Report
Introduction to AI　Programming Assignment 1

1. code 內容：

（1）BFS：

從 starting node 開始，尋找可以走的路線，每一次迴圈都只找一層（one step），且每層都要完全找完才能到下一層。

（2）DFS：

從starting node 開始，只要找到可以走的路，先把現在的點存到陣列中，立刻往下走，直到完全走不了了，才回到陣列最末端存的點，並將這個點從陣列中清除，以此類推，直到找到答案才停止。

（3）IDS：

IDS算是DFS的變形，差別在於IDS會限制每一次的搜尋能夠走多遠，因此會有兩層迴圈，第一層用來記錄目前可以走的最大深度，第二層是檢查在此深度下還有沒有點是符合深度但是還沒被走到的，

（4）A*：

分為兩個部分，主函式和副函式，主函式透過frontier陣列來規劃要走的路線，frontier中除了存座標外還有走過的g，分為三個迴圈，第一個迴圈直到找到答案前不會停止，第二個迴圈決定現在處於哪一個frontier，第三個迴圈則是去找到所有這個frontier所接觸到的所有點，裡面兩個迴圈走完之後，找到f(n)= g(n)+h(n)最小的值，並將這個點存進frontier。副函式則是負責計算f(n)，g(n)根據此點從起始點走了多少步來判斷，h(n)則是(dx+dy)/3

（5）IDA*：

IDA*有點像 IDS和 A*的綜合體，分為三個部分，第一個部分類似主函式，第二個部分要用來找threshold，第三個部分和A*的f(n)一樣。第一個部分有兩層迴圈，第一層紀錄目前的threshold，第二層就根據threshold 找出f(n)小於或等於的點，只要符合條件一樣存至frontier，每一次第二層迴圈跑完就進入第二個部分尋找下一個threshold。第二個部分一樣是兩個迴圈，第一層迴圈負責走完所有frontier，第二層迴圈則是找出所有frontier的點所連接的點，並由第三部分的f(n)計算大小，最小的就會成為第一部分的threshold。

2. memory:

```
5
1 1 7 6
(1,1)(2,3)(3,5)(4,7)(5,5)(7,6)
max memory: 5
4
1 1 7 6
(1,1)(2,3)(3,5)(5,6)(6,4)(7,6)
max memory: 160
3
1 1 7 6
(1,1)(2,3)(3,5)(4,7)(5,5)(7,6)
max memory: 6
2
1 1 7 6
(1,1)(2,3)(3,5)(4,7)(5,5)(6,7)(7,5)(5,6)(7,7)(6,5)(7,3)(5,4)(6,6)(7,4)
    (5,3)(6,1)(4,2)(6,3)(7,1)(5,2)(6,4)(7,6)
max memory: 22
1
1 1 7 6
(1,1)(2,3)(3,5)(4,7)(5,5)(7,6)
max memory: 64
```

```
5
0 0 6 6
(0,0)(1,2)(2,4)(4,5)(6,6)
max memory: 4
4
0 0 6 6
(0,0)(1,2)(2,4)(4,5)(6,6)
max memory: 154
3
0 0 6 6
(0,0)(1,2)(2,4)(4,5)(6,6)
max memory: 5
2
0 0 6 6
(0,0)(1,2)(2,4)(3,6)(5,7)(6,5)(7,7)(5,6)(6,4)(7,6)(5,5)(6,7)(7,5)(5,4)(6,6)
max memory: 15
1
0 0 6 6
(0,0)(1,2)(2,4)(4,5)(6,6)
max memory: 38
```

5代表IDA*，4代表A*，3代表IDS，2代表DFS，1代表BFS，A*似乎是最耗費記憶體的，而IDA*和IDS比較節省記憶體。

以big O 表示：

（1）BFS：O(b^d)

（2）DFS：O(bd)

（3）IDS：O(d)

（4）A*：O(b^d)

（5）IDA*：O(d)

3.time:
時間複雜度沒有太大差別，因此不討論。

4.new Heuristic function:

```
int Heuristic_function(int x, int y, int gx, int gy){
    int dx = x - gx;
    int dy = y - gy;
    dx = dx * dx;
    dy = dy * dy;
    int ans = sqrt(dx+dy);
    return ans;
}
```

Heuristic function 目的是要找 node n 大約距離goal的距離，因此大約以兩點距離公式設計Heuristic function。
結果：
結果有些不同，但應該還是optimal

```
4
6 5 2 1
Astar:
(6,5)(4,4)(3,2)(4,0)(2,1)
3
6 5 2 1
IDS:
(6,5)(7,3)(6,1)(4,2)(2,1)

4
0 0 4 4
Astar:
(0,0)(1,2)(3,3)(5,2)(4,4)
3
0 0 4 4
IDS:
(0,0)(1,2)(2,4)(3,6)(4,4)

4
2 5 4 7
Astar:
(2,5)(3,7)(5,6)(3,5)(4,7)
3
2 5 4 7
IDS:
(2,5)(3,7)(4,5)(6,6)(4,7)
```

```cpp
//
//  main.cpp
//  lab1
//
//  Created by 蕭楚澔 on 2020/4/4.
//
#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;

int arr[8][8];

void copyVector(vector<int> v, vector<int> &copy){
    for(int i=0; i<v.size(); i++){
        copy.push_back(v[i]);
    }
}

//print answer
void printAnswerA(vector<int> ans){
    for(int i=(int)ans.size()-2; i>=0; i-=2){
        cout<<"("<<ans[i]<<","<<ans[i+1]<<")";
    }
    cout<<endl;
}

void printAnswerB(vector<int> ans){
    for(int i=0; i<ans.size(); i+=2){
        cout<<"("<<ans[i]<<","<<ans[i+1]<<")";
    }
    cout<<endl;
}

//save new x, new y in vector v, and save flag in f. finally, let
new node be 1
void BFS_pushback(vector<int> &v, vector<int> &f, int nx, int ny,
int flag){
    v.push_back(nx);
    v.push_back(ny);
    arr[nx][ny] = 1;
    f.push_back(flag-2);
    f.push_back(flag-1);
}

void BFS(int sx, int sy, int gx, int gy){
    //vector v saves the node that has already been searched
    vector<int> v;
    //vector f saves the flag of previous node
```

```
    vector<int> f;
    int x = sx;
    int y = sy;
    arr[x][y] = 1;
    int flag = 0;
    while(x != gx || y != gy){
        //this node has no attached nodes anymore, so go to the
next node
        if(arr[x][y] == 2){
            x = v[flag];
            flag++;
            y = v[flag];
            flag++;
        }
        //save the possible node in vector v
        else if(arr[x+1][y+2] == 0 && x+1 < 8 && y+2 < 8){
            BFS_pushback(v, f, x+1, y+2, flag);
        }
        else if(arr[x+2][y+1] == 0 && x+2 < 8 && y+1 < 8){
            BFS_pushback(v, f, x+2, y+1, flag);
        }
        else if(arr[x+1][y-2] == 0 && x+1 < 8 && y-2 >= 0){
            BFS_pushback(v, f, x+1, y-2, flag);
        }
        else if(arr[x+2][y-1] == 0 && x+2 < 8 && y-1 >= 0){
            BFS_pushback(v, f, x+2, y-1, flag);
        }
        else if(arr[x-2][y+1] == 0 && x-2 >= 0 && y+1 < 8){
            BFS_pushback(v, f, x-2, y+1, flag);
        }
        else if(arr[x-1][y+2] == 0 && x-1 >= 0 && y+2 < 8){
            BFS_pushback(v, f, x-1, y+2, flag);
        }
        else if(arr[x-2][y-1] == 0 && x-2 >= 0 && y-1 >= 0){
            BFS_pushback(v, f, x-2, y-1, flag);
        }
        else if(arr[x-1][y-2] == 0 && x-1 >= 0 && y-2 >= 0){
            BFS_pushback(v, f, x-1, y-2, flag);
        }
        //this node has no attached nodes anymore, x and y will
change in next loop
        else{
            arr[x][y] = 2;
        }
        //if true, find goal x and goal y, break the loop
        if(v.size() >= 2 && v[v.size()-2] == gx && v[v.size()-1]
== gy){
            break;
        }
    }
```

```
    //use vector f to find previous node until we reach the
starting point
    vector<int> ans;
    ans.push_back(gx);
    ans.push_back(gy);
    flag-=2;
    while(f[flag] >= 0){
        ans.push_back(v[flag]);
        ans.push_back(v[flag+1]);
        flag = f[flag];
    }
    ans.push_back(v[flag]);
    ans.push_back(v[flag+1]);
    ans.push_back(sx);
    ans.push_back(sy);
    printAnswerA(ans);
}

void DFS(int sx, int sy, int gx, int gy){
    vector<int> v;
    int x = sx;
    int y = sy;
    arr[x][y] = 1;
    //don't stop until find the goal state
    while(x != gx || y!= gy){
        //can't find any other point within this node(by last
loop)
        if(arr[x][y] == 2){
            y = v[v.size()-1];
            v.pop_back();
            x = v[v.size()-1];
            v.pop_back();
        }
        //find other point to go deeper
        else if(arr[x+1][y+2] == 0 && x+1 < 8 && y+2 < 8){
            v.push_back(x);
            v.push_back(y);
            x = x+1;
            y = y+2;
            arr[x][y] = 1;
        }
        else if(arr[x+2][y+1] == 0 && x+2 < 8 && y+1 < 8){
            v.push_back(x);
            v.push_back(y);
            x = x+2;
            y = y+1;
            arr[x][y] = 1;
        }
        else if(arr[x+1][y-2] == 0 && x+1 < 8 && y-2 >= 0){
            v.push_back(x);
            v.push_back(y);
            x = x+1;
```

```
            y = y-2;
            arr[x][y] = 1;
        }
        else if(arr[x+2][y-1] == 0 && x+2 < 8 && y-1 >= 0){
            v.push_back(x);
            v.push_back(y);
            x = x+2;
            y = y-1;
            arr[x][y] = 1;
        }
        else if(arr[x-2][y+1] == 0 && x-2 >= 0 && y+1 < 8){
            v.push_back(x);
            v.push_back(y);
            x = x-2;
            y = y+1;
            arr[x][y] = 1;
        }
        else if(arr[x-1][y+2] == 0 && x-1 >= 0 && y+2 < 8){
            v.push_back(x);
            v.push_back(y);
            x = x-1;
            y = y+2;
            arr[x][y] = 1;
        }
        else if(arr[x-2][y-1] == 0 && x-2 >= 0 && y-1 >= 0){
            v.push_back(x);
            v.push_back(y);
            x = x-2;
            y = y-1;
            arr[x][y] = 1;
        }
        else if(arr[x-1][y-2] == 0 && x-1 >= 0 && y-2 >= 0){
            v.push_back(x);
            v.push_back(y);
            x = x-1;
            y = y-2;
            arr[x][y] = 1;
        }
        //can't find any other point within this node
        else{
            arr[x][y] = 2;
        }
    }

    v.push_back(gx);
    v.push_back(gy);
    printAnswerB(v);
}

void IDS_pushback(vector<int> &v,int nx, int ny){
    arr[nx][ny] = 1;
    v.push_back(nx);
```

```
    v.push_back(ny);
}

void IDS(int sx, int sy, int gx, int gy){
    vector<int> v;
    int x = sx;
    int y = sy;
    int deep = 1;
    int iter = 0;
    bool flag = true;
    while(x != gx || y != gy){
        v.clear();
        x = sx;
        y = sy;
        iter = 0;
        flag = true;
        //initialize all array when depth+1
        for(int i=0; i<8; i++){
            for(int j=0; j<8; j++){
                arr[i][j] = 0;
            }
        }
        //every time loops represent a depth
        while(1){
            //find further node, and stop if iter < deep
            if(arr[x+1][y+2] == 0 && x+1 < 8 && y+2 < 8 && (iter <
deep)){
                IDS_pushback(v, x, y);
                x = x+1;
                y = y+2;
                arr[x][y] = 1;
                iter++;
            }
            else if(arr[x+2][y+1] == 0 && x+2 < 8 && y+1 < 8 &&
(iter < deep)){
                IDS_pushback(v, x, y);
                x = x+2;
                y = y+1;
                arr[x][y] = 1;
                iter++;
            }
            else if(arr[x+1][y-2] == 0 && x+1 < 8 && y-2 >= 0 &&
(iter < deep)){
                IDS_pushback(v, x, y);
                x = x+1;
                y = y-2;
                arr[x][y] = 1;
                iter++;
            }
            else if(arr[x+2][y-1] == 0 && x+2 < 8 && y-1 >= 0 &&
(iter < deep)){
                IDS_pushback(v, x, y);
```

```
                x = x+2;
                y = y-1;
                arr[x][y] = 1;
                iter++;
            }
            else if(arr[x-2][y+1] == 0 && x-2 >= 0 && y+1 < 8 &&
(iter < deep)){
                IDS_pushback(v, x, y);
                x = x-2;
                y = y+1;
                arr[x][y] = 1;
                iter++;
            }
            else if(arr[x-1][y+2] == 0 && x-1 >= 0 && y+2 < 8 &&
(iter < deep)){
                IDS_pushback(v, x, y);
                x = x-1;
                y = y+2;
                arr[x][y] = 1;
                iter++;
            }
            else if(arr[x-2][y-1] == 0 && x-2 >= 0 && y-1 >= 0 &&
(iter < deep)){
                IDS_pushback(v, x, y);
                x = x-2;
                y = y-1;
                arr[x][y] = 1;
                iter++;
            }
            else if(arr[x-1][y-2] == 0 && x-1 >= 0 && y-2 >= 0 &&
(iter < deep)){
                IDS_pushback(v, x, y);
                x = x-1;
                y = y-2;
                arr[x][y] = 1;
                iter++;
            }
            //if the root can't find any node means this depth
can't reach the goal state
            else if(x == sx && y == sy){
                for(int i=0; i<8; i++){
                    for(int j=0; j<8; j++){
                        arr[i][j] = 0;
                    }
                }
                break;
            }
            //this part means this node can't find any further
node
            else{
                y = v[v.size()-1];
                v.pop_back();
```

```
                x = v[v.size()-1];
                v.pop_back();
                iter--;
            }
            //reach the goal state
            if(x == gx && y == gy){
                break;
            }
        }
        deep++;
    }
    vector<int> ans;
    v.push_back(gx);
    v.push_back(gy);
    printAnswerB(v);
}

//the h(x) function
float Heuristic_function(int ax, int ay, int gx, int gy){
    int dx = ax - gx;
    int dy = ay - gy;
    if(dx < 0){
        dx = -dx;
    }
    if(dy < 0){
        dy = -dy;
    }
    float ans = (dx + dy)/3;
    return ans;
}

//nx = new x, ny = new y, cx = current x, cy = current y, gx =
goal x, gy = goal y, g = steps
void Astar_pushback(vector<int> &v, int nx, int ny, int cx, int
cy, int gx, int gy, int g){
    v.push_back(nx);
    v.push_back(ny);
    arr[nx][ny] = 1;
    //the square root of 5 is almost 2.23, g represents steps, so
g(x) = g * 2.23, and f(x) = g(x)+h(x)
    int h = Heuristic_function(nx, ny, gx, gy);
    v.push_back(g);
    g = g * 2.23;
    int f = g + h;
    v.push_back(f);
    v.push_back(cx);
    v.push_back(cy);
}

void Astar(int sx, int sy, int gx, int gy){
    vector<int> frontier;
    frontier.push_back(sx);
```

```
    frontier.push_back(sy);
    int g = 0;
    frontier.push_back(g);
    int x = sx;
    int y = sy;
    //vector temp saves every node the frontier will reach
    vector<int> temp;
    vector<int> pre;
    while(x != gx || y != gy){
        for(int i=0; i<frontier.size(); i+=3){
            x = frontier[i];
            y = frontier[i+1];
            g = frontier[i+2];
            g++;
            //initialize the array
            for(int i=0; i<8; i++){
                for(int j=0; j<8; j++){
                    arr[i][j] = 0;
                }
            }
            //but the node that have been reached already should be departed
            for(int j=0; j<frontier.size(); j+=3){
                arr[frontier[j]][frontier[j+1]] = 2;
            }

            while(1){
                if(arr[x+1][y+2] == 0 && x+1 < 8 && y+2 < 8){
                    Astar_pushback(temp, x+1, y+2, x, y, gx, gy, g);
                }
                else if(arr[x+2][y+1] == 0 && x+2 < 8 && y+1 < 8){
                    Astar_pushback(temp, x+2, y+1, x, y, gx, gy, g);
                }
                else if(arr[x+1][y-2] == 0 && x+1 < 8 && y-2 >= 0){
                    Astar_pushback(temp, x+1, y-2, x, y, gx, gy, g);
                }
                else if(arr[x+2][y-1] == 0 && x+2 < 8 && y-1 >= 0){
                    Astar_pushback(temp, x+2, y-1, x, y, gx, gy, g);
                }
                else if(arr[x-2][y+1] == 0 && x-2 >= 0 && y+1 < 8){
                    Astar_pushback(temp, x-2, y+1, x, y, gx, gy, g);
                }
                else if(arr[x-1][y+2] == 0 && x-1 >= 0 && y+2 < 8){
```

```
                    Astar_pushback(temp, x-1, y+2, x, y, gx, gy,
g);
                }
                else if(arr[x-2][y-1] == 0 && x-2 >= 0 && y-1 >=
0){
                    Astar_pushback(temp, x-2, y-1, x, y, gx, gy,
g);
                }
                else if(arr[x-1][y-2] == 0 && x-1 >= 0 && y-2 >=
0){
                    Astar_pushback(temp, x-1, y-2, x, y, gx, gy,
g);
                }
                else{
                    break;
                }
            }
        }
        //find the minimum path, and save in frontier
        int min = temp[3];
        int px, py;
        for(int j=0; j<temp.size(); j+=6){
            if(temp[j+2] < min){
                x = temp[j];
                y = temp[j+1];
                g = temp[j+3];
                min = temp[j+2];
                px = temp[j+4];
                py = temp[j+5];
            }
        }
        frontier.push_back(x);
        frontier.push_back(y);
        frontier.push_back(g);
        pre.push_back(px);
        pre.push_back(py);
        temp.clear();
    }


    for(int j=0; j<frontier.size(); j++){
        frontier[j] = frontier[j+3];
    }
    frontier.pop_back();
    frontier.pop_back();
    frontier.pop_back();

    //use vector pre to find the previous node
    vector<int> ans;
    int s = (int)frontier.size();
    x = frontier[frontier.size()-3];
    y = frontier[frontier.size()-2];
```

```
    while(x != sx || y != sy){
        ans.push_back(x);
        ans.push_back(y);
        int t = 2*(s/3);
        x = pre[t-2];
        y = pre[t-1];
        for(int k=0; k<frontier.size(); k++){
            if(frontier[k] == x && frontier[k+1] == y){
                s = k;
            }
        }
        s += 3;
    }
    ans.push_back(sx);
    ans.push_back(sy);
    printAnswerA(ans);
}

float f_function(int g, int nx, int ny, int gx, int gy){
    int gg = g * 2.23;
    float h = Heuristic_function(nx, ny, gx, gy);
    int f = gg + h;
    return f;
}

void findThreshold_pushback(vector<float> &v, int nx, int ny, int
gx, int gy, int g){
    arr[nx][ny] = 1;
    float f = f_function(g, nx, ny, gx, gy);
    v.push_back(f);
}

int findThreshold(vector<int> v, int gx, int gy){
    vector<float> temp;
    int x;
    int y;
    int g;
    for(int i=0; i<v.size(); i+=3){
        for(int j=0; j<8; j++){
            for(int k=0; k<8; k++){
                arr[i][j] = 0;
            }
        }
        for(int j=0; j<v.size(); j+=3){
            arr[v[j]][v[j+1]] = 2;
        }
        x = v[i];
        y = v[i+1];
        g = v[i+2]+1;
        while(1){
            if(arr[x+1][y+2] == 0 && x+1 < 8 && y+2 < 8){
                findThreshold_pushback(temp, x+1, y+2, gx, gy, g);
```

```
            }
            else if(arr[x+2][y+1] == 0 && x+2 < 8 && y+1 < 8){
                findThreshold_pushback(temp, x+2, y+1, gx, gy, g);
            }
            else if(arr[x+1][y-2] == 0 && x+1 < 8 && y-2 >= 0){
                findThreshold_pushback(temp, x+1, y-2, gx, gy, g);
            }
            else if(arr[x+2][y-1] == 0 && x+2 < 8 && y-1 >= 0){
                findThreshold_pushback(temp, x+2, y-1, gx, gy, g);
            }
            else if(arr[x-2][y+1] == 0 && x-2 >= 0 && y+1 < 8){
                findThreshold_pushback(temp, x-2, y+1, gx, gy, g);
            }
            else if(arr[x-1][y+2] == 0 && x-1 >= 0 && y+2 < 8){
                findThreshold_pushback(temp, x-1, y+2, gx, gy, g);
            }
            else if(arr[x-2][y-1] == 0 && x-2 >= 0 && y-1 >= 0){
                findThreshold_pushback(temp, x-2, y-1, gx, gy, g);
            }
            else if(arr[x-1][y-2] == 0 && x-1 >= 0 && y-2 >= 0){
                findThreshold_pushback(temp, x-1, y-2, gx, gy, g);
            }
            else{
                break;
            }
        }
    }
    int min = temp[0];
    for(int i=0; i<temp.size(); i+=3){
        if(min < temp[i]){
            min = temp[i];
        }
    }
    return min;
}

void IDAstar_pushback(vector<int> &v, vector<int> &frontier, int
cx, int cy, int nx, int ny, int g){
    v.push_back(cx);
    v.push_back(cy);
    arr[nx][ny] = 1;
    v.push_back(g);
    frontier.push_back(nx);
    frontier.push_back(ny);
    frontier.push_back(g+1);
}

void IDAstar(int sx, int sy, int gx, int gy){
    int x = sx;
    int y = sy;
    vector<int> frontier;
    vector<int> v;
```

```
    vector<int> ans;
    float deep = Heuristic_function(sx, sy, gx, gy);
    while(x != gx || y != gy){
        for(int i=0; i<8; i++){
            for(int j=0; j<8; j++){
                arr[i][j] = 0;
            }
        }
        arr[sx][sy] = 1;
        int g = 0;
        x = sx;
        y = sy;
        v.clear();
        frontier.clear();
        frontier.push_back(sx);
        frontier.push_back(sy);
        frontier.push_back(g);
        while(x != gx || y != gy){
            if(arr[x+1][y+2] == 0 && x+1 < 8 && y+2 < 8 &&
f_function(g+1, x+1, y+2, gx, gy) <= deep){
                IDAstar_pushback(v, frontier, x, y, x+1, y+2, g);
                g++;
                x = x+1;
                y = y+2;
            }
            else if(arr[x+2][y+1] == 0 && x+2 < 8 && y+1 < 8 &&
f_function(g+1, x+2, y+1, gx, gy) <= deep){
                IDAstar_pushback(v, frontier, x, y, x+2, y+1, g);
                g++;
                x = x+2;
                y = y+1;
            }
            else if(arr[x+1][y-2] == 0 && x+1 < 8 && y-2 >= 0 &&
f_function(g+1, x+1, y-2, gx, gy) <= deep){
                IDAstar_pushback(v, frontier, x, y, x+1, y-2, g);
                g++;
                x = x+1;
                y = y-2;
            }
            else if(arr[x+2][y-1] == 0 && x+2 < 8 && y-1 >= 0 &&
f_function(g+1, x+2, y-1, gx, gy) <= deep){
                IDAstar_pushback(v, frontier, x, y, x+2, y-1, g);
                g++;
                x = x+2;
                y = y-1;
            }
            else if(arr[x-2][y+1] == 0 && x-2 >= 0 && y+1 < 8 &&
f_function(g+1, x-2, y+1, gx, gy) <= deep){
                IDAstar_pushback(v, frontier, x, y, x-2, y+1, g);
                g++;
                x = x-2;
                y = y+1;
```

```
            }
            else if(arr[x-1][y+2] == 0 && x-1 >= 0 && y+2 < 8 &&
f_function(g+1, x-1, y+2, gx, gy) <= deep){
                IDAstar_pushback(v, frontier, x, y, x-1, y+2, g);
                g++;
                x = x-1;
                y = y+2;
            }
            else if(arr[x-2][y-1] == 0 && x-2 >= 0 && y-1 >= 0 &&
f_function(g+1, x-2, y-1, gx, gy) <= deep){
                IDAstar_pushback(v, frontier, x, y, x-2, y-1, g);
                g++;
                x = x-2;
                y = y-1;
            }
            else if(arr[x-1][y-2] == 0 && x-1 >= 0 && y-2 >= 0 &&
f_function(g+1, x-1, y-2, gx, gy) <= deep){
                IDAstar_pushback(v, frontier, x, y, x-1, y-2, g);
                g++;
                x = x-1;
                y = y-2;
            }
            else if(x == sx && y == sy){
                break;
            }
            else{
                g--;
                v.pop_back();
                y = v[v.size()-1];
                v.pop_back();
                x = v[v.size()-1];
                v.pop_back();
            }

            if(x == gx && y == gy){
                break;
            }
        }
        deep = findThreshold(frontier, gx, gy);
    }
    for(int i=0; i<v.size(); i+=3){
        ans.push_back(v[i]);
        ans.push_back(v[i+1]);
    }
    ans.push_back(gx);
    ans.push_back(gy);
    printAnswerB(ans);
}

int main(int argc, const char * argv[]) {
    int n;
```

```
    //n=1, BFS, n=2, DFS, n=3, IDS, n=4, A*, n=5, IDA*, others,
break
    while(cin>>n){
        if(n == 0){
            break;
        }
        //all nodes are initialized to 0
        for(int i=0; i<8; i++){
            for(int j=0; j<8; j++){
                arr[i][j] = 0;
            }
        }
        int startingx, startingy, goalx, goaly;
        cin>>startingx>>startingy>>goalx>>goaly;
        if(n == 1){
            BFS(startingx, startingy, goalx, goaly);
        }
        else if(n == 2){
            DFS(startingx, startingy, goalx, goaly);
        }
        else if(n == 3){
            IDS(startingx, startingy, goalx, goaly);
        }
        else if(n == 4){
            Astar(startingx, startingy, goalx, goaly);
        }
        else if(n == 5){
            IDAstar(startingx, startingy, goalx, goaly);
        }
        else{
            break;
        }
    }
    return 0;
}
```