可逆小石並べゲームの実現

2019se045 小椋　響
2019se054 清水　岳

1．　はじめに

今回この課題について取り組んだ理由は、可逆講師並べゲームの理解を深め、今後の研究を進めるためにゲームそのものの実装が必要であると考えた。

2．　内容

・ｃ言語を用いて今回の課題解決に向けて取り組んだ。
・define goal にてマスＴの値を変える。
・マスに石を置くか取り除くかを選択する。
・石が置けない場合はエラーを出力
・石を取り除けない場合はエラーを出力。
・ゴールマスにおいてある石のみになったらクリア。

3．　今後の課題

・マスＴの変更にファイルの書き換えが必要である。
・最適解が計算できるプログラムの追加が必要である。

4．　参考文献

森田　憲一　第５巻　可逆計算、近代科学社(2012)
横山　哲郎　可逆小石並べゲーム

1．可逆小石並べゲーム

```
#include <stdio.h>

#define goal 9 //ゴールは 4
int num;
int i;
int mid;
int count;
int com;
int add;
int rem;
int select;
int tntn;
int cl;
int arr[goal]; // 0 ～ 4

int playergoal(void){
    printf("ゴールは:%d¥n",goal-1);
    printf("使える数は？");
    scanf("%d",&num);
    printf("使える数は:%d¥n",num);

}

int start(void){
    //配列を初期化
    for(i = 1; i < goal; i++){
        arr[i] = 0;
    }
    arr[0] = 1;
}


//最後と最初以外の 0 を確認
int judge(void){
```

```
        count = 0;
        for(i = 1; i < goal-1; i++){
                if(arr[i] == 0){
                        count = count + 1;
                }
        }
        if(count == goal-2){
                mid = 1;
        }
        else{
                mid = 0;
                }
}

//クリア条件
int clear(void){
        judge();
        if(arr[0] == 1 && arr[goal-1] ==1 && mid == 1){
                cl = 1;
                printf("ゲームクリア。おめでとう！！！");
        }
}

//繰り返し
int game(void){
        //追加か削除を確認
        clear();
        while(!(arr[0] == 1 && arr[goal-1] == 1 && mid == 1)){
                clear();
                if(cl == 1){
                        break;
                }
                tntn = 0;
                for(i = goal-1; i > 0; i--){
                        if(arr[i] == 1){
                                printf("%d:●¥n",i);
```

```
            }
            else if(arr[i] == 0){
                printf("%d:¥n",i);
            }
        }
printf("add:1 or remove:2?¥n");
scanf("%d",&com);
if(com == 1){

        for(i =1; i < goal; i++){
            if(arr[i] == 1){
                tntn = tntn + 1;
            }
        }
        // 1 を探して次のマスに追加

        printf("どこに追加する？¥n");
        scanf("%d",&select);
        if(tntn == num){
            printf("個数オーバーで追加できません¥n");
        }
        else if(select > goal-1){
            printf("追加できません¥n");
        }
        else if(arr[select-1] == 1){
            arr[select] = 1;
        }
        else{
            printf("追加できません¥n");
        }

}
else if(com == 2){

        printf("どこを消去する？¥n");
        scanf("%d",&select);
```

```
                    if(arr[select-1] == 1){
                        arr[select] = 0;
                    }
                    else if(select > goal-1){
                        printf("消去できません¥n");
                    }
                    else{
                        printf("消去できません¥n");
                    }

                }
                else{
                printf("もう一度入力¥n");
                }
            }
        }

        int main(void){
            playergoal();
            start();
            game();
        }
```

２．一般化による計算結果の出力

```
#include <stdio.h>

int i, k, n;
int a = 0;
 int ans = 0;

int power01(int k,int n){

    int i;
    for(i = 0; i < n-1; i++){
        k *= k;
    }
```

```
        return k;
}

int power02(int k, int n){

        int ans = 0;
        ans = n*(k-1) +1;

        return ans;

}

int power03(int k,int n){
        int i;
        int ans = 1;
        for(i = 0; i < n; i++){
            ans = ans*(2*k-1);
        }
        return ans;
}

int main(void){

        int k,n;

        do{
        printf("各手順における段数:");
        scanf("%d", &k);
        }
        while(k < 2);
        do{
        printf("手順を再帰的に適応した回数:");
        scanf("%d", &n);
        }
        while(n < 0);
  printf("盤のマスの数 T は: %d です¥n" ,power01(k,n));
```

```
  printf("使える石の数 S'は: %d です¥n" , power02(k,n));
  printf("クリアまでのターン数 T'は: %d です¥n" ,power03(k,n));
        return 0;
}
```