## **广州大学学生实验报告**

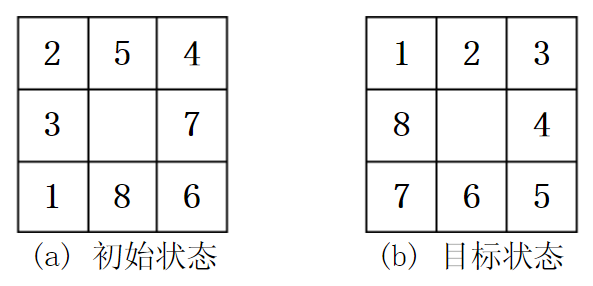
**开课学院及实验室：**计算机科学与工程实验室 **2021年10月2日**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **学院** | **计算机科学与网络工程学院** | **年级/专业/班** | **网络193** | **姓名** | 吴伟俊 | **学号** | 1906200107 |
| **实验课程名称** | **人工智能导论实验** | | | | | **成绩** |  |
| **实验项目名称** | **基于图搜索技术的八数码问题求解** | | | | | **指导老师** | 张少宏 |

(\*\*\*报告只能为文字和图片,老师评语将添加到此处,学生请勿作答\*\*\*)

**一、实验内容**

**问题描述**：八数码，在3×3的方格棋盘上，摆放着1到8这八个数码，有1个方格是空的，其初始状态如图1所示，要求对空格执行空格左移、空格右移、空格上移和空格下移这四个操作使得棋盘从初始状态到目标状态。



**内容提要**：分别用广度优先搜索策略、深度优先搜索策略和启发式搜索算法（至少两种）求解八数码问题；分析估价函数对启发式搜索算法的影响；探究讨论各个搜索算法的特点。

**二、实验设备**

1. 实验设备：计算机；

2. 平台：Windows操作系统，Visual C++ 6.0

**三、实验步骤**

* 随机生成一个八数码问题分布，设计一个可解的目标状态（要求棋盘9个位置都不同）。

const int N = 3;

int st, ed;

void input(int &x) {

    x = 0;

    vector<int> rd;

    for (int i = 0; i < N \* N; i++) rd.push\_back(i);

    shuffle(rd.begin(), rd.end(), default\_random\_engine(rand()));

    for (int i = 0, j = 0; i < N \* N; i++) {

        /\*

        int v; cin >> v;    // 手动输入

        x = x \* 10 + v;

        \*/

        int v = rd[i];   //随机生成

        x = x \* 10 + v;

        cout << v << " ";

        if (++j == 3) {

            j = 0;

            cout << endl;

        }

    }

}

* 分别用广度优先搜索策略、深度优先搜索策略和至少两种启发式搜索算法求解八数码问题。
* 分析估价函数对启发式搜索算法的影响。

估价函数就是评价函数，它用来评价子节点的好坏，因为准确评价是不可能的，所以成为估值。这就是所谓的有信息搜索。如果估价函数只考虑节点的某种性能上的价值，而不考虑深度，比较有名的就是有序搜索，它看重看好能否找出解，而不看解离起始点的距离。估价函数的形式为：f(n)=g(n)+h(n)，g(n)为起点到当前位置的实际路径长度，h(n)为所在位置到终点的最佳路径的估计距离。这个估算，保证了所找到的路径是最短路径

估价函数中，某位置到终点的最佳路径的估计距离h(x)是估价函数的关键。当该距离等于0时，意味着此时是盲目搜索，该算法则相当于是一个BFS；对于Astar算法来说，该值总小于等h’(x)，h’(x)为一个最短路径的值，因此该算法找到的解总是最优的。

* 探究讨论各个搜索算法的特点。

BFS：这是一种基于队列这种数据结构的搜索方式，它的特点是由每一个状态可以扩展出许多状态，然后再以此扩展，直到找到目标状态或者队列中头尾指针相遇，即队列中所有状态都已处理完毕。

DFS：基于递归的搜索方式，它的特点是由一个状态拓展一个状态，然后不停拓展，直到找到目标或者无法继续拓展结束一个状态的递归。

启发式搜索:它是利用问题拥有的启发信息来引导搜索，达到减少搜索范围、降低问题复杂度的目的. 在A\*算法中，估价函数为f(n)=g\*(n)+h\*(n)。这里面的h\*(n)的附加条件为h\*(n)<=h‘(n)，h’(n)为n到目标的直线最短距离，也就说A\*算法中挑选的启发函数是最优的。A算法效率较高，当没有要求最短路径只求通路时，A算法较好；A\*算法有时会走在一个较长的通路，但在抵达终点之前，肯定会舍弃这条路，走更短的路，而A算法则不一定。

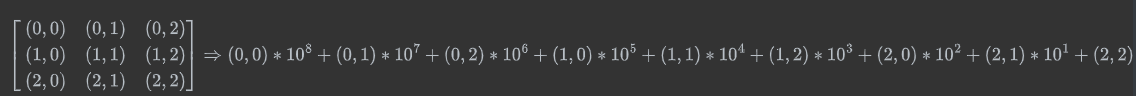
* \*扩展选做题：从初始状态到目标状态的变换，符合什么规律才可解（提示参考：逆序数）

**四、分析说明（包括核心代码及解释）**

1.DFS，BFS，Astar算法：

这里记录棋盘使用了一种hash方法，将3\*3的棋盘映射到一维上的[0,3\*3-1]区间,并用int存储

具体方法是：



DFS:

搜索每个空位的相邻4个格子

对每种搜索过的棋盘进行记忆化vis

int step = 1e3, cur\_step = 0;   //限定1e3步,超过这个步数判定无解

bool dfs(int u) {

    if (++cur\_step >= step) return 0;

    if (vis[u]) return 0; //记忆化

    vis[u] = 1;

    if (u == ed) {

        return 1;

    }

    int p = 0;

    for (int i = u; i % 10; i /= 10) p++;

    int x = p % N, y = p / N;

    for (int i = 0; i < 4; i++) {

        int cx = x + dx[i], cy = y + dy[i];

        if (check\_edge(cx) || check\_edge(cy)) continue;

        int cp = cy \* N + cx;

        int cu = swap(u, p, cp);

        if (vis[cu]) continue;

        fa[cu] = u;

        bool f = dfs(cu);

        if (f) return 1;

    }

    return 0;

}

void dfs() {

    vis.clear();

    fa.clear();

    cur\_step = 0;

    if (dfs(st)) {

        cout << "find the solution" << endl;

        print\_way();

    } else {

        cout << "no solution" << endl;

    }

}

BFS:

void bfs() {

    vis.clear();

    fa.clear();

    queue<int> q;

    q.push(st);

    while (q.size()) {

        int u = q.front(); q.pop();

        if (vis[u]) continue;

        vis[u] = 1;

        if (u == ed) {

            cout << "find the solution" << endl;

            print\_way();

            return;

        }

        int p = 0;

        for (int i = u; i % 10; i /= 10) p++;

        int x = p % N, y = p / N;

        for (int i = 0; i < 4; i++) {

            int cx = x + dx[i], cy = y + dy[i];

            if (check\_edge(cx) || check\_edge(cy)) continue;

            int cp = cy \* N + cx;

            int cu = swap(u, p, cp);

            if (vis[cu]) continue;

            fa[cu] = u;

            q.push(cu);

        }

    }

    cout << "no solution" << endl;

}

ASTAR:

预处理估值函数：每个点到终点的横纵距离之和

这里估值函数越低与结果越相近

用优先队列，每次拿出估值最低的局面进行搜索

void Astar() {

    vis.clear();

    fa.clear();

    priority\_queue<tuple<int, int, int>, vector<tuple<int, int, int>>, greater<tuple<int,int,int>>> q;

    map<int, int> mp;

    for (int i = 0, x = ed; i < N \* N; x /= 10, i++) {

        mp[x % 10] = i;

    }

    int \_h = 0;

    for (int i = 0, x = st; i < N \* N; x /= 10, i++) {

        \_h += abs(mp[x % 10] % N - i % N) + abs(mp[x % 10] / N - i / N);

    }

    q.push({\_h, \_h, st});

    while (q.size()) {

        auto [tot, h, u] = q.top(); q.pop();

        if (vis[u]) continue;

        vis[u] = 1;

        if (u == ed) {

            cout << "find the solution" << endl;

            print\_way();

            return;

        }

        int g = tot - h;

        int p = 0;

        for (int i = u; i % 10; i /= 10) p++;

        int x = p % N, y = p / N;

        for (int i = 0; i < 4; i++) {

            int cx = x + dx[i], cy = y + dy[i];

            if (check\_edge(cx) || check\_edge(cy)) continue;

            int cp = cy \* N + cx;

            int cu = swap(u, p, cp);

            int hu = u / fac10[cp] % 10, hv = u / fac10[p] % 10;

            int ch = abs(mp[hu] % N - x) + abs(mp[hu] / N - y) + abs(mp[hv] % N - cx) + abs(mp[hv] - cy);

            ch -= abs(mp[hu] % N - cx) + abs(mp[hu] / N - cy) + abs(mp[hv] % N - x) + abs(mp[hv] - y);

            if (vis[cu]) continue;

            fa[cu] = u;

            q.push({g + 1 + h + ch, h + ch, cu});

        }

    }

    cout << "no solution" << endl;

}

**五、总结心得**

**实验结果：**

**初始棋盘和目标棋盘**

手机屏幕的截图

描述已自动生成

**DFS:**



**超过1000步，所以判定无解，虽然可能存在解但是会爆栈所以不搜了**

**BFS:**

电子计算器

中度可信度描述已自动生成

**Astar:**

电子计算机

中度可信度描述已自动生成

实验总结：

本次实验，我复习了BFS和DFS以及之前其实就学过的ASTAR。但我了解到了启发式思想在搜索中的作用，并知道BFS和DFS其实是启发式搜索里的特例。另外我还体会到了估值函数在启发式搜索中的作用，不同的估值函数对启发式搜索的收敛速度不一样，较好的估值函数则很快就能找到解，较差的相对来说时间复杂度上就高不少。

实验源代码：

#include <bits/stdc++.h>

#define int long long

#define endl '\n'

using namespace std;

const int N = 3;

int st, ed;

void input(int &x) {

    x = 0;

    vector<int> rd;

    for (int i = 0; i < N \* N; i++) rd.push\_back(i);

    shuffle(rd.begin(), rd.end(), default\_random\_engine(rand()));

    for (int i = 0, j = 0; i < N \* N; i++) {

        /\*

        int v; cin >> v;    // 手动输入

        x = x \* 10 + v;

        \*/

        int v = rd[i];   //随机生成

        x = x \* 10 + v;

        cout << v << " ";

        if (++j == 3) {

            j = 0;

            cout << endl;

        }

    }

}

/\*

1 2 3

4 5 6

7 8 0

1 2 3

4 5 6

7 0 8

\*/

/\*

0 2 3

4 5 6

7 8 9

2 3 6

4 5 9

7 8 0

\*/

int fac10[20];

int dx[] = {1, -1, 0, 0};

int dy[] = {0, 0, 1, -1};

map<int, bool> vis;

map<int, int> fa;

bool check\_edge(int x) {

    return x < 0 || x >= N;

}

int swap(int u, int p, int cp) {

    int v = u;

    int w = u / fac10[cp] % 10;

    v -= w \* fac10[cp];

    v += w \* fac10[p];

    return v;

}

void print\_way() {

    vector<int> trace;

    for (int x = ed; x != st; x = fa[x]) {

        trace.push\_back(x);

    }

    trace.push\_back(st);

    for (int i = trace.size() - 1; i >= 0; i--) {

        string s = "";

        for (int j = 0, x = trace[i]; j < N \* N; j++, x /= 10) {

            s += x % 10 + '0';

        }

        reverse(s.begin(), s.end());

        cout << "->" << endl;

        for (int j = 0; j < N; j++) {

            for (int k = 0; k < N; k++) {

                cout << s[j \* N + k] << " ";

            }

            cout << endl;

        }

    }

}

void bfs() {

    vis.clear();

    fa.clear();

    queue<int> q;

    q.push(st);

    while (q.size()) {

        int u = q.front(); q.pop();

        if (vis[u]) continue;

        vis[u] = 1;

        if (u == ed) {

            cout << "find the solution" << endl;

            print\_way();

            return;

        }

        int p = 0;

        for (int i = u; i % 10; i /= 10) p++;

        int x = p % N, y = p / N;

        for (int i = 0; i < 4; i++) {

            int cx = x + dx[i], cy = y + dy[i];

            if (check\_edge(cx) || check\_edge(cy)) continue;

            int cp = cy \* N + cx;

            int cu = swap(u, p, cp);

            if (vis[cu]) continue;

            fa[cu] = u;

            q.push(cu);

        }

    }

    cout << "no solution" << endl;

}

int step = 1e3, cur\_step = 0;   //限定1e3步,超过这个步数判定无解

bool dfs(int u) {

    if (++cur\_step >= step) return 0;

    if (vis[u]) return 0;   //记忆化处理

    vis[u] = 1;

    if (u == ed) {

        return 1;

    }

    int p = 0;

    for (int i = u; i % 10; i /= 10) p++;

    int x = p % N, y = p / N;

    for (int i = 0; i < 4; i++) {

        int cx = x + dx[i], cy = y + dy[i];

        if (check\_edge(cx) || check\_edge(cy)) continue;

        int cp = cy \* N + cx;

        int cu = swap(u, p, cp);

        if (vis[cu]) continue;

        fa[cu] = u;

        bool f = dfs(cu);

        if (f) return 1;

    }

    return 0;

}

void dfs() {

    vis.clear();

    fa.clear();

    cur\_step = 0;

    if (dfs(st)) {

        cout << "find the solution" << endl;

        print\_way();

    } else {

        cout << "no solution" << endl;

    }

}

void Astar() {

    vis.clear();

    fa.clear();

    priority\_queue<tuple<int, int, int>, vector<tuple<int, int, int>>, greater<tuple<int,int,int>>> q;

    map<int, int> mp;

    for (int i = 0, x = ed; i < N \* N; x /= 10, i++) {

        mp[x % 10] = i;

    }

    int \_h = 0;

    for (int i = 0, x = st; i < N \* N; x /= 10, i++) {  //预处理每个点的估值函数值,即：每个点到它目标点坐标的x,y差值

        \_h += abs(mp[x % 10] % N - i % N) + abs(mp[x % 10] / N - i / N);

    }

    q.push({\_h, \_h, st});

    while (q.size()) {

        auto [tot, h, u] = q.top(); q.pop();

        if (vis[u]) continue;

        vis[u] = 1;

        if (u == ed) {

            cout << "find the solution" << endl;

            print\_way();

            return;

        }

        int g = tot - h;

        int p = 0;

        for (int i = u; i % 10; i /= 10) p++;

        int x = p % N, y = p / N;

        for (int i = 0; i < 4; i++) {

            int cx = x + dx[i], cy = y + dy[i];

            if (check\_edge(cx) || check\_edge(cy)) continue;

            int cp = cy \* N + cx;

            int cu = swap(u, p, cp);

            int hu = u / fac10[cp] % 10, hv = u / fac10[p] % 10;

            int ch = abs(mp[hu] % N - x) + abs(mp[hu] / N - y) + abs(mp[hv] % N - cx) + abs(mp[hv] - cy);

            ch -= abs(mp[hu] % N - cx) + abs(mp[hu] / N - cy) + abs(mp[hv] % N - x) + abs(mp[hv] - y);

            if (vis[cu]) continue;

            fa[cu] = u;

            q.push({g + 1 + h + ch, h + ch, cu});

        }

    }

    cout << "no solution" << endl;

}

signed main() {

    srand((unsigned)time(NULL));

    fac10[0] = 1;

    for (int i = 1; i < 19; i++) {

        fac10[i] = fac10[i - 1] \* 10;

    }

    printf("init\n");

    input(st);

    int tmp\_st = st;

    printf("target\n");

    input(ed);

    printf("BFS:\n");

    bfs();

    printf("DFS:\n");

    st = tmp\_st;

    dfs();

    printf("ASTAR:\n");

    st = tmp\_st;

    Astar();

}