Robot Path-Planning Algorithms for Fully Dynamic Shortest Path Problems Report

//署名+Student ID

# Abstract

D start Lite and A start path-planning algorithms used heuristic method to find shortest path, for this purpose experiment will use D\* Lite replanning algorithm compared to A\* algorithm. Frome Euclidean distance and Manhattan-8 distance

# Contents

[1.1 Related Work](#Related_Work)

[A．Data structures of D\* Lite and A\*](#Data_structures)

[B．C++ cod skeleton](#code_skeleton)

[C．Detail of D\* Lite Algorithm](#detail_algorithm)

[1.2 Result Table](#result_table)

[1.3 Experiments summary](#summary)

[1.4 User’s Guide](#user_guide)

## 1.1 Related Work

### A．Data structures of D\* Lite and A\*

#### 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()

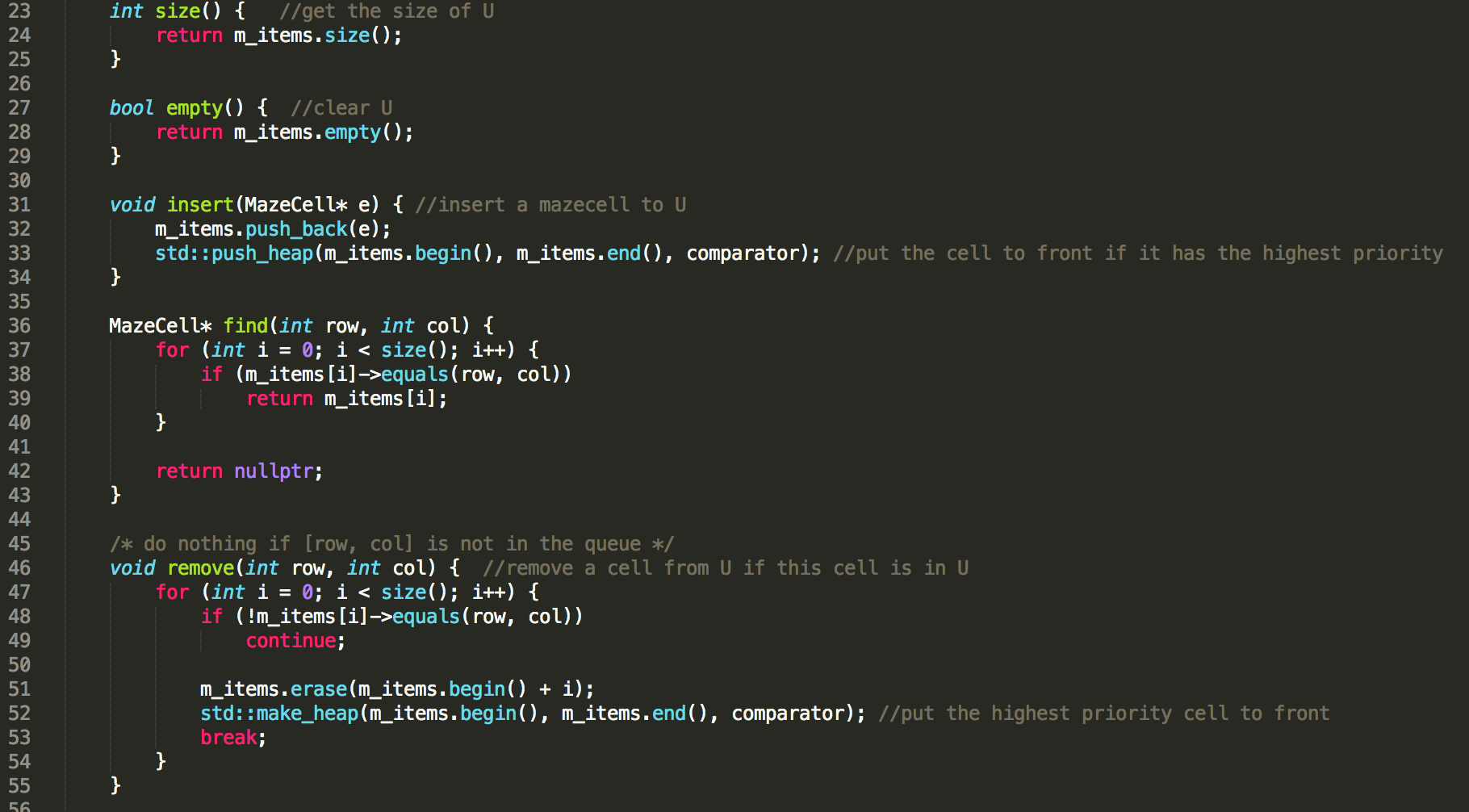
void updateKey();//calle D\*Lite updateKey()

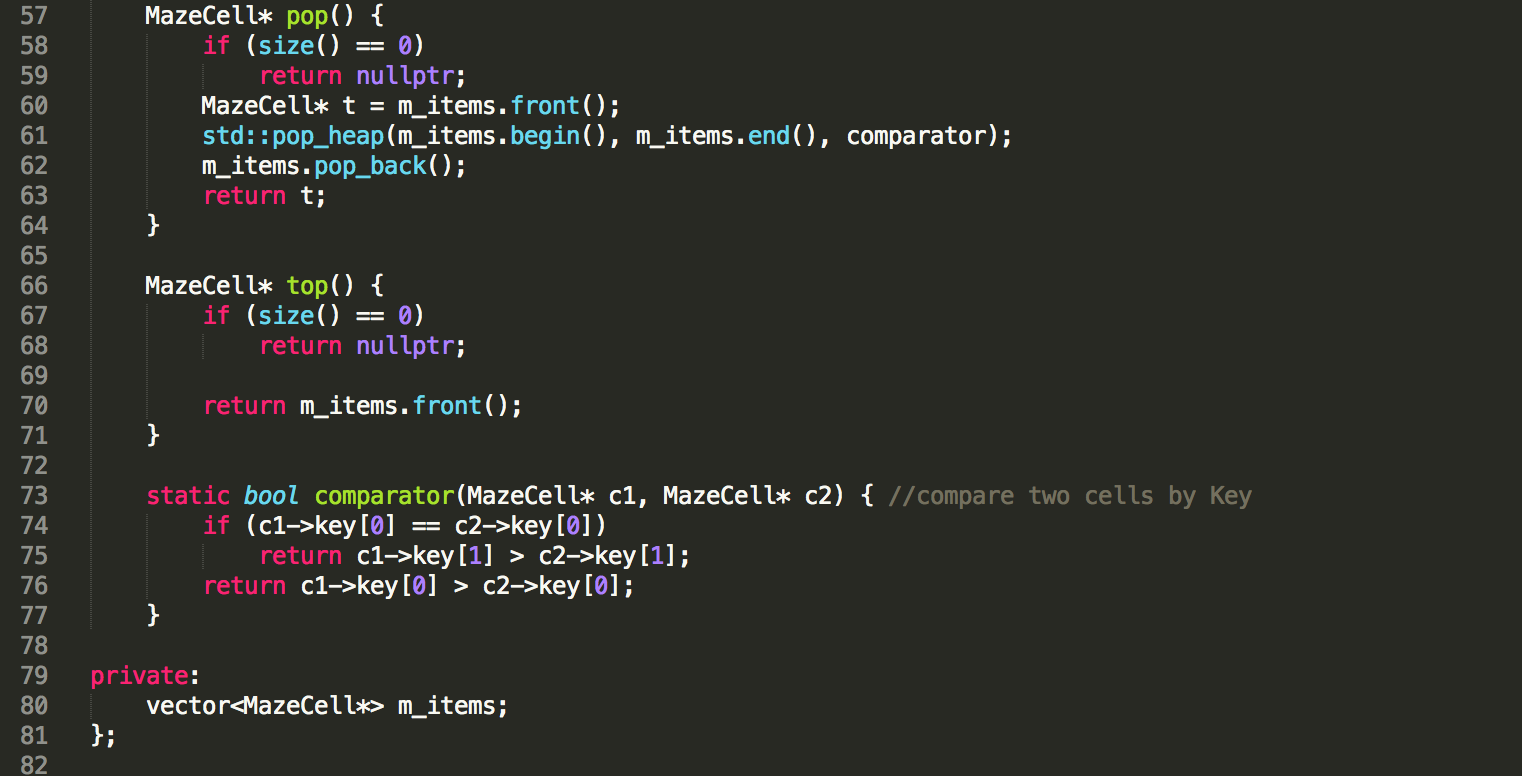
#### Gridworld.h

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:



### B． C++ code skeleton

#### Main.cpp

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

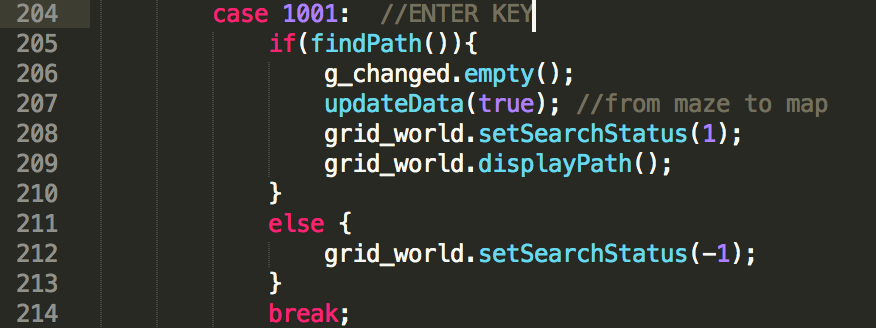
Changes in runsimulation():

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

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

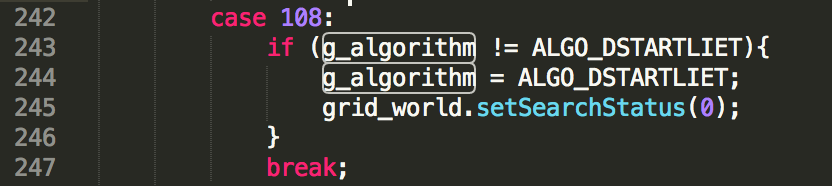
// we define a new member m\_searchStatus(-1,0,1) to define the finding result status.

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



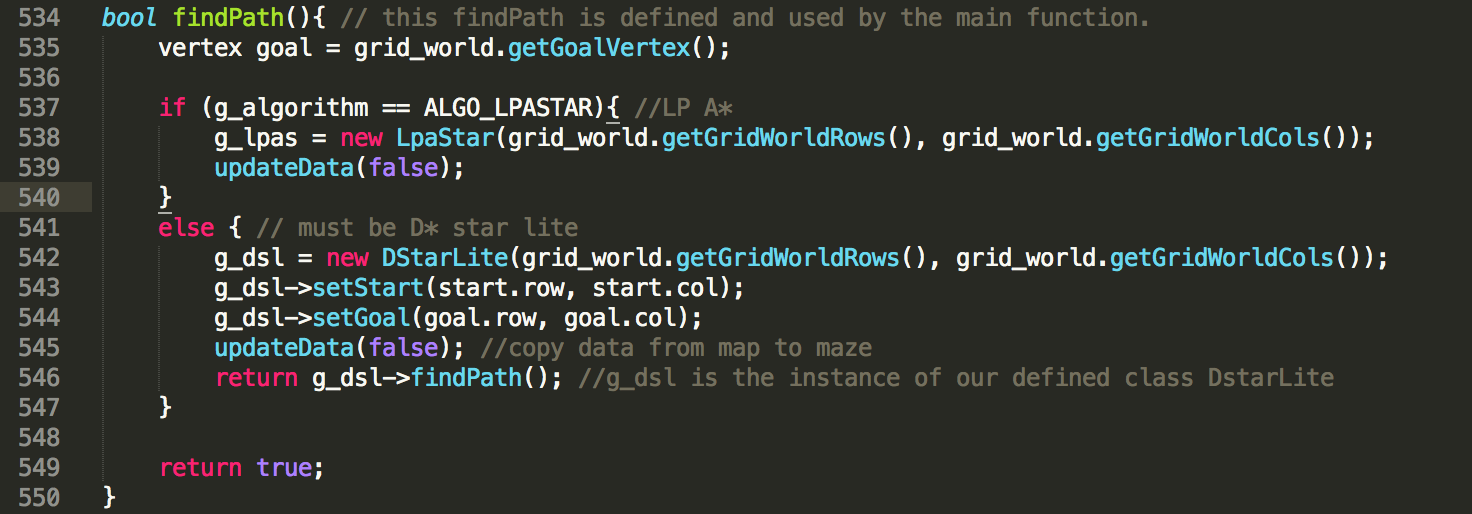
Case 108:

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



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.



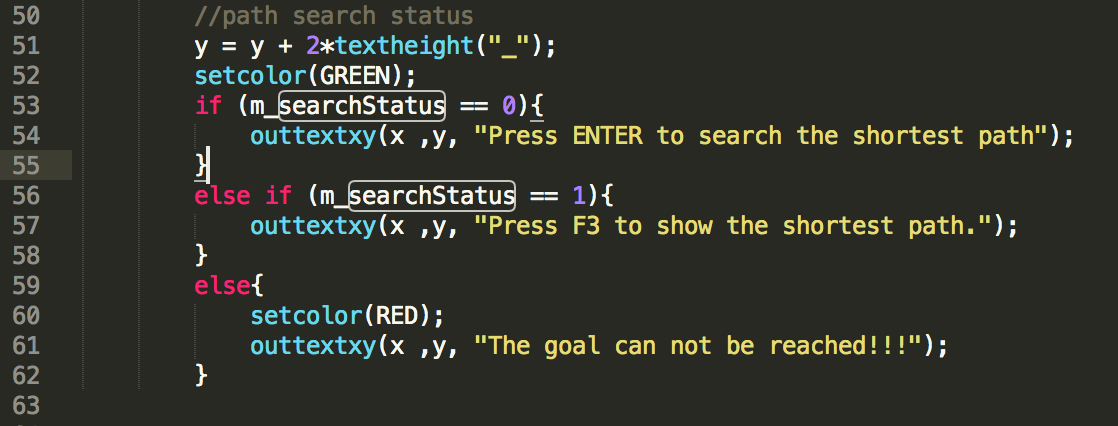
#### Gridword.cpp

In displayHeader(): add m\_searchStatus,

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

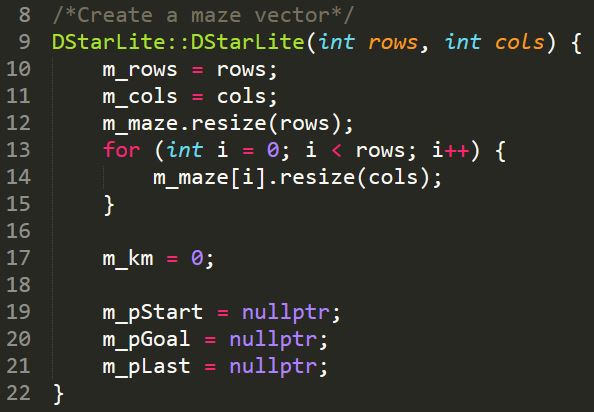
m\_searchStatus=1, show the path;

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

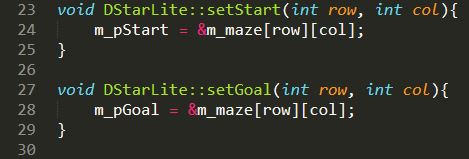


#### Dstart Lite.cpp

Step1 Create grildword maze vector:



Step2 set start and set goal:

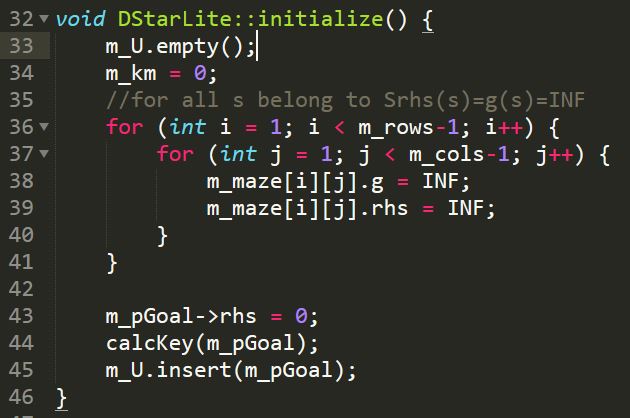


Step3 create initialize function:

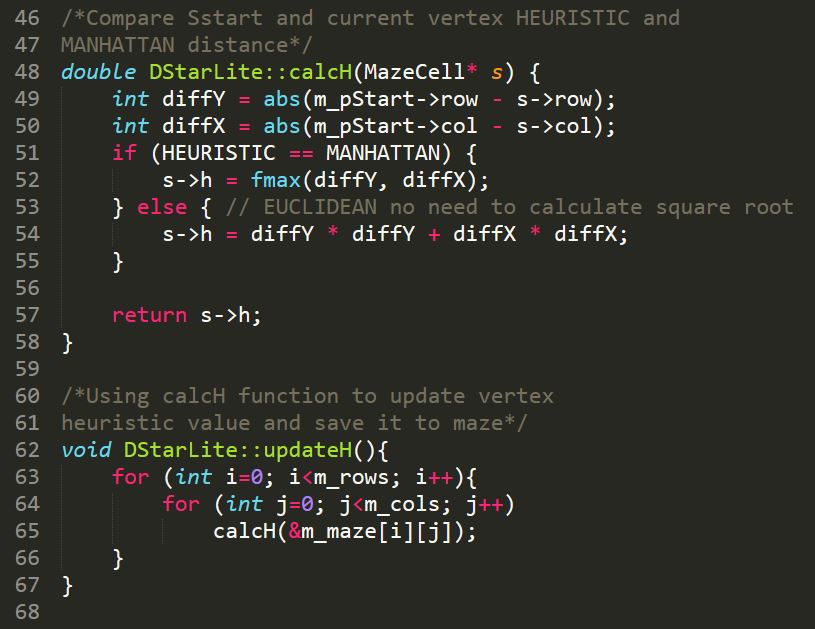
m\_U.empty(); // Empty priority queue

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

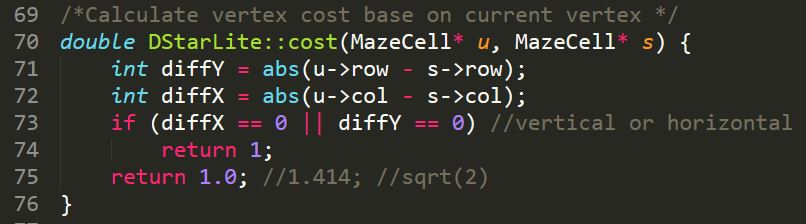
m\_pGoal->rhs = 0;// rhs(Sgoal)=0;



Step4 calculate heuristic value and update heuristic value:



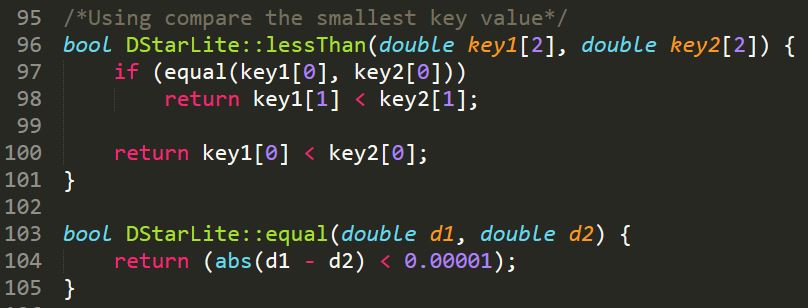
Step5 calculate c(u,s’) value:



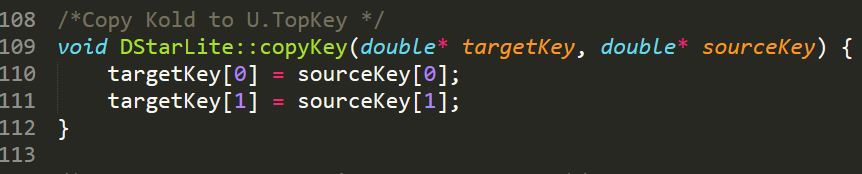
Step6 calculate Key value and update Key value:



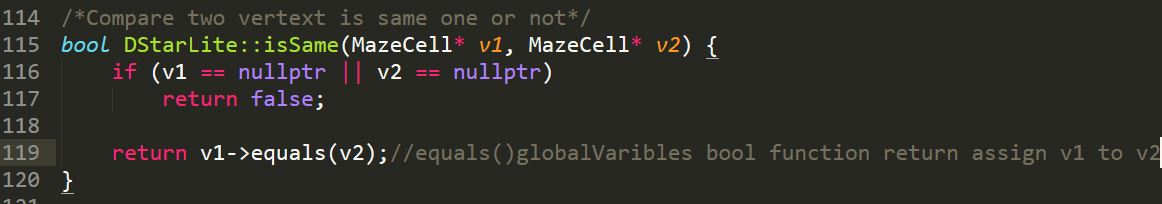
Step7 function for compare the smallest key value:



Step8 copy Kold to U.TopKey:



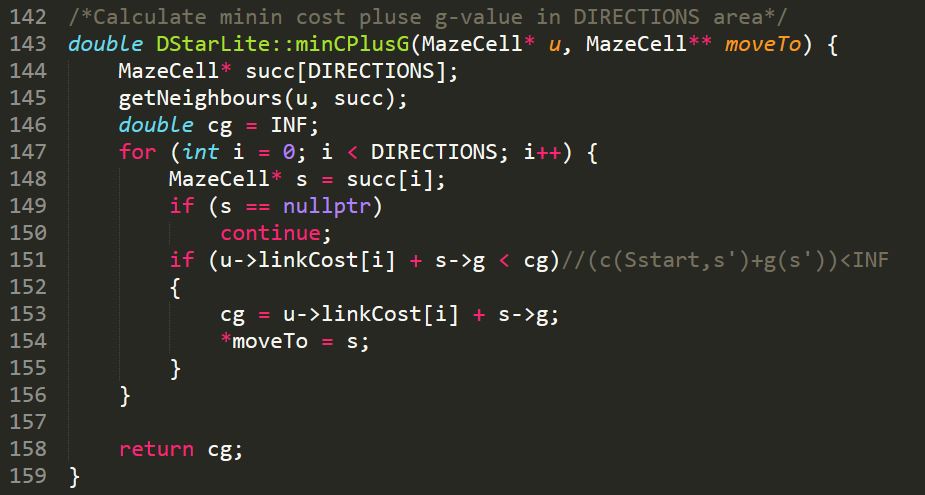
Step9 compare two vertext is same one or not:



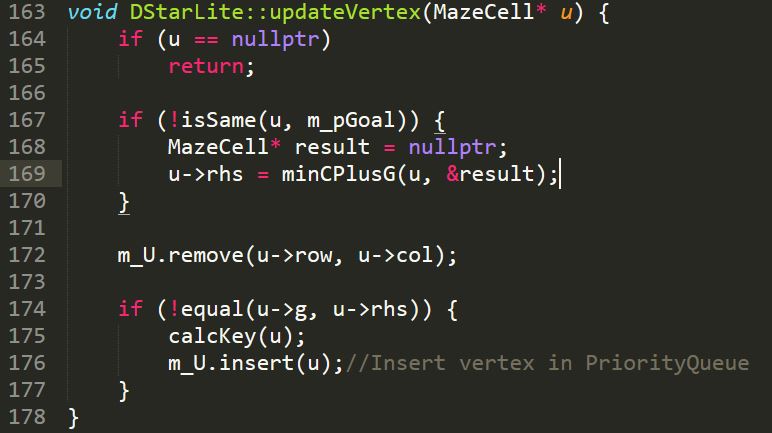
Step10 find all neighbours of current vertex:



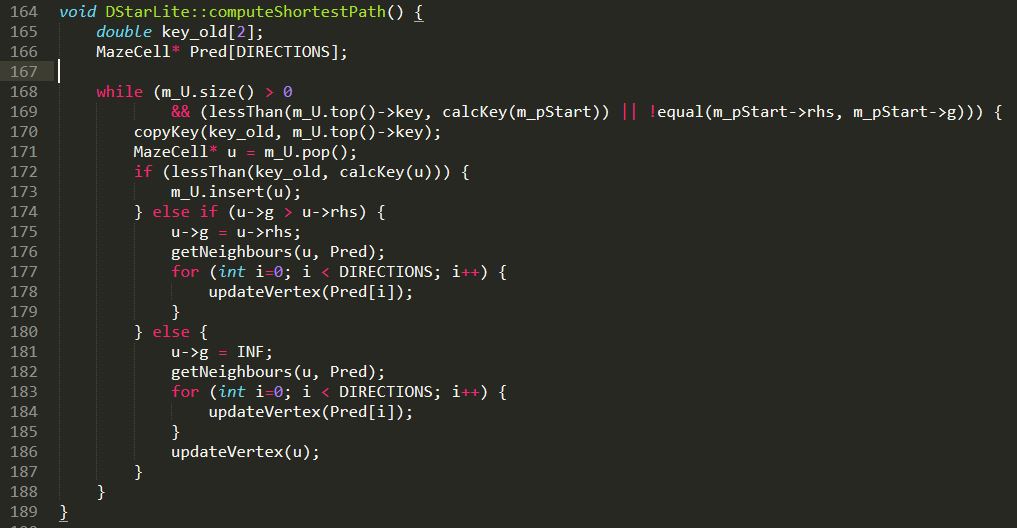
Step11 calculate minin cost pluse g-value:



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



### C． Detail of D\* Lite Algorithm

## 1.2 Result Table

Gridworld: \_\_\_\_\_\_\_\_\_\_\_\_\_

Heuristic: Euclidean distance

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Algorithm** | **Max. Queue length** | **Path length** | **No. of state expansions** | **Vertex accesses** | **Actual Running Time (msec.)** |
| 1 | A\* using the Strict Expanded List | Initial search, second search | Initial search, second search | Initial search, second search | Initial search, second search | Initial search, second search |
| 2 |  |  |  |  |  |  |
| 3 | D\* Lite Final Version | Initial search, second search | Initial search, second search | Initial search, second search | Initial search, second search | Initial search, second search |

Gridworld: \_\_\_\_\_\_\_\_\_\_\_\_\_

Heuristic: Manhattan-8 distance

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Algorithm** | **Max. Queue length** | **Path length** | **No. of state expansions** | **Vertex accesses** | **Actual Running Time (msec.)** |
| 1 | A\* using the Strict Expanded List | Initial search, second search | Initial search, second search | Initial search, second search | Initial search, second search | Initial search, second search |
| 2 |  |  |  |  |  |  |
| 3 | D\* Lite Final Version | Initial search, second search | Initial search, second search | Initial search, second search | Initial search, second search | Initial search, second search |

## 1.3 Experiments summary

## 1.4 User’s Guide

Specify the data structures used in the different algorithms. //指定不同算法使用的数据结构

Provide a skeleton C++ code for each algorithm (more detailed than the pseudocode of the algorithms provided in class). //提供c++代码的框架流程参照伪代码，比伪代码详细

Using the sample gridworld defined in the D\*Lite journal (and also used in the D\*Lite tutorial) Include a snapshot of each algorithm run, indicating graphically the path generated, the map of the g-values, rhs-values, keys, and the vertices expanded. //使用D \* Lite日志中定义的示例gridworld（也用于D \* Lite教程）2. 包括每个算法运行的快照 3. 截图这些值在路径地图中的样式，g值的映射，rhs值，键 ，并扩展顶点（图带不同的点值）。

Tabulate the results of all experiments performed according to the format given. //将程序运行结果制表并列举出所有结果

Discuss briefly the results of all the experiments.//讨论总结实验结果

User’s Guide (simple guide on how to operate the simulation system, short cut keys) //用户指南（关于如何操作模拟系统的简单指南，快捷键）