323.25 Project 8: 8 Puzzle A* Search

Student: Jingshi Liu

Due Date: 5/12/2022

Algorithm Steps for the Project:

```
Step 0:  inFile1, inFile2, outFile1, outFile2 ← open
         initialConfiguration ← get from inFile1
         goalConfiguration ← get from inFile2
         startNode ← Use the constructor to create a AstarNode with initialConfiguration
         goalNode ← Use the constructor to create a AstarNode with goalConfiguration
         Open ← Use the constructor to create a dummy AstarNode
         Close ← Use the constructor to create a dummy AstarNode
Step 1: startNode's gStar ← 0
        startNode's hStar ← computeHstar (StartNode)
        startNode's fStar ← startNode's gStar + startNode's hStar
        OpenInsert (startNode)
Step 2: currentNode ← remove (Open)
Step 3: if (currentNode != null) && isGoalNode (currentNode) // found a solution.
                outFile2 ← "A solution is found!!
                printSolution (currentNode, outFile2)
                exit the program
Step 4: childList ← constructChildList (currentNode)
Step 5: child ← pop (childList)
Step 6: child's gStar ← computeGstar (child)
        child's hStar ← computeHstar (child)
        child's fStar ← child's gStar + child's hStar
        child's parent ← currentNode // back pointer
Step 7:  Spot ← findSpot (Close, child)
Step 8: if Spot's next != null &&  Spot->next->fStar > child's fStar
                CloseDelete (Spot)
Step 9: OpenInsert (child)
Step 10: repeat Step 5 to Step 9 until childList is empty
Step 11: CloseInsert (currentNode)
Step 12: Print "This is Open list:" to outFile1
         printList (Open, outFile1)
         Print "This is CLOSE list:" to outFile1
         printList (Close, outFile1)
         Print up to 20 loops!
Step 13: repeat step 2 to step 12 until currentNode is a goal node or Open is empty.
Step 14: if Open is empty but currentNode is NOT a goal node,
                print error message: "no solution can be found in the search!" to outFile1
Step 15: close all files
```

Source Code:

```
#include <iostream>
#include <fstream>
using namespace std;

class AstarNode{
public:
    string configuration;
    int gStar;
    int hStar;
    int fStar;
    AstarNode* parent = NULL;
    AstarNode* next = NULL;

    AstarNode(string configuration, AstarNode* parent, AstarNode* next){
        this->configuration = configuration;
        this->parent = parent;
        this->next = next;
        this->gStar = 0;
        this->hStar = 0;
        this->fStar = 9999999;
    }

    string printNode(){
        string res = "(" + to_string(fStar) + " :: " + configuration + " :: ";
        if(parent == NULL){
            res += "NULL)";
        }else
            res += parent->configuration + ")";
        return res;
    }
};

class AStar{
public:
AstarNode *startNode;
AstarNode *goalNode;
AstarNode *open;
AstarNode *close;
AstarNode *childList;
int table[9][9] = {
    {0, 1, 2, 1, 2, 3, 2, 3, 4},
    {1, 0, 1, 2, 1 ,2, 3, 2, 3},
    {2, 1, 0, 3, 2, 1, 4, 3, 2},
    {1, 2, 3, 0, 1, 2, 1, 2, 3},
    {2, 1, 2, 1, 0, 1, 2, 1, 2},
    {3, 2, 1, 2, 1, 0, 3, 2, 1},
    {2, 3, 4, 1, 2, 3, 0, 1, 2},
    {3, 2, 3, 2, 1, 2, 1, 0, 1},
    {4, 3, 2, 3, 2 ,1, 2, 1, 0}
    };

AStar(AstarNode* startNode, AstarNode* goalNode){
    this->startNode = startNode;
    this->goalNode = goalNode;
    this->open = new AstarNode("open", NULL, NULL);
    this->close = new AstarNode("close", NULL, NULL);
    this->childList = new AstarNode("childList", NULL, NULL);

}

int computeGstar(AstarNode *node){
    return node->parent->gStar+1;
```

```
}

int computeHstar(AstarNode *node){
    int totalMoves = 0;
    for(int i = 0; i < 9; i++){
        if(node->configuration[i] != goalNode->configuration[i]){
            for(int j = 0; j < 9; j++){
                if(goalNode->configuration[j] == node->configuration[i]){
                    totalMoves += table[i][j];
                    break;
                }
            }
        }
    }

    return totalMoves;

}

bool isGoalNode(AstarNode *node){
    return node->configuration == goalNode->configuration;
}

void openInsert(AstarNode *node){
    if(!open->next){
        open->next = node;
        return;
    }

    AstarNode *spot = open;

    while(spot->next && spot->next->fStar < node->fStar)
        spot = spot->next;

    if(!spot->next){
        spot->next = node;
        return;
    }

    node->next = spot->next;
    spot->next = node;
}

void closeInsert(AstarNode *node){
    AstarNode *spot = close;
    while(spot->next != nullptr && spot->next->fStar < node->fStar)
        spot = spot->next;

    node->next = spot->next;
    spot->next = node;
}

AstarNode* removeOpen(){
    if(open->next == NULL)
        return NULL;
    AstarNode* res = open->next;
    open->next = open->next->next;
    return res;
}

// search if child is in CLOSE list. If true, return the node before child
// else return null
AstarNode* findChildInCloseList(AstarNode *child){
    AstarNode *spot = this->childList;
    while(spot->next){
        if(spot->configuration == child->configuration)
            return spot;
        spot = spot->next;
    }
    return NULL;
}

void closeDelete(AstarNode *spot){
    if(spot->next == NULL)
        return;
    AstarNode* target = spot->next;
    spot->next = spot->next->next;
    target->next = NULL;
}

bool match(string configuration, string configuration2){
    return configuration == configuration2;
}

bool checkAncestors(AstarNode *currentNode){
    AstarNode *ancestor = currentNode->parent;
    while(true){
        if(ancestor->configuration == currentNode->configuration)
            return true;
        if(ancestor->configuration == startNode->configuration)
            return false;
        ancestor = ancestor->parent;
    }
}

AstarNode* constructChildList(AstarNode *currentNode){
    int index;
    for(int i = 0; i < currentNode->configuration.length(); i++){
        if(currentNode->configuration[i] == toascii('0')){
            index = i;
            break;
        }
    }
    int sizeOfArray = sizeof(table[index])/sizeof(table[index][0]);
    string configuration;
    AstarNode *newNode;
    for(int i = 0; i < sizeOfArray; i++){
        if(table[index][i] == 1){
            configuration = "";
            for(int j = 0; j < sizeOfArray; j++){
                if(j == i)
                    configuration += "0";
                else if(j == index)
                    configuration += (char)currentNode->configuration[i];
                else
                    configuration += (char)currentNode->configuration[j];
            }

            newNode = new AstarNode(configuration, currentNode, NULL);
            if(!checkAncestors(newNode)){
                newNode->next = childList->next;
                childList->next = newNode;
            }
        }
    }

    return childList;
}
```

```
AstarNode* popChildList(){
    if(childList->next == NULL)
        return NULL;
    AstarNode* res = childList->next;
    childList->next = childList->next->next;
    res->next = NULL;
    return res;
}

void printList(string list, ofstream& outFile){
    AstarNode* pointer;
    if(list == "open")
        pointer = this->open;
    else
        pointer = this->close;

    while(pointer){
        outFile << pointer->printNode() + " --> ";
        pointer = pointer->next;
    }
    outFile << "\n";
}

void printSolution(AstarNode* currentNode, ofstream& outFile){

    string out = "";
    AstarNode *cur = currentNode;

    while(cur->parent){
        out = cur->printNode() + " --> " + out;
        cur = cur->parent;
    }

    out = "Make moves by following bottom configurations would give the shortest path.
\n " + out;

    outFile << out;
}



};

int main(int argc, const char *argv[])
{
    ifstream inFile1, inFile2;
    ofstream outFile1, outFile2;
    inFile1.open(argv[1]);
    inFile2.open(argv[2]);
    outFile1.open(argv[3]);
    outFile2.open(argv[4]);

    string initalConfiguration = "", goalConfiguration = "";
    string val = "";
    while(inFile1 >> val){
        initalConfiguration += val;
    }
    while(inFile2 >> val)
        goalConfiguration += val;


    AstarNode *startNode = new AstarNode(initalConfiguration, NULL, NULL);
    AstarNode *goalNode = new AstarNode(goalConfiguration, NULL, NULL);
    AStar* aStarSearch = new AStar(startNode, goalNode);

    startNode->gStar = 0;
    startNode->hStar = aStarSearch->computeHstar(startNode);
    startNode->fStar = startNode->gStar + startNode->hStar;
    aStarSearch->openInsert(startNode);

    int debugPrints = 0;
    AstarNode* currentNode;
    AstarNode *child, *spot;
    while(aStarSearch->open->next){

        currentNode = aStarSearch->removeOpen();
        if(aStarSearch->isGoalNode(currentNode)){
            outFile2 << "Found a solution" << endl;
            aStarSearch->printSolution(currentNode, outFile2);
            break;
        }

        //OPEN INSERT:  add currentNode's children that are not visted or has smaller
fStar if visited
        aStarSearch -> constructChildList(currentNode);

        while(aStarSearch->childList->next){
            child = aStarSearch->popChildList();
            child->gStar = aStarSearch->computeGstar(child);
            child->hStar = aStarSearch->computeHstar(child);
            child->fStar = child->gStar + child->hStar;

            spot = aStarSearch->findChildInCloseList(child);
            if(spot && spot->next && spot->next->fStar > child->fStar){
                aStarSearch->closeDelete(spot);
            }
            aStarSearch->openInsert(child);
        }

        aStarSearch->closeInsert(currentNode);
        if(debugPrints < 20){

            outFile1 << "\n\n\nOpen List: ";
            aStarSearch->printList("open", outFile1);

            outFile1 << "\n\nClose List: ";
            aStarSearch->printList("close", outFile1);
            debugPrints++;
        }
    }

    if(aStarSearch->open->next == NULL && !aStarSearch->isGoalNode(currentNode))
        outFile1 << "No Solution Found!!! \n";

    inFile1.close();
    inFile2.close();
    outFile1.close();
    outFile2.close();

    return 0;
}
```

Output:

First Pair:

OutFile1

```
Open List: (9999999 :: open :: NULL) --> (5 :: 283104765 :: 283164705) --> (9 :: 283164075 ::
283164705) --> (9 :: 283164750 :: 283164705) -->

Close List: (9999999 :: close :: NULL) --> (6 :: 283164705 :: NULL) -->

Open List: (9999999 :: open :: NULL) --> (6 :: 203184765 :: 283104765) --> (8 :: 283014765 ::
283104765) --> (8 :: 283140765 :: 283104765) --> (9 :: 283164075 :: 283164705) --> (9 ::
283164750 :: 283164705) -->

Close List: (9999999 :: close :: NULL) --> (5 :: 283104765 :: 283164705) --> (6 :: 283164705 ::
NULL) -->

Open List: (9999999 :: open :: NULL) --> (7 :: 023184765 :: 203184765) --> (8 :: 283014765 ::
283104765) --> (8 :: 283140765 :: 283104765) --> (9 :: 230184765 :: 203184765) --> (9 ::
283164075 :: 283164705) --> (9 :: 283164750 :: 283164705) -->

Close List: (9999999 :: close :: NULL) --> (5 :: 283104765 :: 283164705) --> (6 :: 203184765 ::
283104765) --> (6 :: 283164705 :: NULL) -->

Open List: (9999999 :: open :: NULL) --> (6 :: 123084765 :: 023184765) --> (8 :: 283014765 ::
283104765) --> (8 :: 283140765 :: 283104765) --> (9 :: 230184765 :: 203184765) --> (9 ::
283164075 :: 283164705) --> (9 :: 283164750 :: 283164705) -->

Close List: (9999999 :: close :: NULL) --> (5 :: 283104765 :: 283164705) --> (6 :: 203184765 ::
283104765) --> (6 :: 283164705 :: NULL) --> (7 :: 023184765 :: 203184765) -->

Open List: (9999999 :: open :: NULL) --> (5 :: 123804765 :: 123084765) --> (8 :: 283014765 ::
283104765) --> (8 :: 283140765 :: 283104765) --> (9 :: 123784065 :: 123084765) --> (9 ::
230184765 :: 203184765) --> (9 :: 283164075 :: 283164705) --> (9 :: 283164750 :: 283164705)
-->
```

```
Close List: (9999999 :: close :: NULL) --> (5 :: 283104765 :: 283164705) --> (6 :: 123084765 ::
023184765) --> (6 :: 203184765 :: 283104765) --> (6 :: 283164705 :: NULL) --> (7 ::
023184765 :: 203184765) -->
```

OutFile 2:

```
Found a solution
Make moves by following bottom configurations would give the shortest path.
 (5 :: 283104765 :: 283164705) --> (6 :: 203184765 :: 283104765) --> (7 :: 023184765 ::
203184765) --> (6 :: 123084765 :: 023184765) --> (5 :: 123804765 :: 123084765) -->
```

Pair 2:

OutFile1:

```
Open List: (9999999 :: open :: NULL) --> (11 :: 108357264 :: 158307264) --> (11 :: 158370264 ::
158307264) --> (11 :: 158367204 :: 158307264) --> (13 :: 158037264 :: 158307264) -->


Close List: (9999999 :: close :: NULL) --> (10 :: 158307264 :: NULL) -->


Open List: (9999999 :: open :: NULL) --> (11 :: 158370264 :: 158307264) --> (11 :: 158367204 ::
158307264) --> (12 :: 180357264 :: 108357264) --> (13 :: 158037264 :: 158307264) --> (14 ::
018357264 :: 108357264) -->


Close List: (9999999 :: close :: NULL) --> (10 :: 158307264 :: NULL) --> (11 :: 108357264 ::
158307264) -->


Open List: (9999999 :: open :: NULL) --> (10 :: 158374260 :: 158370264) --> (11 :: 158367204 ::
158307264) --> (12 :: 150378264 :: 158370264) --> (12 :: 180357264 :: 108357264) --> (13 ::
158037264 :: 158307264) --> (14 :: 018357264 :: 108357264) -->


Close List: (9999999 :: close :: NULL) --> (10 :: 158307264 :: NULL) --> (11 :: 158370264 ::
158307264) --> (11 :: 108357264 :: 158307264) -->
```

```
Open List: (9999999 :: open :: NULL) --> (11 :: 158367204 :: 158307264) --> (12 :: 150378264 ::
158370264) --> (12 :: 180357264 :: 108357264) --> (13 :: 158374206 :: 158374260) --> (13 ::
158037264 :: 158307264) --> (14 :: 018357264 :: 108357264) -->


Close List: (9999999 :: close :: NULL) --> (10 :: 158374260 :: 158370264) --> (10 ::
158307264 :: NULL) --> (11 :: 158370264 :: 158307264) --> (11 :: 108357264 :: 158307264) -->



Open List: (9999999 :: open :: NULL) --> (10 :: 158367240 :: 158367204) --> (12 :: 150378264 ::
158370264) --> (12 :: 180357264 :: 108357264) --> (13 :: 158374206 :: 158374260) --> (13 ::
158037264 :: 158307264) --> (14 :: 158367024 :: 158367204) --> (14 :: 018357264 ::
108357264) -->


Close List: (9999999 :: close :: NULL) --> (10 :: 158374260 :: 158370264) --> (10 ::
158307264 :: NULL) --> (11 :: 158367204 :: 158307264) --> (11 :: 158370264 :: 158307264) -->
(11 :: 108357264 :: 158307264) -->



Open List: (9999999 :: open :: NULL) --> (12 :: 150378264 :: 158370264) --> (12 :: 180357264 ::
108357264) --> (13 :: 158360247 :: 158367240) --> (13 :: 158374206 :: 158374260) --> (13 ::
158037264 :: 158307264) --> (14 :: 158367024 :: 158367204) --> (14 :: 018357264 ::
108357264) -->


Close List: (9999999 :: close :: NULL) --> (10 :: 158367240 :: 158367204) --> (10 ::
158374260 :: 158370264) --> (10 :: 158307264 :: NULL) --> (11 :: 158367204 :: 158307264) -->
(11 :: 158370264 :: 158307264) --> (11 :: 108357264 :: 158307264) -->



Open List: (9999999 :: open :: NULL) --> (12 :: 180357264 :: 108357264) --> (13 :: 158360247 ::
158367240) --> (13 :: 158374206 :: 158374260) --> (13 :: 158037264 :: 158307264) --> (14 ::
158367024 :: 158367204) --> (14 :: 018357264 :: 108357264) --> (15 :: 105378264 ::
150378264) -->


Close List: (9999999 :: close :: NULL) --> (10 :: 158367240 :: 158367204) --> (10 ::
158374260 :: 158370264) --> (10 :: 158307264 :: NULL) --> (11 :: 158367204 :: 158307264) -->
(11 :: 158370264 :: 158307264) --> (11 :: 108357264 :: 158307264) --> (12 :: 150378264 ::
158370264) -->



Open List: (9999999 :: open :: NULL) --> (11 :: 187350264 :: 180357264) --> (13 :: 158360247 ::
158367240) --> (13 :: 158374206 :: 158374260) --> (13 :: 158037264 :: 158307264) --> (14 ::
158367024 :: 158367204) --> (14 :: 018357264 :: 108357264) --> (15 :: 105378264 ::
150378264) -->
```

```
Close List: (9999999 :: close :: NULL) --> (10 :: 158367240 :: 158367204) --> (10 ::
158374260 :: 158370264) --> (10 :: 158307264 :: NULL) --> (11 :: 158367204 :: 158307264) -->
(11 :: 158370264 :: 158307264) --> (11 :: 108357264 :: 158307264) --> (12 :: 180357264 ::
108357264) --> (12 :: 150378264 :: 158370264) -->


Open List: (9999999 :: open :: NULL) --> (10 :: 187354260 :: 187350264) --> (13 :: 158360247 ::
158367240) --> (13 :: 158374206 :: 158374260) --> (13 :: 158037264 :: 158307264) --> (14 ::
187305264 :: 187350264) --> (14 :: 158367024 :: 158367204) --> (14 :: 018357264 ::
108357264) --> (15 :: 105378264 :: 150378264) -->

Close List: (9999999 :: close :: NULL) --> (10 :: 158367240 :: 158367204) --> (10 ::
158374260 :: 158370264) --> (10 :: 158307264 :: NULL) --> (11 :: 187350264 :: 180357264) -->
(11 :: 158367204 :: 158307264) --> (11 :: 158370264 :: 158307264) --> (11 :: 108357264 ::
158307264) --> (12 :: 180357264 :: 108357264) --> (12 :: 150378264 :: 158370264) -->


Open List: (9999999 :: open :: NULL) --> (13 :: 187354206 :: 187354260) --> (13 :: 158360247 ::
158367240) --> (13 :: 158374206 :: 158374260) --> (13 :: 158037264 :: 158307264) --> (14 ::
187305264 :: 187350264) --> (14 :: 158367024 :: 158367204) --> (14 :: 018357264 ::
108357264) --> (15 :: 105378264 :: 150378264) -->

Close List: (9999999 :: close :: NULL) --> (10 :: 187354260 :: 187350264) --> (10 ::
158367240 :: 158367204) --> (10 :: 158374260 :: 158370264) --> (10 :: 158307264 :: NULL) -->
(11 :: 187350264 :: 180357264) --> (11 :: 158367204 :: 158307264) --> (11 :: 158370264 ::
158307264) --> (11 :: 108357264 :: 158307264) --> (12 :: 180357264 :: 108357264) --> (12 ::
150378264 :: 158370264) -->


Open List: (9999999 :: open :: NULL) --> (13 :: 158360247 :: 158367240) --> (13 :: 158374206 ::
158374260) --> (13 :: 158037264 :: 158307264) --> (14 :: 187305264 :: 187350264) --> (14 ::
158367024 :: 158367204) --> (14 :: 018357264 :: 108357264) --> (15 :: 105378264 ::
150378264) --> (16 :: 187304256 :: 187354206) --> (16 :: 187354026 :: 187354206) -->

Close List: (9999999 :: close :: NULL) --> (10 :: 187354260 :: 187350264) --> (10 ::
158367240 :: 158367204) --> (10 :: 158374260 :: 158370264) --> (10 :: 158307264 :: NULL) -->
(11 :: 187350264 :: 180357264) --> (11 :: 158367204 :: 158307264) --> (11 :: 158370264 ::
158307264) --> (11 :: 108357264 :: 158307264) --> (12 :: 180357264 :: 108357264) --> (12 ::
150378264 :: 158370264) --> (13 :: 187354206 :: 187354260) -->


Open List: (9999999 :: open :: NULL) --> (13 :: 158374206 :: 158374260) --> (13 :: 158037264 ::
158307264) --> (14 :: 150368247 :: 158360247) --> (14 :: 187305264 :: 187350264) --> (14 ::
158367024 :: 158367204) --> (14 :: 018357264 :: 108357264) --> (15 :: 105378264 ::
```

```
150378264) --> (16 :: 158306247 :: 158360247) --> (16 :: 187304256 :: 187354206) --> (16 ::
187354026 :: 187354206) -->


Close List: (9999999 :: close :: NULL) --> (10 :: 187354260 :: 187350264) --> (10 ::
158367240 :: 158367204) --> (10 :: 158374260 :: 158370264) --> (10 :: 158307264 :: NULL) -->
(11 :: 187350264 :: 180357264) --> (11 :: 158367204 :: 158307264) --> (11 :: 158370264 ::
158307264) --> (11 :: 108357264 :: 158307264) --> (12 :: 180357264 :: 108357264) --> (12 ::
150378264 :: 158370264) --> (13 :: 158360247 :: 158367240) --> (13 :: 187354206 ::
187354260) -->



Open List: (9999999 :: open :: NULL) --> (13 :: 158037264 :: 158307264) --> (14 :: 150368247 ::
158360247) --> (14 :: 187305264 :: 187350264) --> (14 :: 158367024 :: 158367204) --> (14 ::
018357264 :: 108357264) --> (15 :: 105378264 :: 150378264) --> (16 :: 158304276 ::
158374206) --> (16 :: 158374026 :: 158374206) --> (16 :: 158306247 :: 158360247) --> (16 ::
187304256 :: 187354206) --> (16 :: 187354026 :: 187354206) -->


Close List: (9999999 :: close :: NULL) --> (10 :: 187354260 :: 187350264) --> (10 ::
158367240 :: 158367204) --> (10 :: 158374260 :: 158370264) --> (10 :: 158307264 :: NULL) -->
(11 :: 187350264 :: 180357264) --> (11 :: 158367204 :: 158307264) --> (11 :: 158370264 ::
158307264) --> (11 :: 108357264 :: 158307264) --> (12 :: 180357264 :: 108357264) --> (12 ::
150378264 :: 158370264) --> (13 :: 158374206 :: 158374260) --> (13 :: 158360247 ::
158367240) --> (13 :: 187354206 :: 187354260) -->



Open List: (9999999 :: open :: NULL) --> (12 :: 158237064 :: 158037264) --> (14 :: 150368247 ::
158360247) --> (14 :: 187305264 :: 187350264) --> (14 :: 158367024 :: 158367204) --> (14 ::
018357264 :: 108357264) --> (15 :: 105378264 :: 150378264) --> (16 :: 058137264 ::
158037264) --> (16 :: 158304276 :: 158374206) --> (16 :: 158374026 :: 158374206) --> (16 ::
158306247 :: 158360247) --> (16 :: 187304256 :: 187354206) --> (16 :: 187354026 ::
187354206) -->


Close List: (9999999 :: close :: NULL) --> (10 :: 187354260 :: 187350264) --> (10 ::
158367240 :: 158367204) --> (10 :: 158374260 :: 158370264) --> (10 :: 158307264 :: NULL) -->
(11 :: 187350264 :: 180357264) --> (11 :: 158367204 :: 158307264) --> (11 :: 158370264 ::
158307264) --> (11 :: 108357264 :: 158307264) --> (12 :: 180357264 :: 108357264) --> (12 ::
150378264 :: 158370264) --> (13 :: 158037264 :: 158307264) --> (13 :: 158374206 ::
158374260) --> (13 :: 158360247 :: 158367240) --> (13 :: 187354206 :: 187354260) -->



Open List: (9999999 :: open :: NULL) --> (13 :: 158237604 :: 158237064) --> (14 :: 150368247 ::
158360247) --> (14 :: 187305264 :: 187350264) --> (14 :: 158367024 :: 158367204) --> (14 ::
018357264 :: 108357264) --> (15 :: 105378264 :: 150378264) --> (16 :: 058137264 ::
158037264) --> (16 :: 158304276 :: 158374206) --> (16 :: 158374026 :: 158374206) --> (16 ::
158306247 :: 158360247) --> (16 :: 187304256 :: 187354206) --> (16 :: 187354026 ::
187354206) -->
```

```
Close List: (9999999 :: close :: NULL) --> (10 :: 187354260 :: 187350264) --> (10 ::
158367240 :: 158367204) --> (10 :: 158374260 :: 158370264) --> (10 :: 158307264 :: NULL) -->
(11 :: 187350264 :: 180357264) --> (11 :: 158367204 :: 158307264) --> (11 :: 158370264 ::
158307264) --> (11 :: 108357264 :: 158307264) --> (12 :: 158237064 :: 158037264) --> (12 ::
180357264 :: 108357264) --> (12 :: 150378264 :: 158370264) --> (13 :: 158037264 ::
158307264) --> (13 :: 158374206 :: 158374260) --> (13 :: 158360247 :: 158367240) --> (13 ::
187354206 :: 187354260) -->



Open List: (9999999 :: open :: NULL) --> (12 :: 158237640 :: 158237604) --> (14 :: 158207634 ::
158237604) --> (14 :: 150368247 :: 158360247) --> (14 :: 187305264 :: 187350264) --> (14 ::
158367024 :: 158367204) --> (14 :: 018357264 :: 108357264) --> (15 :: 105378264 ::
150378264) --> (16 :: 058137264 :: 158037264) --> (16 :: 158304276 :: 158374206) --> (16 ::
158374026 :: 158374206) --> (16 :: 158306247 :: 158360247) --> (16 :: 187304256 ::
187354206) --> (16 :: 187354026 :: 187354206) -->


Close List: (9999999 :: close :: NULL) --> (10 :: 187354260 :: 187350264) --> (10 ::
158367240 :: 158367204) --> (10 :: 158374260 :: 158370264) --> (10 :: 158307264 :: NULL) -->
(11 :: 187350264 :: 180357264) --> (11 :: 158367204 :: 158307264) --> (11 :: 158370264 ::
158307264) --> (11 :: 108357264 :: 158307264) --> (12 :: 158237064 :: 158037264) --> (12 ::
180357264 :: 108357264) --> (12 :: 150378264 :: 158370264) --> (13 :: 158237604 ::
158237064) --> (13 :: 158037264 :: 158307264) --> (13 :: 158374206 :: 158374260) --> (13 ::
158360247 :: 158367240) --> (13 :: 187354206 :: 187354260) -->



Open List: (9999999 :: open :: NULL) --> (14 :: 158207634 :: 158237604) --> (14 :: 150368247 ::
158360247) --> (14 :: 187305264 :: 187350264) --> (14 :: 158367024 :: 158367204) --> (14 ::
018357264 :: 108357264) --> (15 :: 158230647 :: 158237640) --> (15 :: 105378264 ::
150378264) --> (16 :: 058137264 :: 158037264) --> (16 :: 158304276 :: 158374206) --> (16 ::
158374026 :: 158374206) --> (16 :: 158306247 :: 158360247) --> (16 :: 187304256 ::
187354206) --> (16 :: 187354026 :: 187354206) -->

Close List: (9999999 :: close :: NULL) --> (10 :: 187354260 :: 187350264) --> (10 ::
158367240 :: 158367204) --> (10 :: 158374260 :: 158370264) --> (10 :: 158307264 :: NULL) -->
(11 :: 187350264 :: 180357264) --> (11 :: 158367204 :: 158307264) --> (11 :: 158370264 ::
158307264) --> (11 :: 108357264 :: 158307264) --> (12 :: 158237640 :: 158237604) --> (12 ::
158237064 :: 158037264) --> (12 :: 180357264 :: 108357264) --> (12 :: 150378264 ::
158370264) --> (13 :: 158237604 :: 158237064) --> (13 :: 158037264 :: 158307264) --> (13 ::
158374206 :: 158374260) --> (13 :: 158360247 :: 158367240) --> (13 :: 187354206 ::
187354260) -->



Open List: (9999999 :: open :: NULL) --> (14 :: 150368247 :: 158360247) --> (14 :: 187305264 ::
187350264) --> (14 :: 158367024 :: 158367204) --> (14 :: 018357264 :: 108357264) --> (15 ::
108257634 :: 158207634) --> (15 :: 158270634 :: 158207634) --> (15 :: 158230647 ::
158237640) --> (15 :: 105378264 :: 150378264) --> (16 :: 058137264 :: 158037264) --> (16 ::
158304276 :: 158374206) --> (16 :: 158374026 :: 158374206) --> (16 :: 158306247 ::
158360247) --> (16 :: 187304256 :: 187354206) --> (16 :: 187354026 :: 187354206) --> (17 ::
158027634 :: 158207634) -->
```

```
Close List: (9999999 :: close :: NULL) --> (10 :: 187354260 :: 187350264) --> (10 ::
158367240 :: 158367204) --> (10 :: 158374260 :: 158370264) --> (10 :: 158307264 :: NULL) -->
(11 :: 187350264 :: 180357264) --> (11 :: 158367204 :: 158307264) --> (11 :: 158370264 ::
158307264) --> (11 :: 108357264 :: 158307264) --> (12 :: 158237640 :: 158237604) --> (12 ::
158237064 :: 158037264) --> (12 :: 180357264 :: 108357264) --> (12 :: 150378264 ::
158370264) --> (13 :: 158237604 :: 158237064) --> (13 :: 158037264 :: 158307264) --> (13 ::
158374206 :: 158374260) --> (13 :: 158360247 :: 158367240) --> (13 :: 187354206 ::
187354260) --> (14 :: 158207634 :: 158237604) -->



Open List: (9999999 :: open :: NULL) --> (14 :: 187305264 :: 187350264) --> (14 :: 158367024 ::
158367204) --> (14 :: 018357264 :: 108357264) --> (15 :: 108257634 :: 158207634) --> (15 ::
158270634 :: 158207634) --> (15 :: 158230647 :: 158237640) --> (15 :: 105378264 ::
150378264) --> (16 :: 058137264 :: 158037264) --> (16 :: 158304276 :: 158374206) --> (16 ::
158374026 :: 158374206) --> (16 :: 158306247 :: 158360247) --> (16 :: 187304256 ::
187354206) --> (16 :: 187354026 :: 187354206) --> (17 :: 105368247 :: 150368247) --> (17 ::
158027634 :: 158207634) -->


Close List: (9999999 :: close :: NULL) --> (10 :: 187354260 :: 187350264) --> (10 ::
158367240 :: 158367204) --> (10 :: 158374260 :: 158370264) --> (10 :: 158307264 :: NULL) -->
(11 :: 187350264 :: 180357264) --> (11 :: 158367204 :: 158307264) --> (11 :: 158370264 ::
158307264) --> (11 :: 108357264 :: 158307264) --> (12 :: 158237640 :: 158237604) --> (12 ::
158237064 :: 158037264) --> (12 :: 180357264 :: 108357264) --> (12 :: 150378264 ::
158370264) --> (13 :: 158237604 :: 158237064) --> (13 :: 158037264 :: 158307264) --> (13 ::
158374206 :: 158374260) --> (13 :: 158360247 :: 158367240) --> (13 :: 187354206 ::
187354260) --> (14 :: 150368247 :: 158360247) --> (14 :: 158207634 :: 158237604) -->



Open List: (9999999 :: open :: NULL) --> (14 :: 158367024 :: 158367204) --> (14 :: 018357264 ::
108357264) --> (15 :: 107385264 :: 187305264) --> (15 :: 187365204 :: 187305264) --> (15 ::
108257634 :: 158207634) --> (15 :: 158270634 :: 158207634) --> (15 :: 158230647 ::
158237640) --> (15 :: 105378264 :: 150378264) --> (16 :: 058137264 :: 158037264) --> (16 ::
158304276 :: 158374206) --> (16 :: 158374026 :: 158374206) --> (16 :: 158306247 ::
158360247) --> (16 :: 187304256 :: 187354206) --> (16 :: 187354026 :: 187354206) --> (17 ::
187035264 :: 187305264) --> (17 :: 105368247 :: 150368247) --> (17 :: 158027634 ::
158207634) -->


Close List: (9999999 :: close :: NULL) --> (10 :: 187354260 :: 187350264) --> (10 ::
158367240 :: 158367204) --> (10 :: 158374260 :: 158370264) --> (10 :: 158307264 :: NULL) -->
(11 :: 187350264 :: 180357264) --> (11 :: 158367204 :: 158307264) --> (11 :: 158370264 ::
158307264) --> (11 :: 108357264 :: 158307264) --> (12 :: 158237640 :: 158237604) --> (12 ::
158237064 :: 158037264) --> (12 :: 180357264 :: 108357264) --> (12 :: 150378264 ::
158370264) --> (13 :: 158237604 :: 158237064) --> (13 :: 158037264 :: 158307264) --> (13 ::
158374206 :: 158374260) --> (13 :: 158360247 :: 158367240) --> (13 :: 187354206 ::
187354260) --> (14 :: 187305264 :: 187350264) --> (14 :: 150368247 :: 158360247) --> (14 ::
158207634 :: 158237604) -->
```

OutFile2:

```
Found a solution
Make moves by following bottom configurations would give the shortest path.
 (11 :: 108357264 :: 158307264) --> (14 :: 018357264 :: 108357264) --> (15 :: 318057264 ::
018357264) --> (16 :: 318507264 :: 318057264) --> (17 :: 318570264 :: 318507264) --> (16 ::
318574260 :: 318570264) --> (19 :: 318574206 :: 318574260) --> (22 :: 318574026 ::
318574206) --> (25 :: 318074526 :: 318574026) --> (26 :: 018374526 :: 318074526) --> (25 ::
108374526 :: 018374526) --> (24 :: 178304526 :: 108374526) --> (23 :: 178340526 ::
178304526) --> (24 :: 170348526 :: 178340526) --> (25 :: 107348526 :: 170348526) --> (24 ::
147308526 :: 107348526) --> (23 :: 147328506 :: 147308526) --> (24 :: 147328056 ::
147328506) --> (25 :: 147028356 :: 147328056) --> (24 :: 147208356 :: 147028356) --> (23 ::
147258306 :: 147208356) --> (22 :: 147258360 :: 147258306) -->
```