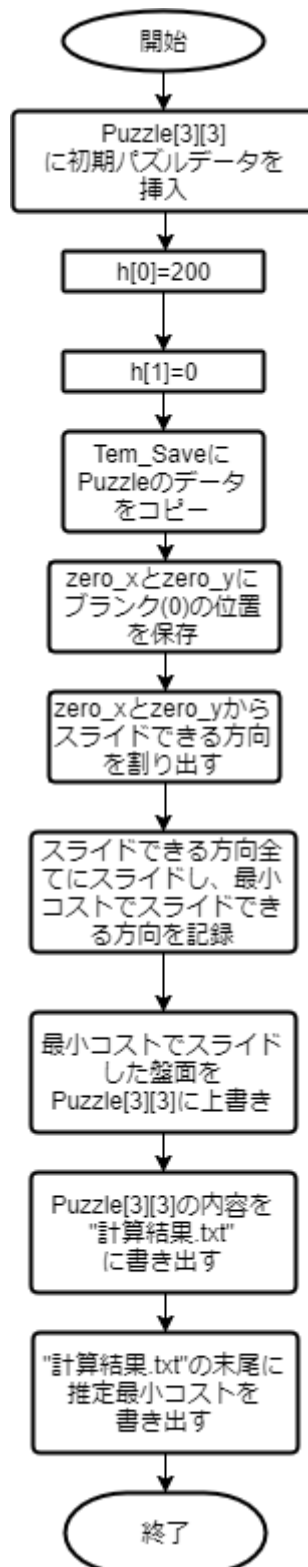


1. アルゴリズム説明



2. ソースコード

```
// A_starALGORITHM.cpp : コンソール アプリケーションのエントリ ポイントを定義します。
//

#include "stdafx.h"
#include <stdio.h>
#include <windows.h>
#include <iostream>
#include <fstream>
#include <cstdlib>

//マンハッタン距離計算関数
int Manhattan_Distance(int Puzzle[][3]) {

    int Goal[3][3] { { 1, 2, 3 }, //パズルのゴール状態
                     { 4, 5, 6 },
                     { 7, 8, 0 } };

    int x_dist = 0;
    int y_dist = 0;

    int Tortal_M_dist = 0; //合計マンハッタン距離計算用

    while
(Puzzle != Goal) { //パズルの現在状態とゴール状態が完全一致するまでループ

    //マンハッタン距離
    の計算開始
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
```

```

        int a = Puzzle[i][j];           //横[j+1]番目、 縦[i+1]番目
の現在パズルの数値を[a]に代入
        for (int k = 0; k < 3; k++) {
            for (int l = 0; l < 3; l++) {
                if (a == Goal[k][l] && a != 0) {    //検索
中、現状とゴールの値が一致、かつ空白では無い場所を発見した場合
                    x_dist = j - l;    //横のズレを計算
                    y_dist = i - k;    //縦のズレを計算

                    if (x_dist < 0)      //jl_dist
が0未満だったら
                        x_dist *= (-1);  //jl_dist
に-1を掛ける(符号を+に)

                    if (y_dist < 0)      //ik_dist
が0未満だったら
                        y_dist *= (-1);  //ik_dist
に-1を掛ける(符号を+に)

                    Tortal_M_dist += x_dist + y_dist;
//[横のズレ]+[縦のズレ] = [a]のマンハッタン距離

                }
            }
        }

    }

//マンハッタン距離の計算完了

return Tortal_M_dist;    //合計マンハッタン距離を返す
}

```

```

int _tmain(int argc, _TCHAR* argv[])
{
    int Puzzle[3][3] { { 8, 1, 5 },           //パズル現在状態(0は空き空
間とする)
    { 2, 0, 4 },
    { 6, 3, 7 }
    };

    int Goal[][3] { { 1, 2, 3 },              //パズルのゴール状態
    { 4, 5, 6 },
    { 7, 8, 0 }
    };

    int Tem_Save[3][3];                      //パズル
一時保存用

    int Safe_ID;                             //推定最
小コストのパズルID一次保存用

    int f;
    //最少コスト(g+h)
    int g = 0;
    //現在状態までのコスト
    int h[2];
    //現在状態からゴールまでの推定最少コスト

    int zero_x = 0;
    //blank (0)のx座標保存用
    int zero_y = 0;
    //blank (0)のy座標保存用

    std::ofstream ofs("計算結果.txt", std::ios_base::out);
    if (!ofs) {
        std::cerr << "error" << std::endl;
    }
}

```

```

std::exit(EXIT_FAILURE);
}

h[0] = 200;           //推定最小コストを初期化
h[1] = 0;             //計算コストを初期化
Safe_ID = 0;

for (int y = 0; y < 3; y++) {
    for (int x = 0; x < 3; x++) {

        Tem_Save[y][x] = Puzzle[y][x];    //一次保存用に、現
在の状態をコピー

        if (Puzzle[y][x] == 0) {          //ブランク(0)を検
出した場合

            zero_x = x;
            zero_y = y;

            //ブランクの座標を保存しておく
        }

    }

    //コピーとブランクの位置割り出し終了

    //////////////////////////////////////
    // 最小コストの割り出し //
    //////////////////////////////////////

    if (zero_x <= 1) {                    //0を右にスライドできる場合
        Tem_Save[zero_y][zero_x] = Puzzle[zero_y][zero_x + 1];
        Tem_Save[zero_y][zero_x + 1] = 0;
        h[1] = Manhattan_Distance(Tem_Save);

        printf("左スライド = %d¥n", h[1]);

        if (h[0] > h[1]) {                //推定最小コストよりコストが低かった
場合

```

```

        h[0] = h[1];
        Safe_ID = 1;
        for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 3; j++) {
                Tem_Save[i][j] = Puzzle[i][j];
            }
        }

        //一次保存用に、現在の状態をコピー

    }

    if (zero_x >= 1) {        //0を左にスライドできる場合
        Tem_Save[zero_y][zero_x] = Puzzle[zero_y][zero_x - 1];
        Tem_Save[zero_y][zero_x - 1] = 0;
        h[1] = Manhattan_Distance(Tem_Save);

        printf("右スライド = %d¥n", h[1]);
        if (h[0] > h[1]) {        //推定最小コストよりコストが低かった
            h[0] = h[1];
            Safe_ID = 2;
        }
        for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 3; j++) {
                Tem_Save[i][j] = Puzzle[i][j];        //一次保
            }
        }

        //一次保存用に、現在の状態をコピー

    }

    if (zero_y >= 1) {        //0を上にもスライドできる場合
        Tem_Save[zero_y][zero_x] = Puzzle[zero_y - 1][zero_x];
        Tem_Save[zero_y - 1][zero_x] = 0;
        h[1] = Manhattan_Distance(Tem_Save);

```

```
printf("下スライド = %d¥n", h[1]);
```

場合

```
if (h[0] > h[1]) { //推定最小コストよりコストが低かった
```

```
h[0] = h[1];
```

```
Safe_ID = 3;
```

```
}
```

```
for (int i = 0; i < 3; i++) {
```

```
for (int j = 0; j < 3; j++) {
```

```
Tem_Save[i][j] = Puzzle[i][j]; //一次保
```

存用に、現在の状態をコピー

```
}
```

```
}
```

```
}
```

```
if (zero_y <= 1) { //0を下にスライドできる場合
```

```
Tem_Save[zero_y][zero_x] = Puzzle[zero_y + 1][zero_x];
```

```
Tem_Save[zero_y + 1][zero_x] = 0;
```

```
h[1] = Manhattan_Distance(Tem_Save);
```

```
printf("上スライド = %d¥n", h[1]);
```

場合

```
if (h[0] > h[1]) { //推定最小コストよりコストが低かった
```

```
h[0] = h[1];
```

```
Safe_ID = 4;
```

```
}
```

```
for (int i = 0; i < 3; i++) {
```

```
for (int j = 0; j < 3; j++) {
```

```
Tem_Save[i][j] = Puzzle[i][j]; //一次保
```

存用に、現在の状態をコピー

```
}
```

```
}
```

```
}
```

```
//割り出し完了
```

//IDを基に、パズルをスライドさせる

```
if (Safe_ID == 1) {  
    Puzzle[zero_y][zero_x] = Puzzle[zero_y][zero_x + 1];  
    Puzzle[zero_y][zero_x + 1] = 0;  
    printf("スライド方向:左\n");  
}
```

```
if (Safe_ID == 2) {  
    Puzzle[zero_y][zero_x] = Puzzle[zero_y][zero_x - 1];  
    Puzzle[zero_y][zero_x - 1] = 0;  
    printf("スライド方向:右\n");  
}
```

```
if (Safe_ID == 3) {  
    Puzzle[zero_y][zero_x] = Puzzle[zero_y - 1][zero_x];  
    Puzzle[zero_y - 1][zero_x] = 0;  
    printf("スライド方向:下\n");  
}
```

```
if (Safe_ID == 4) {  
    Puzzle[zero_y][zero_x] = Puzzle[zero_y + 1][zero_x];  
    Puzzle[zero_y + 1][zero_x] = 0;  
    printf("スライド方向:上\n");  
}
```

//現在の推定最小コストを計算

```
f = g + h[0];
```

```
printf("現在の推定最小コスト = %d\n", f);
```

```
for (int y = 0; y < 3; y++) {  
    for (int x = 0; x < 3; x++) {  
        ofs<<(Puzzle[y][x]) <<" ";
```



```

        if (Puzzle[y][x] != 0)
            printf("%d ", Puzzle[y][x]);
        else
            printf(" ");
    }
    printf("\n");
    ofs << "\n";
}
printf("\n");

getchar();

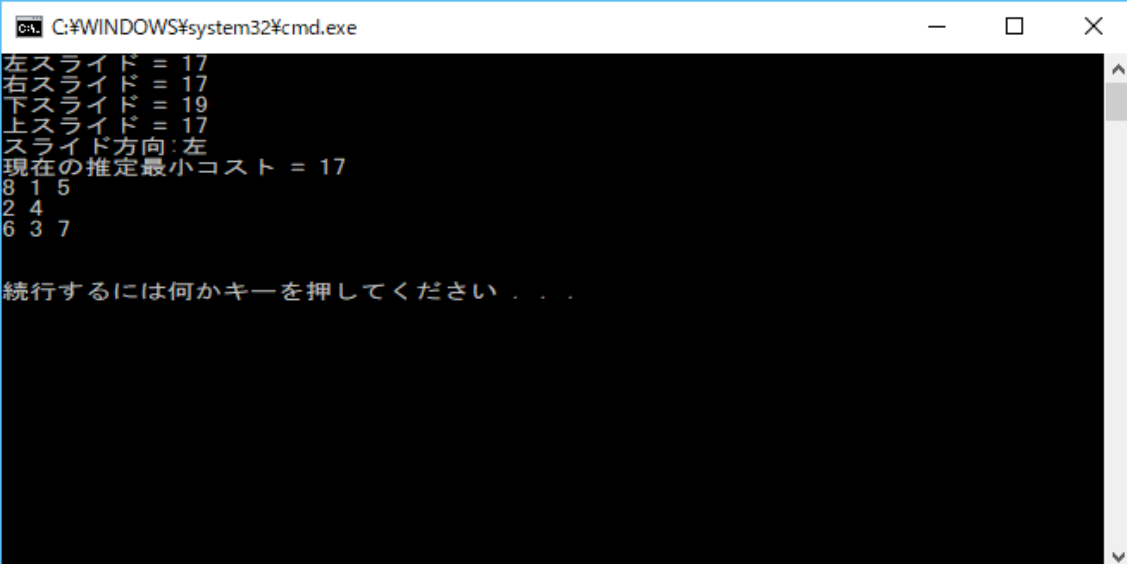
g++;

ofs << "\n";
ofs << "推定コスト:" << f << std::endl;

return 0;
}

```

3. 出力結果



```

C:\WINDOWS\system32\cmd.exe
左スライド = 17
右スライド = 17
下スライド = 19
上スライド = 17
スライド方向: 左
現在の推定最小コスト = 17
8 1 5
2 4
6 3 7

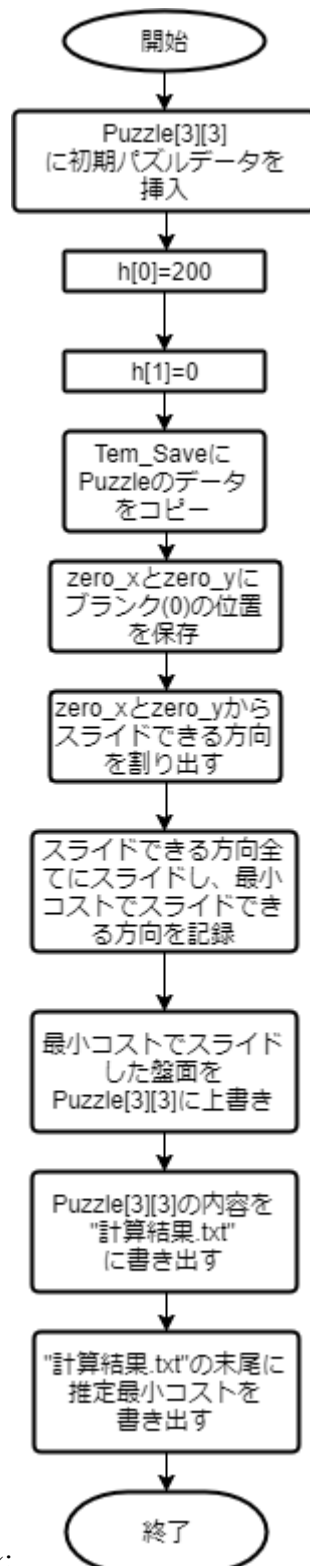
続行するには何かキーを押してください . . .

```

実行環境

Windows10

Microsoft Visual Studio2017



バージョン:

15.0.26730.16