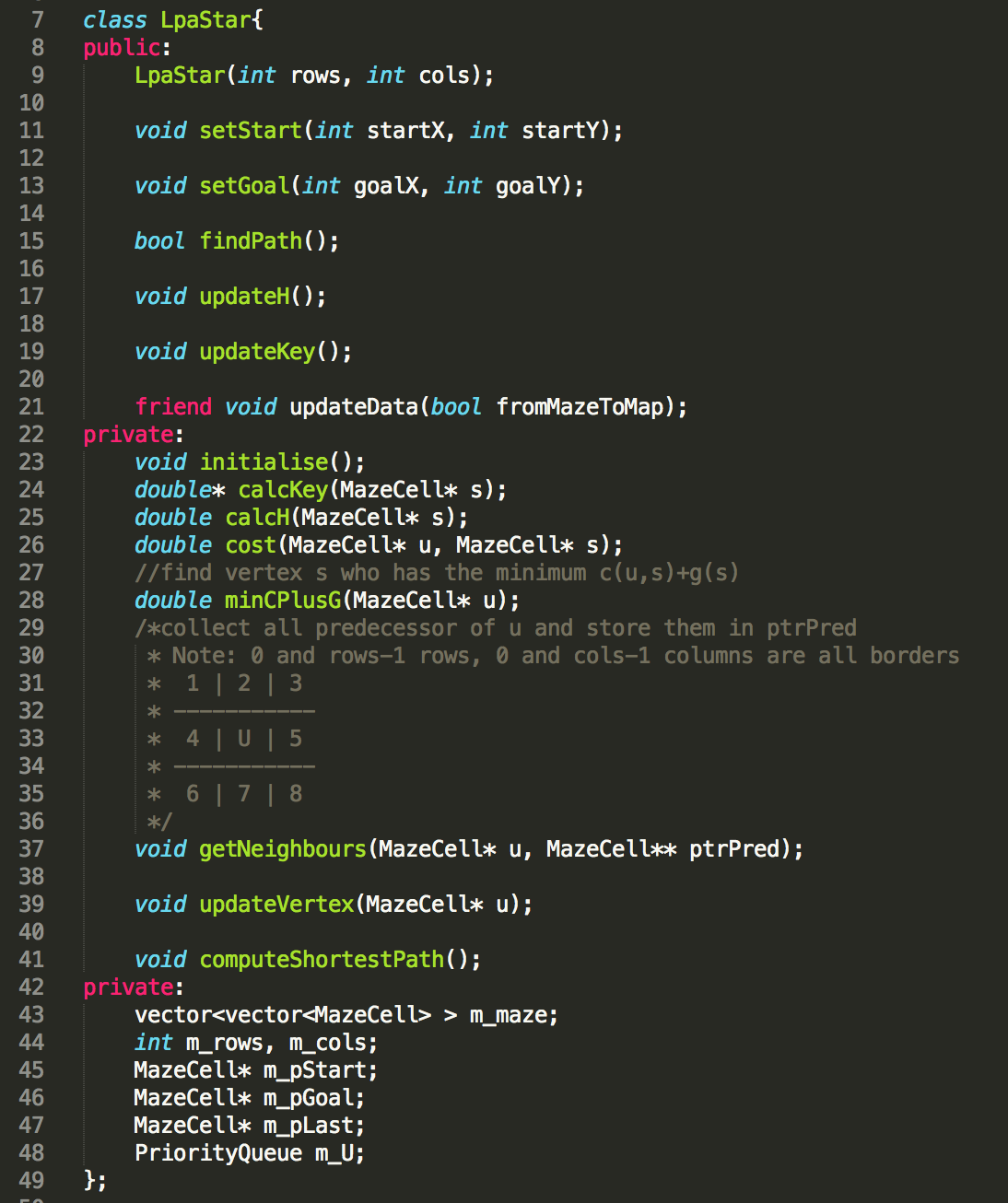
Step6 use D\*Lite function lessThan() to compare the smallest key value.



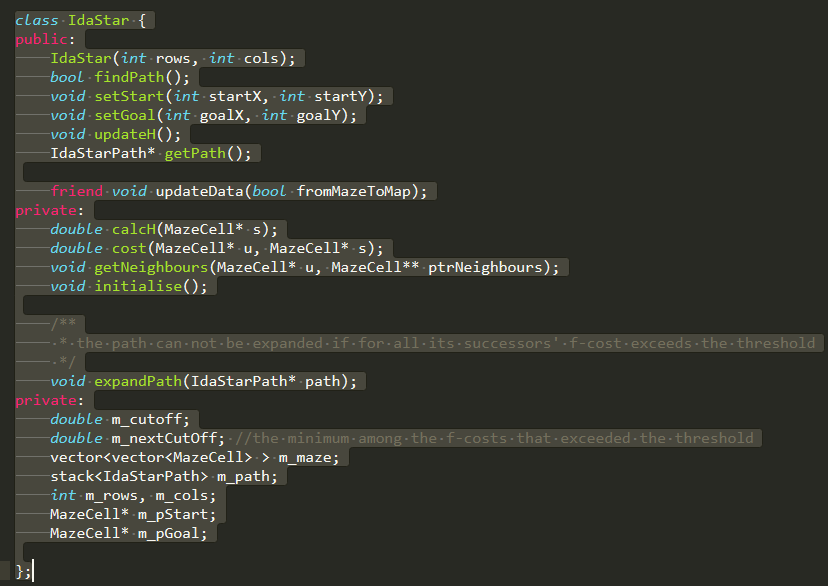
*-find path*

**

#### **LPAstarLite.h**

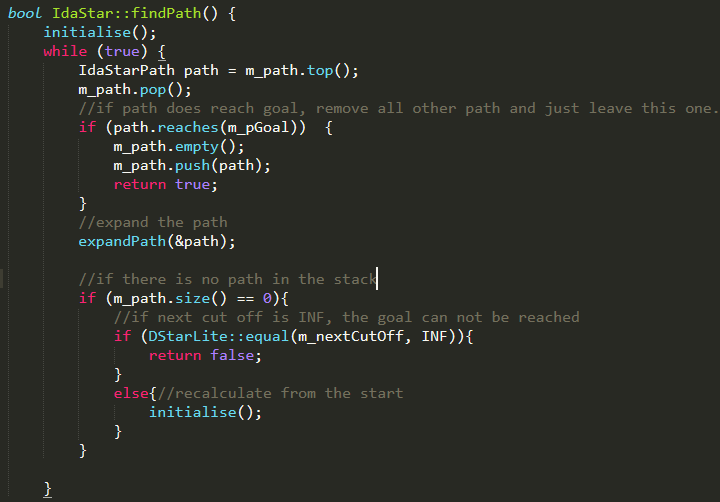


#### **IdaStar.h**

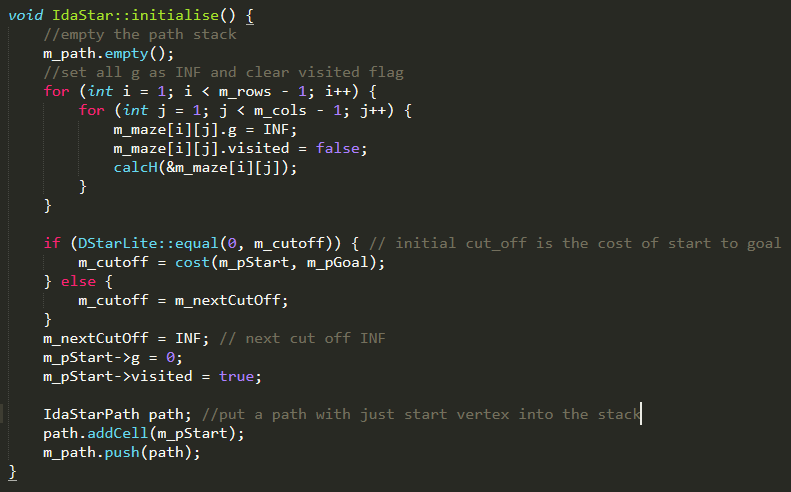


#### **IdaStar.cpp**

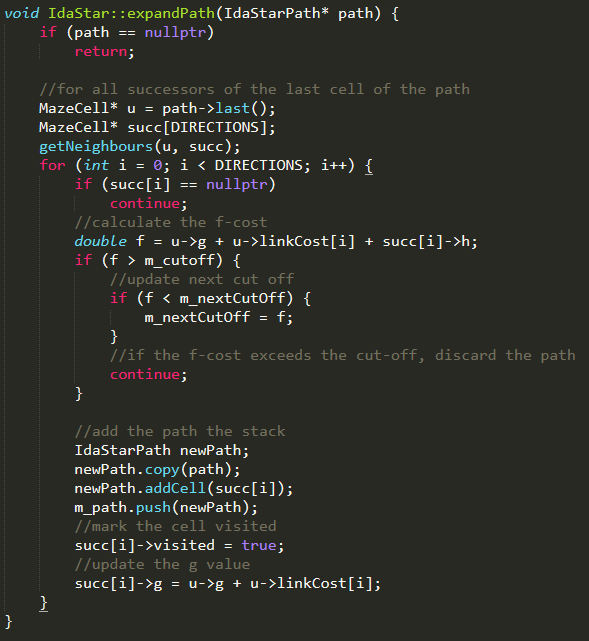
Main function for finding path.



Initialize the algorithm

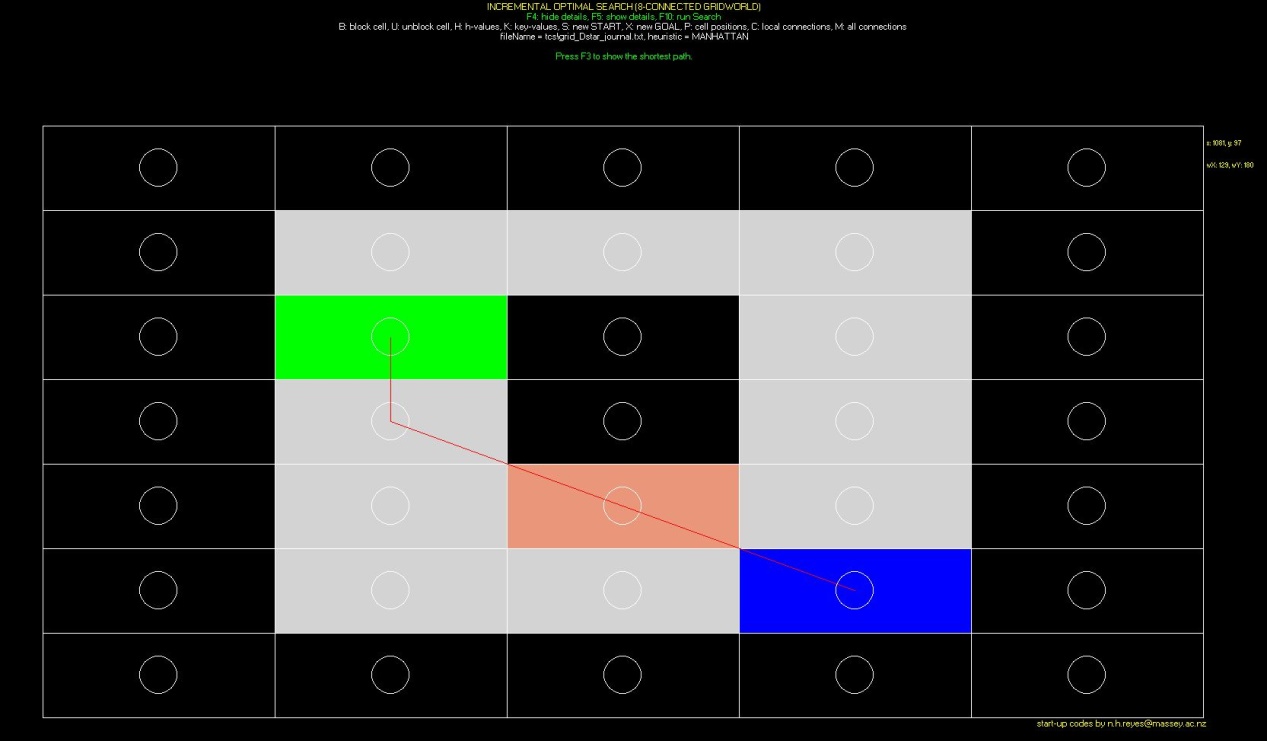


Expand the path

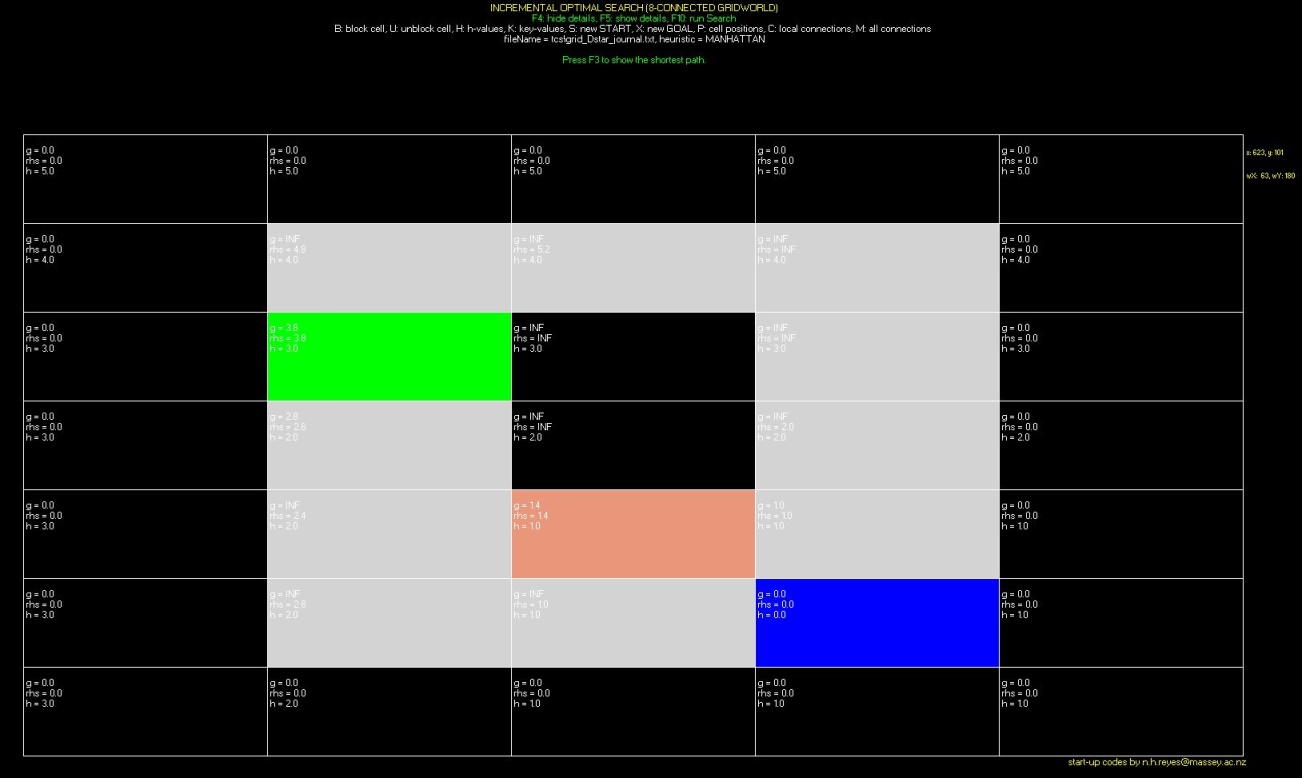


## **Detail of D\* Lite Algorithm, LPA\* and IDA**

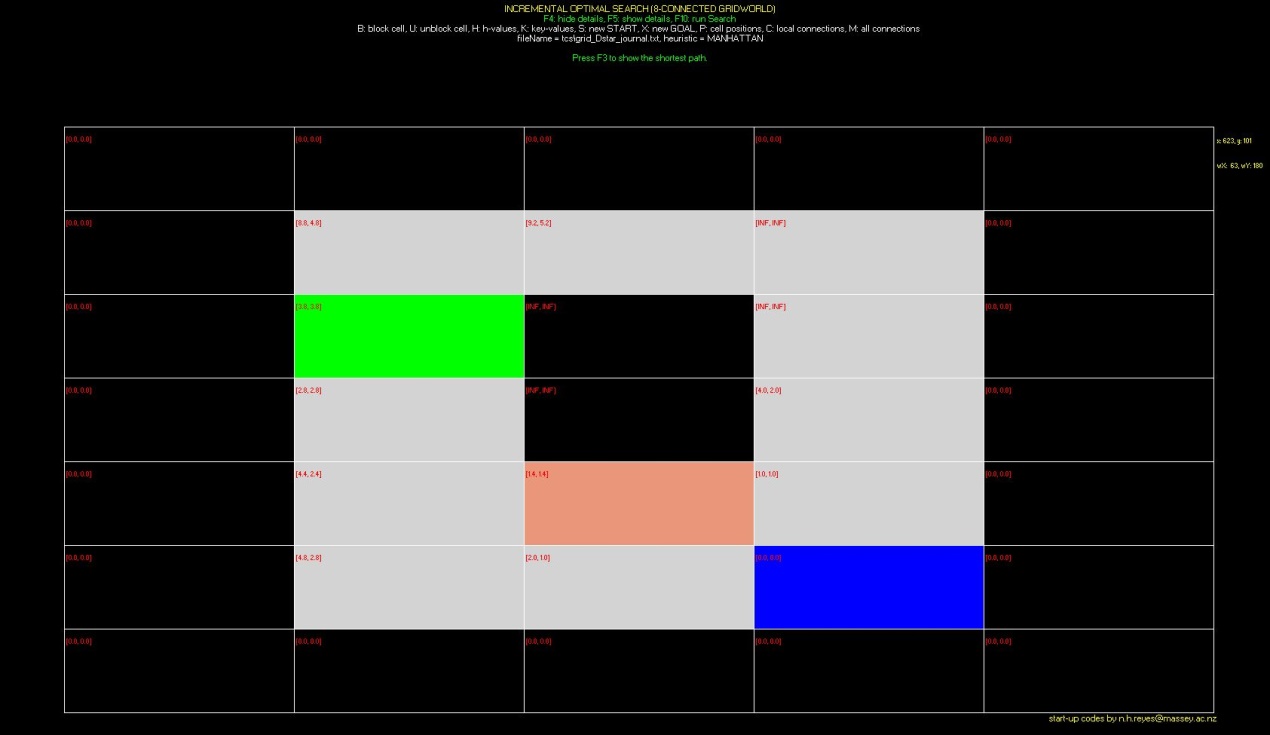
D\*Lite: shortest path



D\*Lite: g-value, rhs-value and h-value

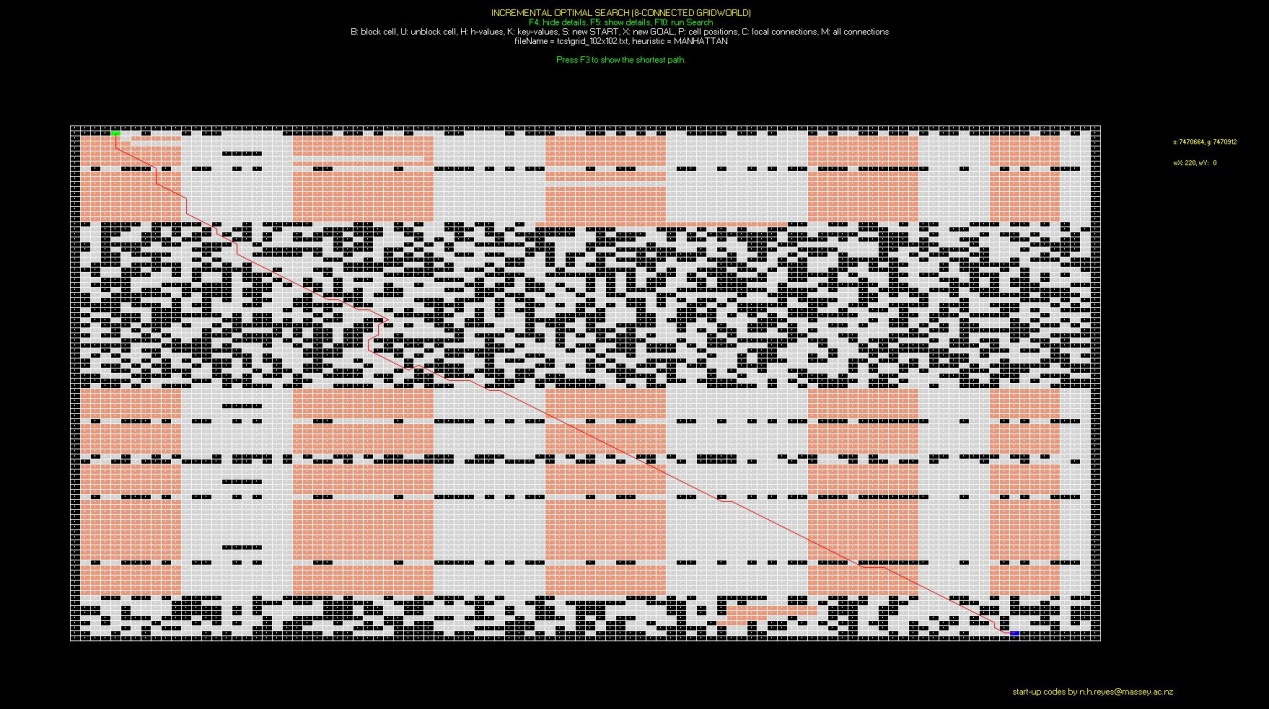


D\*Lite: key values

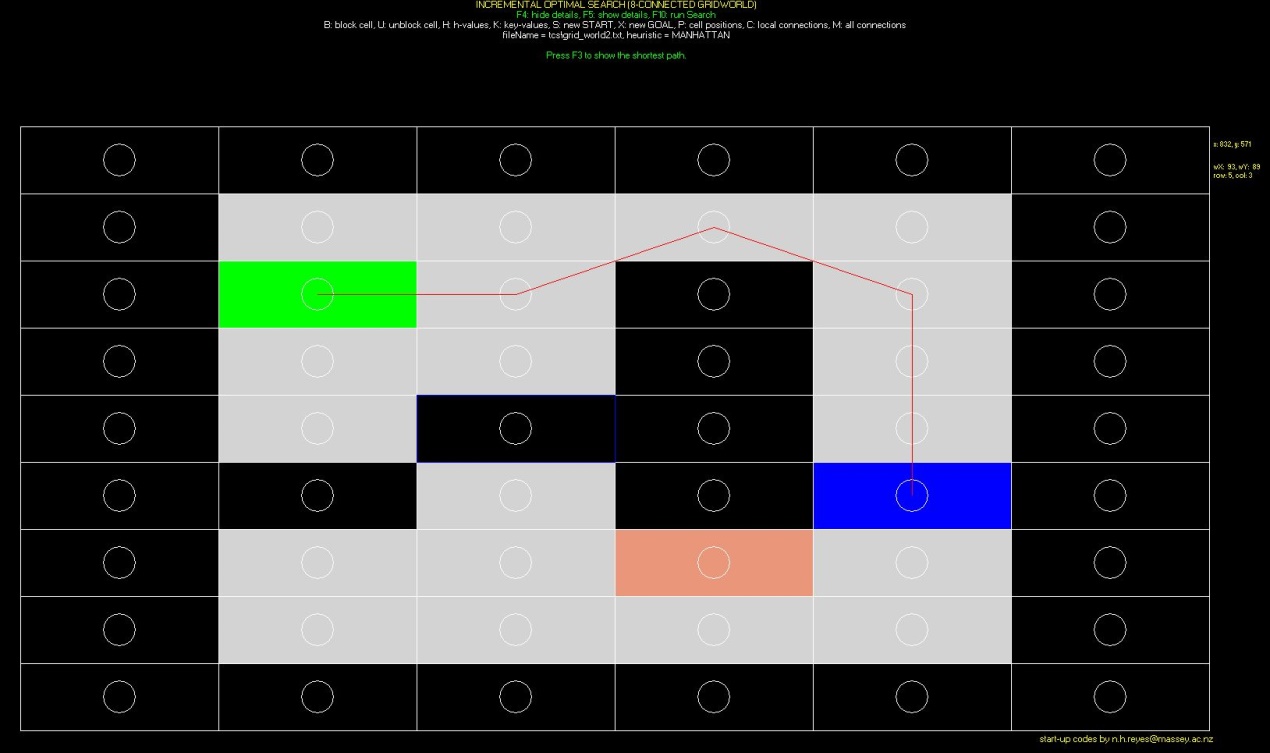


D\*Lite: Path relationship

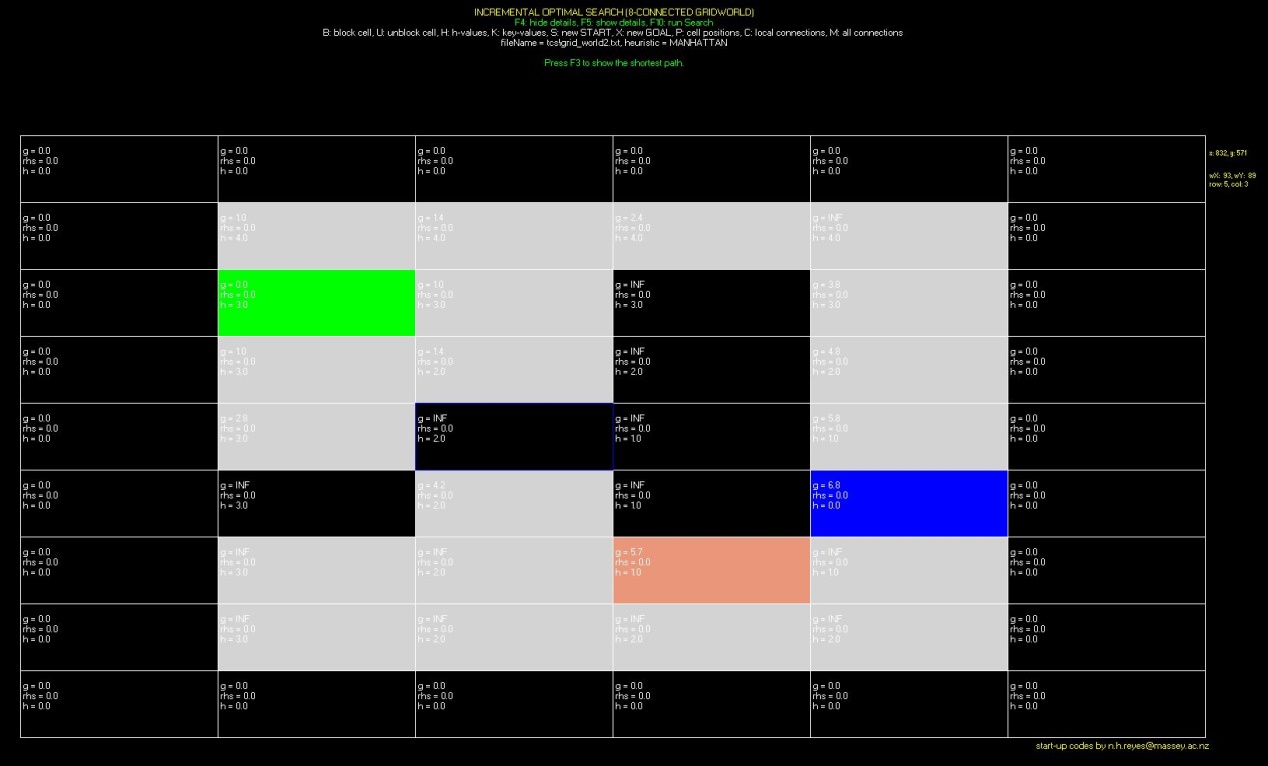
**LPA\*: shortes**t **path**



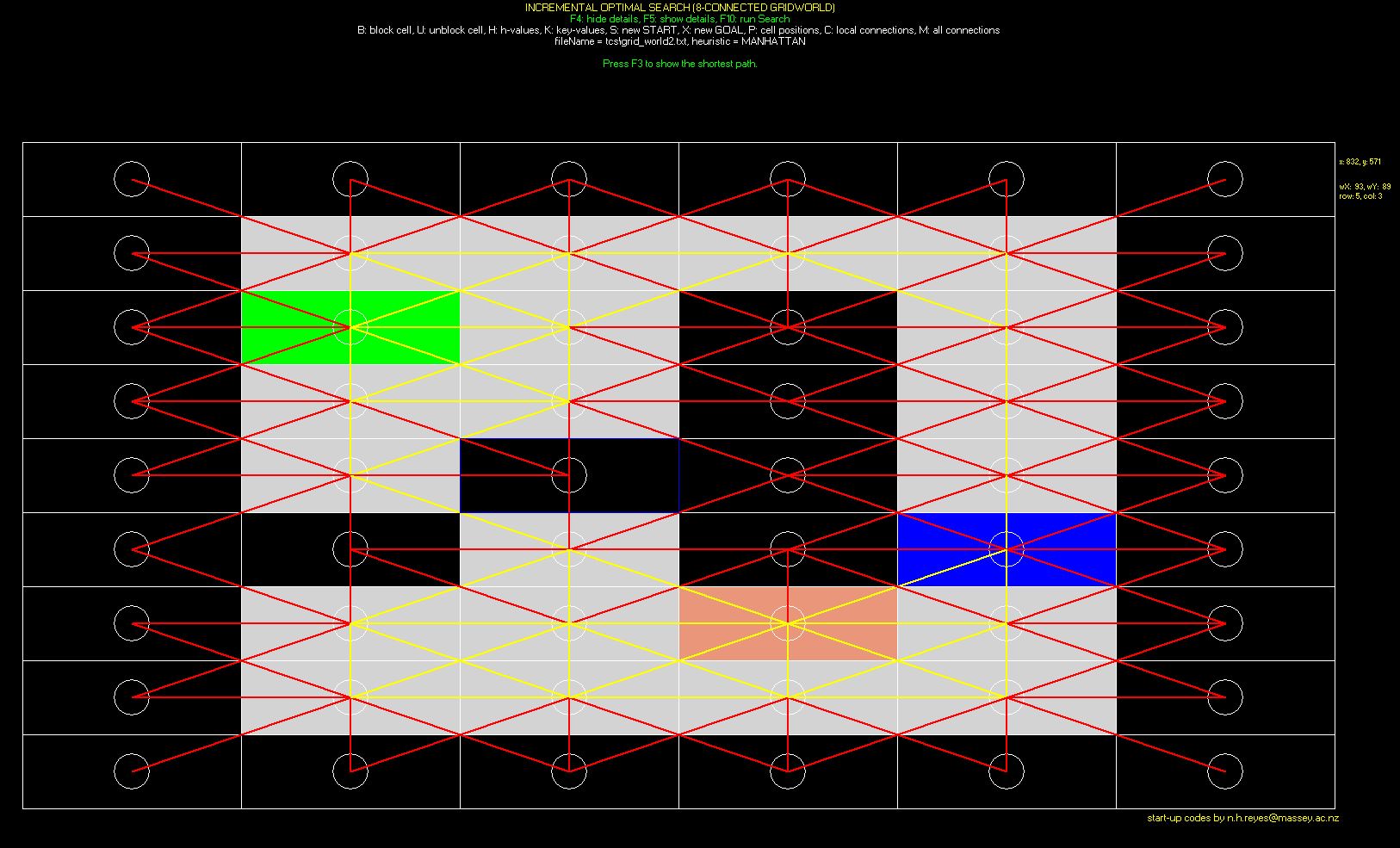
IDA: shortest path



IDA: h-value



IDA: Path



### **1.2 Result Table**

Gridworld: \_\_ grid\_102x102.txt\_\_\_

Heuristic: Euclidean-8 distance

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Algorithm | Max. Queue length | | Path length | | No. of state expansions | | Vertex accesses | | Running Time (msec.) | |
| initial | second | initial | second | initial | second | initial | second | initial | second |
| IDA\* | 251 | 254 | 149.846 | 149.846 | 879 | 878 | 1873 | 1872 | 14920 | 14215 |
| LPA\* | 285 | 287 | 157.604 | 150.362 | 442 | 560 | 705 | 710 | 23 | 28 |
| D\*Lite | 288 | 287 | 155.26 | 149.948 | 412 | 580 | 690 | 1056 | 4 | 21 |

Gridworld: \_\_\_ grid\_102x102.txt\_\_\_\_

Heuristic: Manhattan-8 distance

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Algorithm | Max. Queue length | | Path length | | No. of state expansions | | Vertex accesses | | Running Time (msec.) | |
| initial | second | initial | second | initial | second | initial | second | initial | second |
| IDA\* | 299 | 299 | 163.4 | 163.4 | 495 | 495 | 4081 | 4036 | 25284 | 23371 |
| LPA\* | 225 | 228 | 144.776 | 145.19 | 3772 | 3896 | 3996 | 4008 | 24 | 29 |
| D\*Lite | 197 | 357 | 144.776 | 145.948 | 4279 | 1000 | 4473 | 1702 | 151 | 509 |

### **1.3 Experiments summary**

Through this assignment, we implemented D\*Lite final version, LPA\* and IDA\* algorithms. We found that the running time of IDA\* is much slower than the other two algorithms because it expands vertexes in each step. While LPA\* and D\*Lite are incremental search methods.

Another significant difference is the number of state expansions. If we use Euclidean distance IDA\* uses as twice as the number of state expansions used in LPA\* and D\*Lite. But if we use Manhattan distance, the numbers of state expansions in each algorithm are close.

There are not many differences for other attributes among these three algorithms.

All of them can find the shortest path.

Generally, LPA\* and D\*Lite are better than IDA\*.

## **User’s Guide**

1. Enter Main <gridworld> {MANHATTAN, EUCLIDEAN} {lpa, dlite ida} to start algorithms. E.g. main grid\_Dstar\_journal.txt MANHATTAN lpa
2. Press ENTER**,** when you change any algorithm or cell position need press ENTER first to initialize the calculation.
3. Cannot change map cell position and algorithms during the calculation.

|  |  |  |
| --- | --- | --- |
| Name | Keyboard | Description |
| Show path | F3 | Show shortest path |
| Show detial | F5 | Show g-value, rhs-value and h-value in map |
| Hide detial | F4 | Hide g-value, rhs-value and h-value in map |
| Block Cell | B | Block map cell |
| Unblock Cell | U | Unblock map cell |
| Heuristic value | H | Show heuristic value |
| G value | G | Show g value |
| Key value | K | Show key value |
| Start | S | Change start cell |
| Goal | X | Change goal cell |
| axis value | P | Show cell axis value |
| Cell connection | C | Show map cell connetion relationship |
| All cell connection | M | Show all cell connection relationshop on the map |