人工知能Ⅱ　課題

5CS35 番 鈴木大貴

1. アルゴリズムの説明

作成したプログラムのフローチャートを下図に示す。

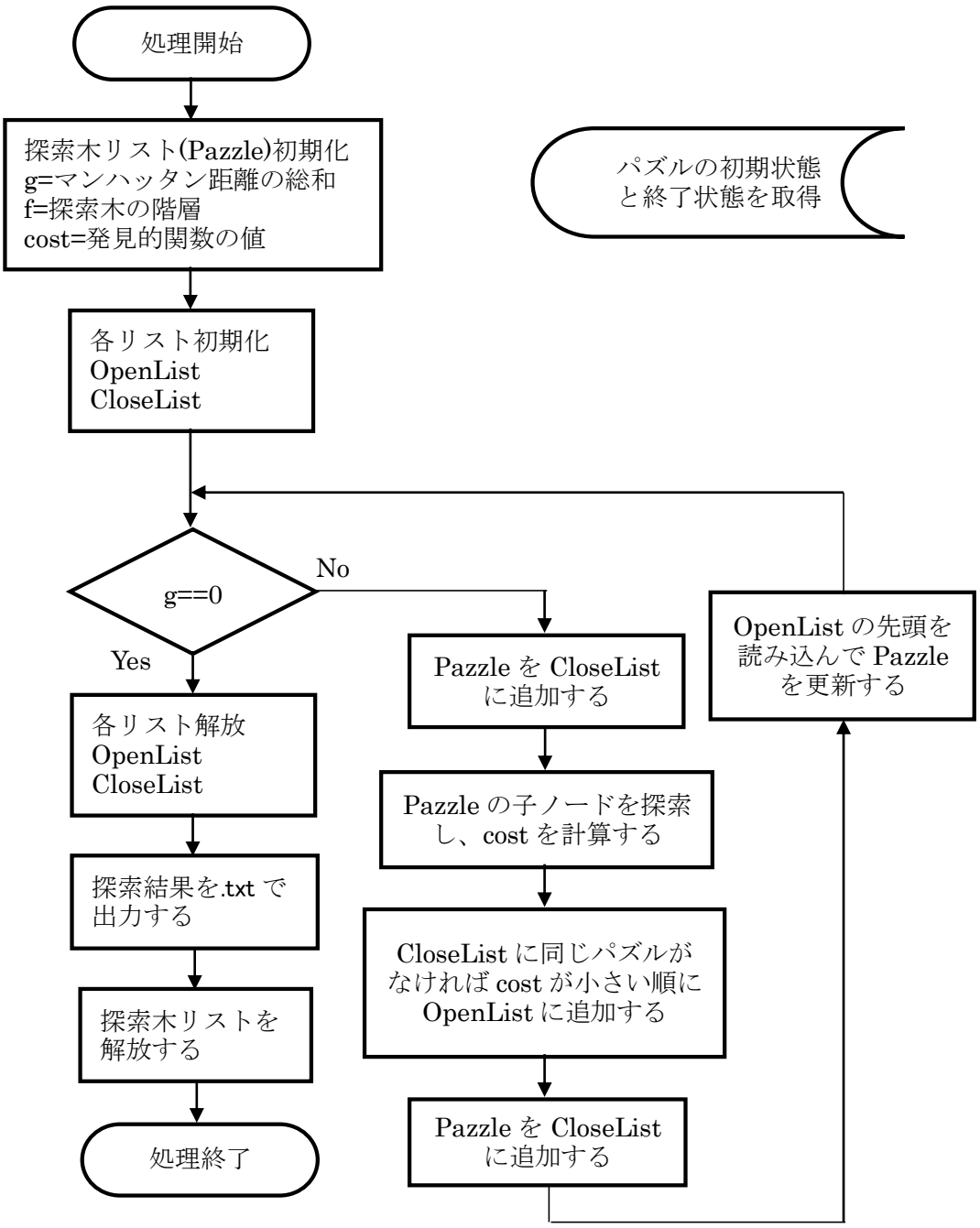

図 1. 作成したプログラムのフローチャート

2. 作成したソースコード

図 2 に本プログラムのソースコードを示す。

```
#include <stdio.h>
#include <stdlib.h>
#include <malloc.h>
#include <math.h>

int input[3][3]={{8,1,5},{2,0,4},{6,3,7}};
int answer[3][3]={{1,2,3},{4,5,6},{7,8,0}};

//パズルデータ
typedef struct _Pazzle{
        int data[3][3];          //パズルデータ
        int g;                   //マンハッタン距離の総和
        int f;                   //探索木の階層
        int cost;                //発見的関数の値
        unsigned long int ID;    //パズルデータをunsigned long int型にまとめたもの

        //探索木の子のポインタ
        struct _Pazzle *t_next[4];
        //探索木の親のポインタ
        struct _Pazzle *t_prev;

        //"探索終了後"に見つかった解を辿るためのポインタ
        //探索中はNULL
        struct _Pazzle *ans;

}Pazzle;

//Openリスト
typedef struct _OpenList {

        //対象のパズルデータ
        struct _Pazzle *pzl;

        //双方向リストのためのポインタ
        struct _OpenList *next;
        struct _OpenList *prev;
}OpenList;

//Closeリスト
typedef struct _ClozeList{

        //パズルデータをunsigned long int型にまとめたもの
        unsigned long int ID;

        //単方向リストのためのポインタ
        struct _ClozeList *next;
}CloseList;
```

```
int MainSearch(Pazzle *pazzle);
int O_ListAdd(OpenList *openL, Pazzle *pazzle);
void O_ListClear(OpenList *openL);
int C_ListAdd(CloseList *closeL, unsigned long int ID);
int C_ListCheck(CloseList *closeL, unsigned long int ID);
void C_ListClear(CloseList *closeL);
int Manhattan(int data[3][3]);
unsigned long int SetID(int data[3][3]);
void ShowPazzle(int data[3][3]);
int FileOutput(Pazzle *pazzle);


int main(){
	Pazzle *pazzle;
	int i,n;

	//メモリ確保
	pazzle = (Pazzle*)malloc(sizeof(Pazzle));
	if(pazzle == NULL) return -1;

	//初期設定
	//パズルの初期データを読み込み代入
	pazzle->ID = 0;
	for (i = 0; i < 3; i++) for (n = 0; n < 3; n++) {
		pazzle->data[i][n] = input[i][n];
		pazzle->ID += pow(10, (8 - (i*3 + n)))*input[i][n];
	}
	//マンハッタン距離を計算し代入
	pazzle->g = Manhattan(pazzle->data);
	//経路コスト(階層数)の初期値を代入
	pazzle->f = 0;
	//発見的関数を計算し代入
	pazzle->cost = pazzle->f + pazzle->g;
	//リストの初期設定
	for(i = 0; i < 4; i++) pazzle->t_next[i] = NULL;
	pazzle->t_prev = NULL;

	//探索開始
	MainSearch(pazzle);

	//ファイル出力
	FileOutput(pazzle);

	getchar();

	return 0;
}
```

```
//発見的関数を求める
int MainSearch(Pazzle *pazzle){

	//オープンリスト
	OpenList *openL, *openL_now;

	//クローズリストとその先頭
	CloseList *closeL, *closeL_start;
	int i,n,m;
	int count = 0;

	//オープンリストメモリ確保
	openL = (OpenList*)malloc(sizeof(OpenList));
	if (openL == NULL) return -1;
	//オープンリスト初期化
	openL->prev = NULL;
	openL->next = NULL;
	openL->pzl = pazzle;

	//クローズリストメモリ確保
	closeL = (CloseList*)malloc(sizeof(CloseList));
	if (closeL == NULL) return -1;
	//クローズリスト初期化
	closeL->ID = 0;
	closeL->next = NULL;

	//クローズリストの初期地点を記憶
	closeL_start = closeL;

	//探索が終了するまでループを繰り返す
	while (1){

		//対象の子ノード（pazzle->t_next[]）の初期化
		for (i = 0; i < 4; i++) {
			//メモリ確保
			pazzle->t_next[i] = (Pazzle*)malloc(sizeof(Pazzle));
			if (pazzle->t_next[i] == NULL) return -1;
			//階層の更新
			pazzle->t_next[i]->f = pazzle->f + 1;
			//親のデータをコピー
			for (n = 0; n < 3; n++) for (m = 0; m < 3; m++)
				pazzle->t_next[i]->data[n][m] = pazzle-
>data[n][m];
			//コストの初期化
			pazzle->t_next[i]->cost = pazzle->cost;
			//マンハッタン距離の初期化
			pazzle->t_next[i]->g = pazzle->g;
			//ポインタの初期化
			for (n = 0; n < 4; n++) pazzle->t_next[i]->t_next[n] =
NULL;
			pazzle->t_next[i]->t_prev = pazzle;
		}
```

```
                //基準のパズルデータを記憶
                openL_now = openL;
                C_ListAdd(closeL, pazzle->ID);

                //空欄の入れ替えを行う
                for (n = 0; n < 3; n++) for (m = 0; m < 3; m++){
                        //空欄の移動先が範囲外だった場合、コストを負数にする(フラグ
の代用)
                        if (pazzle->data[n][m] == 0){
                                //上
                                if (n != 0){
                                        pazzle->t_next[0]->data[n][m] = pazzle-
>t_next[0]->data[n - 1][m];
                                        pazzle->t_next[0]->data[n - 1][m] = 0;
                                        pazzle->t_next[0]->ID = SetID(pazzle-
>t_next[0]->data);
                                        pazzle->t_next[0]->t_prev = pazzle;
                                }
                                else pazzle->t_next[0]->cost = -1;
                                //左
                                if (m != 0){
                                        pazzle->t_next[1]->data[n][m] = pazzle-
>t_next[1]->data[n][m - 1];
                                        pazzle->t_next[1]->data[n][m - 1] = 0;
                                        pazzle->t_next[1]->ID = SetID(pazzle-
>t_next[1]->data);
                                        pazzle->t_next[1]->t_prev = pazzle;
                                }
                                else pazzle->t_next[1]->cost = -1;
                                //下
                                if (n != 2){
                                        pazzle->t_next[2]->data[n][m] = pazzle-
>t_next[2]->data[n + 1][m];
                                        pazzle->t_next[2]->data[n + 1][m] = 0;
                                        pazzle->t_next[2]->ID = SetID(pazzle-
>t_next[2]->data);
                                        pazzle->t_next[2]->t_prev = pazzle;
                                }
                                else pazzle->t_next[2]->cost = -1;
                                //右
                                if (m != 2){
                                        pazzle->t_next[3]->data[n][m] = pazzle-
>t_next[3]->data[n][m + 1];
                                        pazzle->t_next[3]->data[n][m + 1] = 0;
                                        pazzle->t_next[3]->ID = SetID(pazzle-
>t_next[3]->data);
                                        pazzle->t_next[3]->t_prev = pazzle;
                                }
                                else pazzle->t_next[3]->cost = -1;
                        }
                }
```

```
                    //空欄が移動できた場合、マンハッタン距離と発見的関数を計算する
                    for (i = 0; i < 4; i++){
                            //ここで空欄が移動できたか判断、更にクローズリスト内に対象
のIDが存在しないことを確認
                            if (pazzle->t_next[i]->cost != -1 &&
C_ListCheck(closeL_start, pazzle->t_next[i]->ID) == 0) {
                                    //マンハッタン距離の総和を計算し、コストを算出、そ
の後オープンリストに加える
                                    pazzle->t_next[i]->g = Manhattan(pazzle-
>t_next[i]->data);
                                    pazzle->t_next[i]->cost = pazzle->t_next[i]->f +
pazzle->t_next[i]->g;
                                    O_ListAdd(openL, pazzle->t_next[i]);
                            }
                    }

                    //現在のOpenListを記録する
                    openL = openL_now;

                    //ポインタの位置が末端でない場合リストをつなぎなおす
                    if (openL->prev != NULL) {
                            openL->prev->next = openL->next;
                    }
                    if (openL->next != NULL) {
                            openL->next->prev = openL->prev;
                    }

                    //次のポインタに移動
                    openL = openL->next;

                    //オープンリストから消す
                    free(openL_now);

                    //リストを先頭に戻す
                    while (openL->prev != NULL) openL = openL->prev;
                    //オープンリストの中身表示
                    while (1) {
                            if (openL->next == NULL) break;
                            else openL = openL->next;
                    }
                    //リストを先頭に戻す
                    while (openL->prev != NULL) openL = openL->prev;
                    //マンハッタン距離の総和がが0になったときに探索を終了する
                    if (openL->pzl->g == 0){
                            printf("探索終了");
                            ShowPazzle(openL->pzl->data);
                            printf("コスト%d マンハッタン%d 階層%d¥n", openL->pzl-
>cost, openL->pzl->g, openL->pzl->f);
                            break;
                    }
```

```
//オープンリストをすべて探索したとき
                if (openL->next == NULL) {
                        printf("探索失敗 解にたどり着けません¥n");
                        return -1;
                }

                //オープンリストから次のパズルを読み込む
                pazzle = openL->pzl;

        }

        //探索終了後、親のノードをたどって手順を記録する
        if (openL->pzl->g != 0) {
                printf("探索が完了していません¥n");
                return -1;
        }
        else {
                pazzle = openL->pzl;
                pazzle->ans = NULL;
                while (pazzle->t_prev != NULL) {
                        pazzle->t_prev->ans = pazzle;
                        pazzle = pazzle->t_prev;
                }
        }

        //リストを先頭に戻す
        while (openL->prev != NULL) openL = openL->prev;
        //オープンリスト開放
        O_ListClear(openL);
        printf("オープンリストを開放しました¥n");
        //クローズリスト開放
        C_ListClear(closeL_start);
        printf("クローズリストを開放しました¥n");

        return 0;
}

//コストが小さい順にオープンリストに挿入する
int O_ListAdd(OpenList *openL, Pazzle *pazzle){

        OpenList *newInput;

        newInput = (OpenList*)malloc(sizeof(OpenList));

        if(openL==NULL || pazzle==NULL || newInput==NULL){
                printf("error リストが存在しないため追加できません¥n");
                return -1;
        }
```

```
	//値のコピー
	newInput->next = NULL;
	newInput->prev = NULL;
	newInput->pzl = pazzle;

	//リストを先頭に移動
	while (openL->prev != NULL) openL = openL->prev;

	//発見的関数の値、階層で小さい順になるようにリストに挿入
	while (1){
		if(newInput->pzl->cost < openL->pzl->cost){
			newInput->prev=openL->prev;
			newInput->next=openL;
			if (openL->prev != NULL) openL->prev->next=newInput;
			openL->prev=newInput;
			break;
		}
		else if(openL->next==NULL){
			newInput->next = openL->next;
			openL->next = newInput;
			newInput->prev = openL;
			break;
		}
		openL=openL->next;
	}

	return 0;

}

//オープンリストを削除する　引数はリストの先頭を渡す必要がある
void O_ListClear(OpenList *openL_start) {

	OpenList *p;

	p = openL_start;

	if (p->next != NULL) O_ListClear(p->next);
	if (p->prev != NULL) p->prev->next = NULL;
	free(p);

}

//クローズリストに追加する
int C_ListAdd(CloseList *closeL, unsigned long int ID) {
	CloseList *new_closeL;

	new_closeL = (CloseList*)malloc(sizeof(CloseList));

	if (closeL == NULL || new_closeL == NULL) return -1;

	new_closeL->next = NULL;
	new_closeL->ID = ID;
```

```
	//末端に移動
	while (closeL->next != NULL) closeL = closeL->next;
	closeL->next = new_closeL;

	return 0;
}

//クローズリストを削除する　引数はリストの先頭を渡す必要がある
void C_ListClear(CloseList *closeL_start) {

	CloseList *p;

	while (1) {
		p = closeL_start;
		if (closeL_start->next != NULL) {
			closeL_start = closeL_start->next;
			free(p);
		}
		else {
			free(p);
			break;
		}
	}
}

//クローズリストに引数のIDがあるか確認する(リストの先頭を引数にする必要あり)
//戻り値0・・・クローズリスト内に存在しない
//戻り値1・・・クローズリスト内に存在する
int C_ListCheck(CloseList *closeL, unsigned long int ID) {

	if (closeL == NULL) return -1;

	while (closeL->next != NULL) {
		if (closeL->ID == ID) return 1;
		//printf("%ld¥n", closeL->ID);
		closeL = closeL->next;
	}

	return 0;
}

//パズルからマンハッタン距離の総和を計算し戻り値として返す
int Manhattan(int data[3][3]){
	int sum=0;
	int i,n,m,l,j;
```

```
	//1～8の値を順番に探索
	for(i=1;i<9;i++){
		//入力データから探索
		for(n=0;n<3;n++) for(m=0;m<3;m++){
			if(data[n][m]==i){
				//正解データから探索
				for(l=0;l<3;l++) for(j=0;j<3;j++){
					if(answer[l][j]==i){
						//要素数のずれからマンハッタン距
離を計算
						sum += abs(n-l)+abs(m-j);
					}
				}
			}
		}
	}
	return sum;
}

//パズルデータからIDを発行し戻り値として返す
unsigned long int SetID(int data[3][3]) {
	unsigned long int ID = 0;
	int n, m;

	for (n = 0; n<3; n++) for (m = 0; m<3; m++) {
		ID += pow(10, (8 - (n*3 + m)))*data[n][m];
	}
	//printf("%ld¥n", ID);

	return ID;
}

//パズルの配置を表示する
void ShowPazzle(int data[3][3]){
	int n,m;
	printf("¥n");
	for(n=0;n<3;n++){
		for(m=0;m<3;m++){
			printf("%d ",data[n][m]);
		}
		printf("¥n");
	}
}

//ファイル出力
int FileOutput(Pazzle *pazzle) {
	int n, m;
	FILE *file;
	errno_t error;
```

```
	if ((error = fopen_s(&file, "8puzzle.txt", "w+")) != 0) {
		// エラー処理
		printf("error ファイルを開くことができません¥n");
		return -1;
	}

	//リストの先頭へ
	while (pazzle->t_prev != NULL) pazzle = pazzle->t_prev;

	//ポインタを辿ってファイルに書き込む
	while (1) {
		for (n = 0; n < 3; n++) {
			for (m = 0; m < 3; m++) {
				fprintf(file, "%d ", pazzle->data[n][m]);
			}
			fprintf(file, "¥n");
		}
		fprintf(file, "発見的関数の値:%d¥nマンハッタン距離の総和:%d 階
層:%d¥n¥n", pazzle->cost, pazzle->g, pazzle->f);

		if (pazzle->ans == NULL) break;
		else pazzle = pazzle->ans;
	}

	fclose(file);

	return 0;

}
```

図 2　作成したプログラムのソースコード

3. プログラムの実行結果

3.1 実行環境

以下に本プログラムの実行環境を示す。

- Windows10 Pro 64bit
- Microsoft Visual Studio 2017

3.2 パズルの状態と発見的関数の遷移

図 3 に初期状態から終了状態までのパズルの状態と発見的関数の値を示す。また、パズルの空欄は 0 を用いて表す。

8 1 5
2 0 4
6 3 7
発見的関数の値:18

8 1 5
2 3 4
6 0 7
発見的関数の値:18

8 1 5
2 3 4
0 6 7
発見的関数の値:18

8 1 5
0 3 4
2 6 7
発見的関数の値:20

0 1 5
8 3 4
2 6 7
発見的関数の値:20

1 0 5

8 3 4

2 6 7

発見的関数の値:20

1 3 5

8 0 4

2 6 7

発見的関数の値:20

1 3 5

8 4 0

2 6 7

発見的関数の値:20

1 3 5

8 4 7

2 6 0

発見的関数の値:22

1 3 5

8 4 7

2 0 6

発見的関数の値:22

1 3 5

8 4 7

0 2 6

発見的関数の値:22

1 3 5

0 4 7

8 2 6

発見的関数の値:22

1 3 5
4 0 7
8 2 6
発見的関数の値:22

1 3 5
4 7 0
8 2 6
発見的関数の値:22

1 3 0
4 7 5
8 2 6
発見的関数の値:22

1 0 3
4 7 5
8 2 6
発見的関数の値:22

0 1 3
4 7 5
8 2 6
発見的関数の値:24

4 1 3
0 7 5
8 2 6
発見的関数の値:26

4 1 3
7 0 5
8 2 6
発見的関数の値:26

4 1 3
7 2 5
8 0 6
発見的関数の値:26

4 1 3
7 2 5
0 8 6
発見的関数の値:26

4 1 3
0 2 5
7 8 6
発見的関数の値:26

0 1 3
4 2 5
7 8 6
発見的関数の値:26

1 0 3
4 2 5
7 8 6
発見的関数の値:26

1 2 3
4 0 5
7 8 6
発見的関数の値:26

1 2 3
4 5 0
7 8 6
発見的関数の値:26

1 2 3 4 5 6 7 8 0 発見的関数の値:26

図 3 初期状態から終了状態までのパズルの状態と発見的関数の値

3.3 パズルの発見的関数の値の推移

図 4 に探索毎に出力した発見的関数の値の推移を示す。

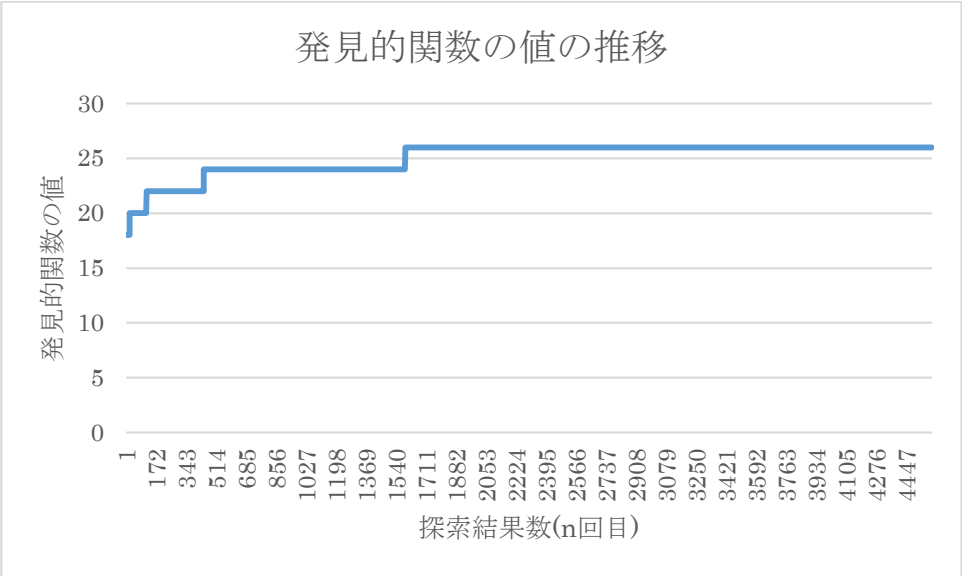

図 4 探索毎に出力した発見的関数の値