BỘ MÔN KHOA HỌC MÁY TÍNH
CT332: TRÍ TUỆ NHÂN TẠO
BÀI THỰC HÀNH SỐ 1
BIỂU DIỄN KHÔNG GIAN TRẠNG THÁI & TÌM KIẾM MÙ

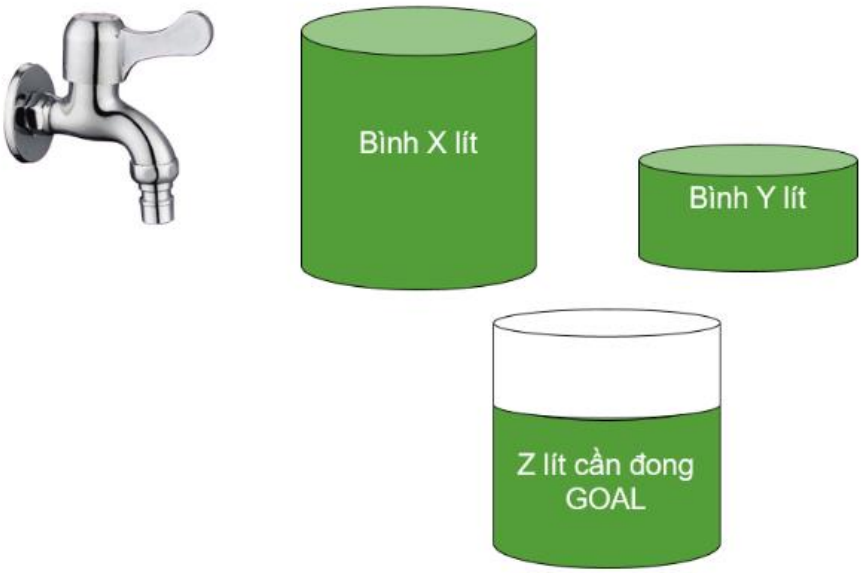
I. Mục tiêu

- ✓ Biểu diễn bài toán trên không gian trạng thái và áp dụng giải thuật tìm kiếm mù – tìm kiếm thiếu thông tin bổ sung để tìm ra giải pháp cho bài toán đong nước.
- ✓ Ngôn ngữ sử dụng: C.

II. Nội dung

1. Mô tả bài toán

Cho 2 bình nước, bình thứ nhất có sức chứa được X lít (Bình X), bình thứ hai có sức chứa Y lít (Bình Y) và 1 vòi bơm (Xem hình bên dưới). Cả hai bình X và Y đều không có vạch chia. Làm cách nào đong được Z lít. Biết rằng chúng ta có thể thực hiện các thao tác/ hành động sau đây để đong nước.



Hành động 1. Sử dụng vòi bơm để đong đầy nước bình X. Nghĩa là sau khi thực hiện hành động này thì bình nước X có được X lít nước, đúng với sức chứa của bình X.

Ví dụ: Bình X hiện tại đang có 4 lít và bình có sức chứa là 9 lít thì sau khi thực hiện hành động này thì bình X có 9 lít nước.

Hành động 2. Sử dụng vòi bơm để đong đầy nước bình Y. Tương tự như hành động 1 thì bình nước cũng sẽ có được Y lít nước, đúng với sức chứa của bình Y.

Ví dụ: Bình Y hiện tại đang có 3 lít và bình có sức chứa là 4 lít thì sau khi thực hiện hành động này thì bình Y có 4 lít nước.

Hành động 3. Rót hết nước trong X ra. Sau khi thực hiện phép toán này thì lượng nước của bình X bằng 0.

Ví dụ: Bình X hiện tại đang có 3 lít thì sau khi thực hiện hành động này thì bình X có 0 (empty) lít nước.

Hành động 4. Rót hết nước trong Y ra. Tương tự như hành động 3 thì lượng nước của bình Y bằng 0.

Ví dụ: Bình Y hiện tại đang có 2 lít thì sau khi thực hiện hành động này thì bình Y có 0 (empty) lít nước.

Hành động 5. Rót nước từ bình X sang bình Y. Ví dụ: Có hai bình X (có sức chứa là 9) và Y (có sức chứa là 4). Hiện tại lượng nước trong bình X là 8 và lượng nước trong bình Y là 3. Sau khi thực hiện hành động chuyển nước từ bình X sang bình Y. Thì lượng nước trong X là 7 và lượng nước trong bình Y là 4. Sinh viên xem thêm ví dụ trong bảng bên dưới.

Ví dụ 1:	Bình X (Sức chứa là 9)	Bình Y (Sức chứa là 4)
Lượng nước hiện tại	8	3
Sau khi rót nước từ X sang Y	7	4

Ví dụ 2:	Bình X (Sức chứa là 9)	Bình Y (Sức chứa là 4)
Lượng nước hiện tại	2	1
Sau khi rót nước từ X sang Y	0	3

Hành động 6. Rót nước từ bình Y sang bình X. Tượng tự như hành động 5. Ví dụ: Có hai bình X (có sức chứa là 9) và Y (có sức chứa là 4). Hiện tại lượng nước trong bình X là 8 và lượng nước trong bình Y là 3. Sau khi thực hiện hành động chuyển nước từ bình Y sang bình Y. Thì lượng nước trong X là 9 và lượng nước trong bình Y là 2. Sinh viên xem thêm ví dụ trong bảng bên dưới.

Ví dụ 1:	Bình X (Sức chứa là 9)	Bình Y (Sức chứa là 4)
Lượng nước hiện tại	8	3
Sau khi rót nước từ Y sang X	9	2

Ví dụ 2:	Bình X (Sức chứa là 9)	Bình Y (Sức chứa là 4)
Lượng nước hiện tại	2	1
Sau khi rót nước từ X sang Y	3	0

Ví dụ một lời giải cho bài toán đong nước:

Bình X có sức chứa là 9 và bình Y là 4. Mục đích người ta cần đong được bình nước được 6 lít nước. Dưới đây là một lời giải có thể thực hiện để đong được bình nước 6.

Bước	a	b
Start	0	0
1	9	0
2	5	4
3	5	0
4	1	4
5	1	0
6	0	1
7	9	1
8	6	4

2. Phân tích bài toán

Bài toán được mô tả trong không gian trạng thái với các đặc điểm sau:

- Trạng thái bài toán: lượng nước của mỗi bình X, Y.
- Các thao tác/ hành động (operator) để tạo ra trạng thái mới: chuyển nước từ bình X sang bình Y và ngược lại, làm đầy nước bình X/ Y, và làm rỗng (đổ ra) bình X/ Y.
- Trạng thái đầu: các bình X, Y rỗng.
- Trạng thái cuối: Bình X hoặc Y có Z lít (GOAL: mục tiêu).

Việc tìm giải pháp cho trò chơi là tìm đường đi từ trạng thái bắt đầu đến trạng thái đích. Hay nói cách khác là tìm loạt các thao tác/ hành động để đong được Z lít nước.

3. Cài đặt

3. 1. Cài đặt cấu trúc trạng thái:

Cho hai cái bình: bình X có sức chứa là 9 lít, bình Y có sức chứa là 4 lít và một vòi bơm nước. Mục tiêu cần đong được bình có 6 lít nước (GOAL);

- Đầu vào: bình X có dung tích 9 lít, bình Y có dung tích 4 lít.
- Đầu ra: 6 lít.

Sử dụng một cấu trúc gồm các trường sau để biểu diễn cho một tráng thái:

- x: lượng nước trong bình X
- y: lượng nước trong bình Y

```
#define tankcapacity_X 9 //Suc chua binh X
#define tankcapacity_Y 4 //Suc chua binh Y
#define empty 0
#define goal 6 //Muc tieu luong nuoc can dong duoc
#define Maxlength 100 //Su dung cai dat Ngan xep (Stack)

//Khai bao cau truc trang thai
typedef struct{
    int x; //Luong nuoc trong binh x
    int y; //Luong nuoc trong binh y
}State;
```

3. 2. Biểu diễn trạng thái bắt đầu

Gán lượng nước trong bình x = 0 và bình y = 0

```
//Khoi tao trang thai binh X = 0 va Y = 0
void makeNullState(State *state){
    state->x = 0;
    state->y = 0;
}
```

3. 3. In trạng thái (State): Viết hàm để in lượng nước trong bình X và bình Y.

```
//In trang thai
void print_State(State state){
    printf("\n   X:%d --- Y:%d", state.x, state.y);
}
```

3. 4. Kiểm tra trạng thái có phải trạng thái mục tiêu:

Lượng nước trong bình x hoặc bình y bằng với giá trị mục tiêu (goal)

```
//Ham kiem tra trang thai muc tieu
int goalcheck(State state){
    return (state.x==goal || state.y==goal);
}
```

3. 5. Xây dựng các hành động làm thay đổi trạng thái:

Trạng thái hiện tại là **cur_state** và kết quả sau khi thực hiện thao tác/ hành động được lưu vào ***result**. Nếu thao tác thực hiện thành công thì kết quả trả về là **1** (**int**), ngược lại trả về **0**.

STT	Tên hàm	Ý nghĩa hàm
1	**int** pourWaterFullX(State cur_state, State *result)	Làm đầy nước bình X. Sử dụng vòi bom để bom đầy nước cho bình X.
2	**int** pourWaterFullY(State cur_state, State *result)	Làm đầy nước bình Y. Sử dụng vòi bom để bom đầy nước cho bình Y.
3	**int** pourWaterEmptyX(State cur_state, State *result)	Làm rỗng nước bình X. Rót hết nước trong X ra ngoài.
4	**int** pourWaterEmptyY(State cur_state, State *result)	Làm rỗng nước bình Y. Rót hết nước trong Y ra ngoài.
5	**int** pourWaterXY(State cur_state, State *result)	Chuyển nước từ bình X sang bình Y.
6	**int** pourWaterYX(State cur_state, State *result)	Chuyển nước từ bình X sang bình Y.

3.4.1. **Hành động làm đầy nước bình X (pourWaterFullX):** hành động này được thực hiện nếu lượng nước trong bình X < sức chứa của bình X, nghĩa là bình X chưa đầy.

```
//Lam day nuoc binh X
int pourWaterFullX(State cur_state, State *result){
    if(cur_state.x < tankcapacity_X){
        result->x = tankcapacity_X;
        result->y = cur_state.y;
        return 1;
    }
    return 0;
}
```

3.4.2. **Hành động làm đầy nước bình Y (pourWaterFullY):** hành động này được thực hiện nếu lượng nước trong bình Y < sức chứa của bình Y, nghĩa là bình Y chưa đầy.

```
//Lam day nuoc binh Y
int pourWaterFullY(State cur_state, State *result){
    //Sinh viên code tương tự hàm pourWaterFullX
}
```

3.4.3. **Hành động làm rỗng nước bình X (pourWaterEmptyX):** hành động này được thực hiện nếu bình X có chứa nước (X>0).

```
//Ham lam rong nuoc trong X
int pourWaterEmptyX(State cur_state, State *result){
    if(cur_state.x>0){
        result->x = empty;
        result->y = cur_state.y;
        return 1;
    }
    return 0;
}
```

3.4.4. **Hành động làm rỗng nước bình Y (pourWaterEmptyY):** hành động này được thực hiện nếu bình Y có chứa nước (Y>0).

```
//Ham lam rong nuoc trong binh Y
int pourWaterEmptyY(State cur_state, State *result){
    //Sinh viên code tương tự hàm pourWaterEmptyX
}
```

3.4.5. **Hành động chuyển nước từ bình X sang bình Y (pourWaterXY):** hành động này được thực hiện nếu bình X có chứa nước (x>0) và bình Y chưa đầy (y<tankcapacity_Y). ***Lưu ý:*** Cần viết hai hàm max và min của hai số nguyên x và y trước khi viết hàm chuyển nước từ bình X sang bình Y.

```
//Chuyen nuoc tu binh X sang binh Y
int pourWaterXY(State cur_state, State *result){
    if(cur_state.x>0 && cur_state.y<tankcapacity_Y){
        result->x = max(cur_state.x - (tankcapacity_Y - cur_state.y), empty);
        result->y = min(cur_state.x+cur_state.y, tankcapacity_Y);
        return 1;
    }
    return 0;
}
```

3.4.6. **Hành động chuyển nước từ bình Y sang bình X (pourWaterYX):** hành động này được thực hiện nếu bình Y có chứa nước (y>0) và bình X chưa đầy (x<tankcapacity_X).

```
//Chuyen nuoc tu binh Y sang binh X
int pourWaterYX(State cur_state, State *result){
    //Sinh viên code tương tự hàm pourWaterXY
}
```

3. 6. Xây dựng hàm để gọi các hành động: Trạng thái hiện tại là **cur_state** và trạng thái kết quả là ***result.** option là số nguyên từ 1 đến 6 tương ứng với 6 hành động chuyển trạng thái của bài toán đong nước.

```
//Goi cac phep toan tren trang thai
int call_operator(State cur_state, State *result, int option){
    switch(option){
        case 1: return pourWaterFullX(cur_state, result);
        case 2: return pourWaterFullY(cur_state, result);
        case 3: return pourWaterEmptyX(cur_state, result);
        case 4: return pourWaterEmptyY(cur_state, result);
        case 5: return pourWaterXY(cur_state, result);
        case 6: return pourWaterYX(cur_state, result);
        default: printf("Error calls operators");
            return 0;
    }
}
```

3. 7. Bài tập 1: Biểu diễn trạng thái và các phép toán (hành động) của bài toán, gọi các hành động và in ra kết quả tương ứng từng hành động.

Các bước thực hiện:

- Khai báo cấu trúc trạng thái: State
- Cài đặt hàm biểu diễn trạng thái bắt đầu.
- Cài đặt các hàm hành động chuyển trạng thái: hàm làm đầy nước, làm rỗng bình, chuyển nước từ bình x sang bình y và ngược lại.
- Cài đặt hàm call_operator để gọi các hành động.

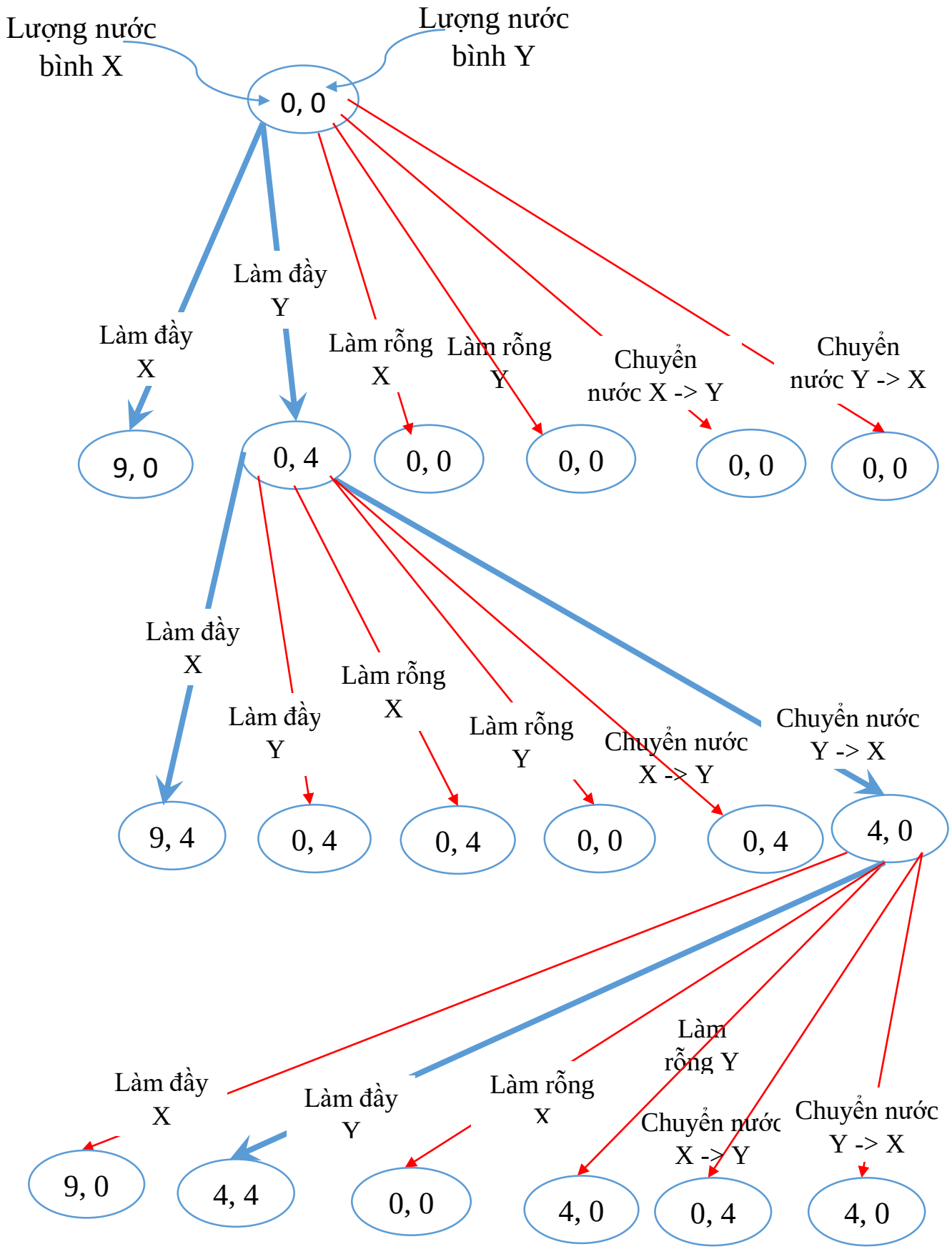
```c
#include <stdio.h>
#include <stdlib.h>

//Dinh nghia gia tri cac bien tankcapacity_X, tankcapacity_Y, empty, goal, Maxlength
//Hằng chuỗi để in ra tên các hành động
const char* action[] = {"First State","pour Water Full X","pour Water Full Y",
                        "pour Water Empty X","pour Water Empty Y",
                        "pour Water X to Y","pour Water Y to X"};

//Khai bao cau truc trang thai
//Cài đặt các hàm hành động trạng thái
//Cài đặt hàm call_operator

int main(){
    State cur_state = {5, 4}, result;
    printf("Trang thai bat dau");
    print_State(cur_state);
    int opt;
    for(opt=1; opt<=6; opt++){
        int thuchien = call_operator(cur_state, &result, opt);
        if(thuchien == 1){//thuc hien hanh dong thanh cong
            printf("\nHanh dong %s thanh cong", action[opt]);
            print_State(result);
        }
        else
            printf("\nHanh dong %s KHONG thanh cong", action[opt]);
    }
    return 0;
}
```

Kết quả chương trình:

```
Trang thai bat dau
   X:5 --- Y:4
Hanh dong pour Water Full X thanh cong
   X:9 --- Y:4
Hanh dong pour Water Full Y KHONG thanh cong
Hanh dong pour Water Empty X thanh cong
   X:0 --- Y:4
Hanh dong pour Water Empty Y thanh cong
   X:5 --- Y:0
Hanh dong pour Water X to Y KHONG thanh cong
Hanh dong pour Water Y to X thanh cong
   X:9 --- Y:0
--------------------------------
Process exited after 0.5929 seconds with return value 0
Press any key to continue . . .
```

3. 8. Mô hình minh họa dựng cây tìm kiếm – duyệt theo chiều sâu:

Lưu ý: Những cung triển khai không gian trạng thái có màu xanh (đậm) là phép toán tương ứng hợp lệ, những cung màu đỏ (nhạt) là những phép toán triển khai không hợp lệ.



3. 9. Cài đặt cấu trúc để xây dựng cây tìm kiếm Không gian trạng thái:

Sử dụng một cấu trúc gồm các trường sau để biểu diễn cho một tráng thái:

- state: dùng để lưu trạng thái lượng nước trong bình x và y
- Parent: dùng để lưu nút cha của một nút mới được triển khai.
- no_function: Lưu số thứ tự phép toán đã thực hiện tương ứng. Bài toán đong nước chúng ta có 6 phép toán.

```
/*-----------------------------------------*/
//Khai bao cau truc nut (dinh) de dung cay tim kiem
typedef struct Node{
    State state;
    struct Node* Parent;
    int no_function;
}Node;
```

Ví dụ: Nút (9,4) được tạo ra từ nút cha là nút (0,4) qua thực hiện phép toán Làm đầy X. Nút 9,4 sẽ được lưu như sau:

```
Node(9,4){
        state = (9, 4);
        Parent = Node (0, 4);
        no_function = 1;
}
```

3. 10. Cài đặt cấu trúc Ngăn xếp: để lưu trạng thái (đã duyệt/ đang chờ duyệt) trong quá trình duyệt để tìm đến trạng thái mục tiêu.

```
/*-----------------------------------------*/
//Khai bao cau truc Stack de luu trang thai duyet
typedef struct{
    Node* Elements[Maxlength];
    int Top_idx;
}Stack;
```

```
//Dua 1 phan tu len ngan xep
void push(Node* x, Stack *stack){
    if(full_Stack(*stack))
        printf("Error!Stack is full");
    else{
        stack->Top_idx -=1;
        stack->Elements[stack->Top_idx]=x;
    }
}
```

```
//Khoi tao ngan xep rong
void makeNull_Stack(Stack *stack){
    stack->Top_idx = Maxlength;
}

//Kiem tra xem ngan xep co rong hay khong
int empty_Stack(Stack stack){
    return stack.Top_idx == Maxlength;
}

/*Kiem tra ngan xep co day khong*/
int full_Stack(Stack stack){
    return stack.Top_idx == 0;
}
```

```
//Tra ve phan tu tren dinh ngan xep
Node* top(Stack stack){
    if(!empty_Stack(stack))
        return stack.Elements[stack.Top_idx];
    return NULL;
}

//Xoa phan tu tai dinh ngan xep
void pop(Stack *stack){
    if(!empty_Stack(*stack))
        stack->Top_idx+=1;
    else printf("Error! Stack is empty");
}
```

3. 11. Kiểm tra trạng thái

Quá trình duyệt cây tìm kiếm không gian trạng thái, chúng ta cần kiểm tra xem trạng thái mới sinh ra có tồn tại trong ngăn xếp (Open/ Close) hay chưa, để tránh vòng lặp xảy ra trong quá trình duyệt. Hàm **find_State** dùng để kiểm tra xem trang thái có tồn tại trong Ngăn xếp hay chưa?

```
//Tim trang thai trong Stack Open/Close
int find_State(State state, Stack openStack){
    while(!empty_Stack(openStack)){
        if(compareStates(top(openStack)->state,state))
            return 1;
        pop(&openStack);
    }
    return 0;
}
```

3. 12. Duyệt chiều sâu trên Cây tìm kiếm không gian trạng thái (depth-first search):

/**/

Giải thuật tìm kiếm theo chiều sâu depth-first search (dfs)

Procedure depth – first –search;

Begin % khởi đầu

Open:= [start]; Closed:= [ ]; // được cài đặt bằng ngăn xếp

While open ≠ [ ] do % còn các trạng thái chưa khảo sát

Begin

Lấy một phần tử từ ngăn xếp open, gọi nó là X;

If X là mục tiêu then trả lời kết quả (thành công) % tìm thấy đích else begin

Phát sinh các con của X;

Đưa X vào ngăn xếp closed;

Loại các con của X nằm trong open hoặc closed;

Đưa các con còn lại vào ngăn xếp open;

End;

Trả lời kết quả (thất bại); % không còn trạng thái nào

End;

/**/

```
//Thuat toan duyet theo chieu sau
Node* DFS_Algorithm(State state){
    //Khai bao hai ngan xep Open va Close
    Stack Open_DFS;
    Stack Close_DFS;
    makeNull_Stack(&Open_DFS);
    makeNull_Stack(&Close_DFS);
    //Tao nut trang thai cha
    Node* root = (Node*)malloc(sizeof(Node));
    root->state = state;
    root->Parent = NULL;
    root->no_function = 0;
    push(root, &Open_DFS);
    while(!empty_Stack(Open_DFS)){
        //Lay mot dinh trong ngan xep
        Node* node = top(Open_DFS);
        pop(&Open_DFS);
        push(node, &Close_DFS);
        //Kiem tra xem dinh lay ra co phai trang thai muc tieu?
        if(goalcheck(node->state))
            return node;
        int opt;
        //Goi cac phep toan tren trang thai
        for(opt=1; opt<=6; opt++){
            State newstate;
            makeNullState(&newstate);
            if(call_operator(node->state, &newstate, opt)){
                //Neu trang thai moi sinh ra da ton tai thi bo qua
                if(find_State(newstate, Close_DFS) || find_State(newstate, Open_DFS))
                    continue;
                //Neu trang thai moi chua ton tai thi them vao ngan xep
                Node* newNode = (Node*)malloc(sizeof(Node));
                newNode->state = newstate;
                newNode->Parent = node;
                newNode->no_function = opt;
                push(newNode, &Open_DFS);
            }
        }
    }
    return NULL;
}
```

3. 13. In kết quả duyệt chiều sâu:

```
//In ket qua chuyen nuoc de dat den trang thai muc tieu
void print_WaysToGetGoal(Node* node){
    Stack stackPrint;
    makeNull_Stack(&stackPrint);
    //Duyet nguoc ve nut parent de
    while(node->Parent != NULL){
        push(node, &stackPrint);
        node = node->Parent;
    }
    push(node, &stackPrint);
    //In ra thu tu hanh dong chuyen nuoc
    int no_action = 0;
    while(!empty_Stack(stackPrint)){
        printf("\nAction %d: %s", no_action, action[top(stackPrint)->no_function]);
        print_State(top(stackPrint)->state);
        pop(&stackPrint);
        no_action++;
    }
}
```

3. 14. Bài tập 2:

Biểu diễn trạng thái và các phép toán (hành động) của bài toán, gọi các hành động và xây dựng cây tìm kiếm không gian trạng thái theo chiều sâu cho bài toán đong nước.

Các bước thực hiện:

- Khai báo các hằng cần thiết cho bài toán đong nước
- Khai báo cấu trúc trạng thái: State
- Cài đặt các hàm hành động chuyển trạng thái: hàm làm đầy nước, làm rỗng bình, chuyển nước từ bình x sang bình y và ngược lại, và những hàm liên quan đến State.
- Cài đặt cấu trúc Node để hỗ trợ xây dựng cây trong quá trình duyệt.
- Cài đặt cấu trúc ngăn xếp để lưu trữ trạng thái duyệt.
- Cài đặt thuật toán toán duyệt theo chiều sâu (DFS)
- Cài đặt hàm in kết quả tìm kiếm.
- Viết hàm **main()** cho chương trình.

```c
#include <stdio.h>
#include <stdlib.h>
//1. Khai báo các hằng cần thiết cho bài toán đong nước
//2. Khai báo cấu trúc trạng thái (State)
//3. Cài đặt các hàm chuyển trạng thái (hành động)
// và các hàm cần thiết liên quan đến trạng thái(State)
/*----------------------------------------*/
//Khai báo cấu trúc nút (đỉnh) để xây dựng cây tìm kiếm
typedef struct Node{
    ..............
}Node;

/*----------------------------------------*/
//Khai bao cau truc Stack để lưu trạng thái duyệt
typedef struct{
    ..............
}Stack;

//Cài đặt các hàm cần thiết đến cấu trúc Stack

//Hàm kiểm tra trạng thái có nằm trong Stack
int find_State(State state, Stack openStack){
    ..............
}
```

```c
//Thuat toan duyet theo chieu sau
Node* DFS_Algorithm(State state){
    ..............
}

//In ket qua chuyen nuoc de dat den trang thai muc tieu
void print_WaysToGetGoal(Node* node){
    ..............
}
int main(){
    State cur_state = {0, 0};
    Node* p = DFS_Algorithm(cur_state);
    print_WaysToGetGoal(p);
    return 0;
}
```

Kết quả chương trình:

```
Action 0: First State
   X:0 --- Y:0
Action 1: pour Water Full Y
   X:0 --- Y:4
Action 2: pour Water Y to X
   X:4 --- Y:0
Action 3: pour Water Full Y
   X:4 --- Y:4
Action 4: pour Water Y to X
   X:8 --- Y:0
Action 5: pour Water Full Y
   X:8 --- Y:4
Action 6: pour Water Y to X
   X:9 --- Y:3
Action 7: pour Water Empty X
   X:0 --- Y:3
Action 8: pour Water Y to X
   X:3 --- Y:0
Action 9: pour Water Full Y
   X:3 --- Y:4
Action 10: pour Water Y to X
   X:7 --- Y:0
Action 11: pour Water Full Y
   X:7 --- Y:4
Action 12: pour Water Y to X
   X:9 --- Y:2
Action 13: pour Water Empty X
   X:0 --- Y:2
Action 14: pour Water Y to X
   X:2 --- Y:0
Action 15: pour Water Full Y
   X:2 --- Y:4
Action 16: pour Water Y to X
   X:6 --- Y:0
--------------------------------
Process exited after 0.6994 seconds with return value 0
Press any key to continue . . .
```

3. 15. Bài tập 3:

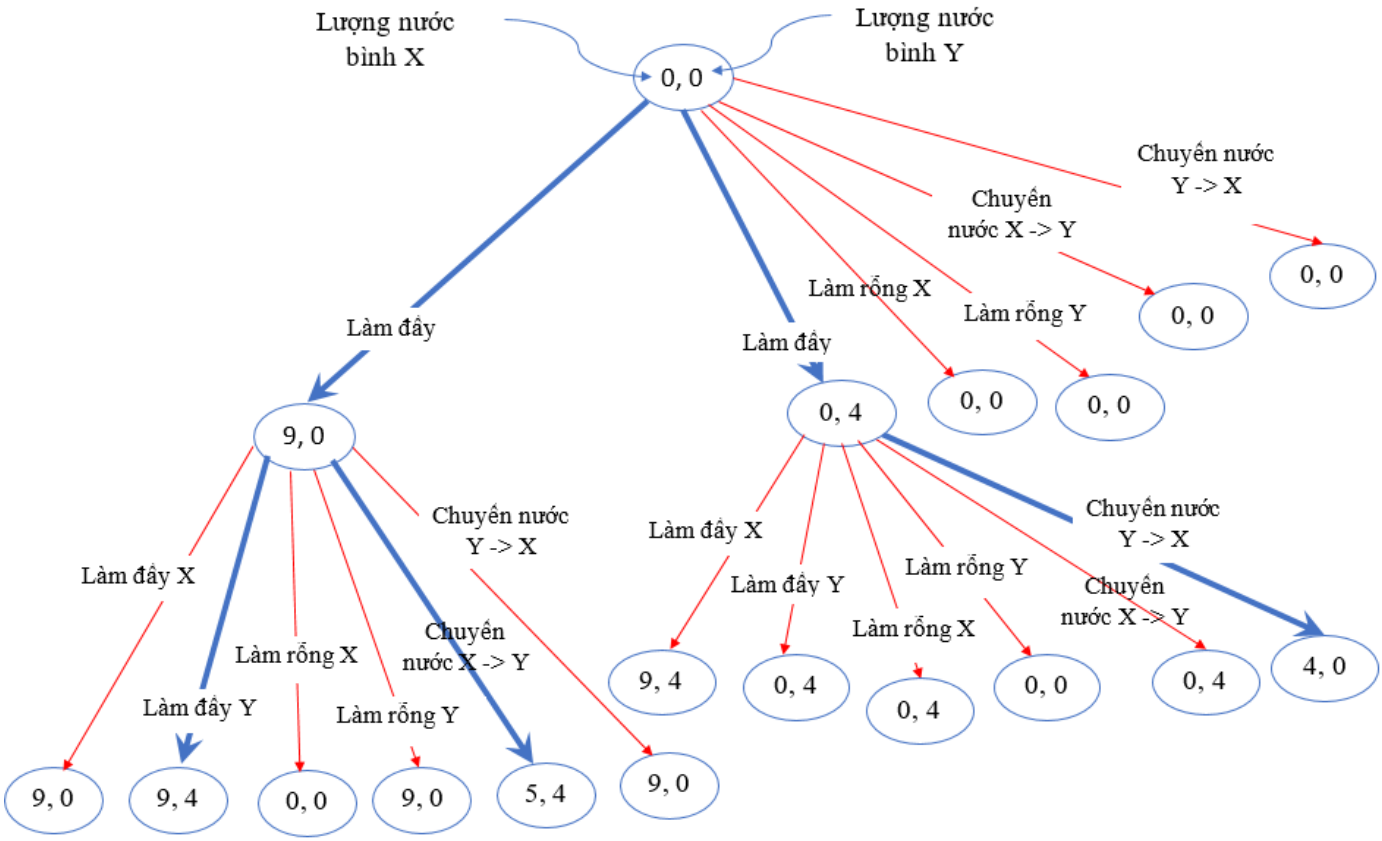
Biểu diễn trạng thái và các phép toán (hành động) của bài toán, gọi các hành động và xây dựng cây tìm kiếm không gian trạng thái theo chiều rộng (DBF) cho bài toán đong nước.

Các bước thực hiện:

- Khai báo các hằng cần thiết cho bài toán đong nước
- Khai báo cấu trúc trạng thái: State
- Cài đặt các hàm hành động chuyển trạng thái: hàm làm đầy nước, làm rỗng bình, chuyển nước từ bình x sang bình y và ngược lại, và những hàm liên quan đến State.
- Cài đặt cấu trúc Node để hỗ trợ xây dựng cây trong quá trình duyệt.
- Cài đặt cấu trúc Hàng đợi (QUEUE) để lưu trữ trạng thái duyệt.
- Cài đặt thuật toán toán duyệt theo chiều rộng (BFS). Sinh viên lưu ý Open và Close được khai báo kiểu QUEUE và sử dụng các hàm tương ứng với QUEUE để đưa một phần tử vào hàng đợi hoặc lấy một phần tử từ hàng đợi.
- Cài đặt hàm in kết quả tìm kiếm.
- Viết hàm main() cho chương trình.

Mô hình minh họa cây tìm kiếm, duyệt theo chiều rộng:

Lưu ý: Những cung triển khai không gian trạng thái có màu xanh (đậm) là phép toán tương ứng hợp lệ, những cung màu đỏ (nhạt) là những phép toán triển khai không hợp lệ.



Cài đặt cấu trúc hàng đợi

```
/*-----------------------------------------*/
//Khai bao cau truc Queue (hang doi) de luu trang thai duyet
typedef struct{
    Node* Elements[Maxlength];
    int front, rear;
}Queue;

//Khoi tao hang doi rong
void makeNull_Queue(Queue *queue){
    queue->front = -1;
    queue->rear = -1;
}

//Kiem tra xem hang doi co rong hay khong
int empty_Queue(Queue queue){
    return queue.front == -1;
}
```

```
//Kiem tra xem hang doi co day hay khong
int full_Queue(Queue queue){
    return ((queue.rear-queue.front+1)%Maxlength)==0;
}

//Tra ve phan tu dau hang doi
Node* get_Front(Queue queue){
    if(empty_Queue(queue))
        printf("Queue is empty");
    else
        return queue.Elements[queue.front];
}
```

```
//Xoa bo mot phan tu khoi hang doi
void del_Queue(Queue *queue){
    if(!empty_Queue(*queue)){
        if(queue->front == queue->rear)
            makeNull_Queue(queue);
        else queue->front = (queue->front+1)%Maxlength;
    }
    else printf("Error, Delete");
}

//Them phan tu vao hang doi
void push_Queue(Node* x, Queue *queue){
    if(!full_Queue(*queue)){
        if(empty_Queue(*queue))
            queue->front = 0;
        queue->rear = (queue->rear+1) % Maxlength;
        queue->Elements[queue->rear]=x;
    }
    else printf("Error, Push");
}
```

3. 16. Bài tập 4: Sử dụng thư viện C++.

Biểu diễn trạng thái và các phép toán (hành động) của bài toán, gọi các hành động và xây dựng cây tìm kiếm không gian trạng thái theo chiều sâu (DFS) cho bài toán đong nước. Để duyệt theo chiều sâu, chúng ta cần sử dụng Ngăn xếp (Stack) để lưu trữ các đỉnh trong quá trình duyệt. Bài tập này các em không cần cài đặt cấu trúc Stack mà các em sử dụng

thư viện có sẵn của thư viên STL. Để sử dụng tính năng này, em phải viết chương trình bằng ngôn ngữ C++ và phải đặt tên chương trình có phần mở rộng *.cpp hoặc *.cc.

Các bước thực hiện:

- Khai báo các hằng cần thiết cho bài toán đong nước
- Khai báo cấu trúc trạng thái: State
- Cài đặt các hàm hành động chuyển trạng thái: hàm làm đầy nước, làm rỗng bình, chuyển nước từ bình x sang bình y và ngược lại, và những hàm liên quan đến State.
- Cài đặt cấu trúc Node để hỗ trợ xây dựng cây trong quá trình duyệt.
- Cài đặt thuật toán toán duyệt theo chiều sâu (DFS) dưới sụ hỗ trợ của thư viên STL.
- Cài đặt hàm in kết quả tìm kiếm.
- Viết hàm main() cho chương trình.

Thuật toán DFS được cài đặt bằng sự hỗ trợ của thư viện C++.

```
//Thuat toan duyet theo chieu sau
Node* DFS_Algorithm(State state){
    //Khai bao hai ngan xep Open va Close
    stack<Node*> Open_DFS;
    stack<Node*> Close_DFS;
    //Tao nut trang thai cha
    Node* root = (Node*)malloc(sizeof(Node));
    root->state = state;
    root->Parent = NULL;
    root->no_function = 0;
    Open_DFS.push(root);
    while(!Open_DFS.empty()){
        //Lay mot dinh trong ngan xep
        Node* node = Open_DFS.top();
        Open_DFS.pop();
        Close_DFS.push(node);
        //Kiem tra xem dinh lay ra co phai trang thai muc tieu?
        if(goalcheck(node->state))
            return node;
        int opt;
        //Goi cac phep toan tren trang thai
        for(opt=1; opt<=6; opt++){
            State newstate;
            makeNullState(&newstate);
            if(call_operator(node->state, &newstate, opt)){
                //Neu trang thai moi sinh ra da ton tai thi bo qua
                if(find_State(newstate, Close_DFS) || find_State(newstate, Open_DFS))
                    continue;
                //Neu trang thai moi chua ton tai thi them vao ngan xep
                Node* newNode = (Node*)malloc(sizeof(Node));
                newNode->state = newstate;
                newNode->Parent = node;
                newNode->no_function = opt;
                Open_DFS.push(newNode);
            }
        }
    }
    return NULL;
}
```

Hàm in kết quả tìm kiếm trạng thái:

```
//In ket qua chuyen nuoc de dat den trang thai muc tieu
void print_WaysToGetGoal(Node* node){
    stack<Node*> stackPrint;
    //Duyet nguoc ve nut parent de
    while(node->Parent != NULL){
        stackPrint.push(node);
        node = node->Parent;
    }
    stackPrint.push(node);
    //In ra thu tu hanh dong chuyen nuoc
    int no_action = 0;
    while(!stackPrint.empty()){
        printf("\nAction %d: %s", no_action, action[stackPrint.top()->no_function]);
        print_State(stackPrint.top()->state);
        stackPrint.pop();
        no_action++;
    }
}
```

Hàm kiểm tra xem một trạng thái có thuộc ngăn xếp (Open/Close)

```
//Tim trang thai trong Stack Open/Close
int find_State(State state, stack<Node*> openStack){
    while(!openStack.empty()){
        if(compareStates(openStack.top()->state,state))
            return 1;
        openStack.pop();
    }
    return 0;
}
```

Chương trình cài đặt tìm kiếm trạng thái với sự hỗ trợ của STL.

```
//Chương trình lưu file *.cpp hoặc *.cc
#include <stdio.h>
#include <stdlib.h>
#include <stack> //Lưu ý nhớ thêm thư viện stack

using namespace std; //Lưu ý nhớ thêm dòng này

//1. Khai báo các hằng cần thiết cho bài toán đong nước
//2. Khai báo cấu trúc trạng thái (State)
//3. Cài đặt các hàm chuyển trạng thái (hành động)
// và các hàm cần thiết liên quan đến trạng thái(State)
/*----------------------------------------*/
//Khai bao cau truc nut (dinh) de dung cay tim kiem
typedef struct Node{
    ..............
}Node;

//Hàm kiểm tra trạng thái có nằm trong Stack
int find_State(State state, stack<Node*> openStack){
    ..............
}
//Thuat toan duyet theo chieu sau
Node* DFS_Algorithm(State state){
    ..............
}
```

```
//In ket qua chuyen nuoc de dat den trang thai muc tieu
void print_WaysToGetGoal(Node* node){
    ..............
}
int main(){
    State cur_state = {0, 0};
    Node* p = DFS_Algorithm(cur_state);
    print_WaysToGetGoal(p);
    return 0;
}
```

3. 17. Bài tập 5: Sử dụng thư viện C++

Biểu diễn trạng thái và các phép toán (hành động) của bài toán, gọi các hành động và xây dựng cây tìm kiếm không gian trạng thái theo chiều rộng (BFS) cho bài toán đong nước. Để duyệt theo chiều rộng, chúng ta cần sử dụng Hàng đợi (Queue) để lưu trữ các đỉnh trong quá trình duyệt. Bài tập này các em không cần cài đặt cấu trúc Queue mà các em sử dụng thư viện có sẵn của thư viên STL. Để sử dụng tính năng này, em phải viết chương trình bằng ngôn ngữ C++ và phải đặt tên chương trình có phần mở rộng *.cpp hoặc *.cc.

Các bước thực hiện:

- Khai báo các hằng cần thiết cho bài toán đong nước
- Khai báo cấu trúc trạng thái: State
- Cài đặt các hàm hành động chuyển trạng thái: hàm làm đầy nước, làm rỗng bình, chuyển nước từ bình x sang bình y và ngược lại, và những hàm liên quan đến State.

- Cài đặt cấu trúc Node để hỗ trợ xây dựng cây trong quá trình duyệt.
- Cài đặt thuật toán toán duyệt theo chiều rộng (BFS) dưới sụ hỗ trợ của thư viên STL. Nhớ **#include<queue>** vào chương trình.
- Cài đặt hàm in kết quả tìm kiếm.
- Viết hàm main() cho chương trình.

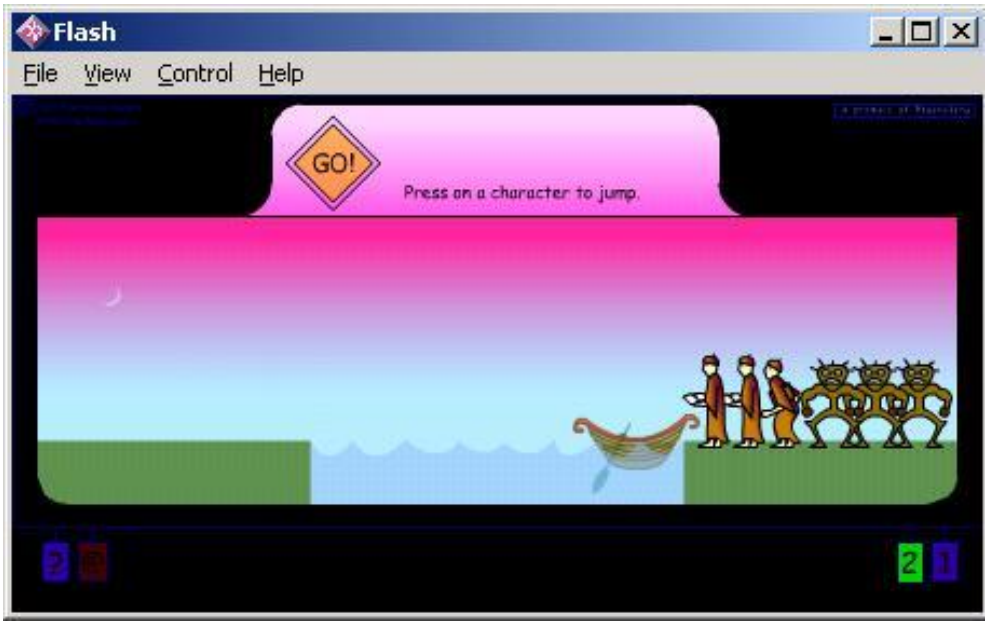
III. Bài tập sinh viên tự học:

Bài tập 1: Cần đong 8 lít từ bình 10 lít chứa đầy sữa, và 2 bình 5 lít và 6 lít rỗng, không có vạch chia. Hãy phân tích các thành phần của bài toán và áp dụng chiến thuật tìm kiếm sâu/ rộng để tìm giải pháp cho bài toán.

Bài tập 2: Cần đong 4 lít từ bình 8 lít chứa đầy sữa, và 2 bình 5 lít và 3 lít rỗng, không có vạch chia. Hãy phân tích các thành phần của bài toán và áp dụng chiến thuật tìm kiếm sâu/ rộng để tìm giải pháp cho bài toán.

Bài tập 3:

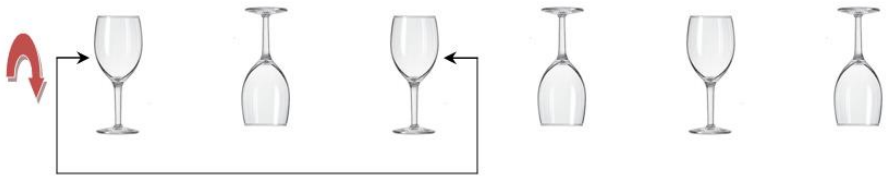
- **Bài toán 3 tu sĩ và 3 con quỷ:**
  - Có 3 tu sĩ và 3 con quỷ đang ở bờ sông và cả **6 đều muốn qua bờ bên kia.**



  - Dưới sông có một chiếc thuyền. Thuyền không thể tự mình di chuyển được mà phải có người chèo. Thuyền chỉ có thể chở tối đa 2 hành khách.
  - Trên mỗi bờ sông, nếu số lượng quỷ nhiều hơn số tu sĩ, thì các tu sĩ sẽ bị ăn thịt.
- **Yêu cầu:**
  - Mô hình hóa bài toán về dạng tìm kiếm trong không gian trạng thái.
- **Gợi ý:**
  - Cần quản lý số lượng tu sĩ và số lượng quỷ của một bờ sông. Chúng ta có thể thực hiện phép tính trừ để biết số lượng tu sĩ của quỷ của bờ sông còn lại. Em có thể sử dụng cấu trúc sau đây để cài đặt cho State bài toán này.

```c
//Khai bao cau truc trang thai
typedef struct{
    int so_tusi; //So luong tu si bo A
    int so_quy; //so luong quy bo A
    char vitri_thuyen; //Vi tri cua thuyen
}State;
```

Bài tập 4: Có 3 cái ly úp và 3 cái ly ngửa đặt thành một hàng như hình vẽ bên dưới:



- Một con robot được lập trình để lật tất cả các ly thành ngửa. Do có cấu tạo khá đặc biệt nên mỗi lần thực hiện việc lật ly, con robot lại đổi trạng thái (úp thành ngửa và ngửa thành úp) của **cả 3 ly cạnh nhau**.
- Mô hình hóa bài toán về dạng tìm kiếm trong không gian trạng thái.