نحوه ی مدلسازی : از یک کلاس search که تمامی حالات مختلف search در آن به صورت تابع قرارداده شده استفاده شده. این توابع به هیچ وجه نباید به نحوه ی پیاده سازی کلاس های , problem state , action ارتباط داشته باشند. سپس هر مسیله را با استفاده از کلاس های problem , action state , مدل شده است که هر کلاس های state , action از کلاس های پدرشان ارث برده اند که کد دسته بندی شود و کار ها مشخص شود . کلاس problem نیز از یک interface , ایمپلمنت کرده است.

الف)

ساختار state:

تعداد آدم ها و زامبی ها در سمت چپ و راست و محل قایق را ذخیره میکند.

```
public class StateZombies extends State{
    int leftZombies;
    int leftHuman;
    int rightZombies;
    int rightHuman;
    boolean boatIsLeft;
    public StateZombies(int leftZombies, int leftHuman, int rightZombies, int rightHuman , boolean boatIsLeft) {
        this.leftZombies = leftZombies;
        this.leftHuman = leftHuman;
        this.rightZombies = rightZombies;
        this.rightHuman = rightHuman;
        this.boatIsLeft = boatIsLeft;
    }
    @Override
    public String print() {
        String str = "";
        if (boatIsLeft) {
            str = "( z="+leftZombies + " h=" + leftHuman + " ~ , z="+rightZombies + " h=" + rightHuman + ")";
        }
        else {
            str = "( z="+leftZombies + " h=" + leftHuman + " , z="+rightZombies + " h=" + rightHuman + " ~ )";
        }
        return str;
    }

}
```

تابع تست هدف:

اگر ۳ انسان و ۳ زامبی سمت راست داشته باشیم و قایق نیز سمت راست باشد حالت پایانی است.

```
@Override
public boolean goalTest(State s_inter) {
    StateZombies s = (StateZombies)s_inter;
    if (s.leftZombies == 0 && s.leftHuman == 0 && s.rightZombies == 3 && s.rightHuman == 3 && s.boatIsLeft == false) {
        return true;
    }
    return false;
}
```

تابع actions:

اگر قایق سمت چپ باشد به تعداد انسان های سمت چپ و زامبی های سمت چپ میشود برد سمت راست به شرطی که قایق بیش از ۲ نفر پر نشده باشد و در هیچ سمتی تعداد زامبی ها بیشتر از انسان ها نشود که با تابع result هندل شده است.به گونه ای که هر اکشنی که تولید میشود چک میشود که حالت غلط تولید نکند اگر حالت غلط تولید نکرد به مجموعه اکشن های صحیح اضافه میشود.
به همین ترتیب به صورت برعکس برای اینکه قایق سمت چپ باشد.

```
@Override
public ArrayList<Action> actions(State state_inter) {
    StateZombies state = (StateZombies)state_inter;
    StateZombies
                    actions = new ArrayList<Action>();
                    Press 'F2' for focus
                    eft) {
                    0; i <= Math.min ( state.leftZombies , 2 ) ; i++) {
                    j = 0; j <= Math.min (state.leftHuman , 2 ); j++) {
                if ((i + j) > 0  && (i+j) <=2) {
                    ActionZombies action = new ActionZombies(i , j);
                    StateZombies nxtState = this.result(state, action);
                    if ((nxtState.leftZombies <= nxtState.leftHuman || nxtState.leftHuman == 0)  &&  (nxtState.rightZombies <= nxtState.rightHuman || nxtState.rightHuman ==
                        actions.add(action);
                    }
                }

            }
        }
    }
    else {
        for (int i = 0; i <= Math.min(state.rightZombies , 2 ); i++) {
            for (int j = 0; j <= Math.min(state.rightHuman , 2); j++) {
                if ((i + j) > 0  && (i+j) <=2) {
                    ActionZombies action = new ActionZombies(i , j);
                    StateZombies nxtState = this.result(state, action);
                    if ((nxtState.leftZombies <= nxtState.leftHuman || nxtState.leftHuman == 0)  &&  (nxtState.rightZombies <= nxtState.rightHuman || nxtState.rightHuman ==
                        actions.add(action);
                    }
                }

            }
        }
    }
    return actions;
}
```

تابع result :

اگر قایق سمت چپ باشد یعنی حرکت بعدی ما رفتن قایق از چپ به راست است پس به تعداد انسان ها و زامبی هایی که در قایق اند به انسان ها و زامبی های سمت راست اضافه و از سمت چپ کم میشود
همینطور برای آنکه قایق سمت راست باشد.

```
@Override
public StateZombies result(State state_inter, Action action_inter) {
    StateZombies state = (StateZombies)state_inter;
    ActionZombies action = (ActionZombies)action_inter;

    int leftZombies , leftHuman , rightZombies , rightHuman;
    if (state.boatIsLeft) {
        rightZombies = state.rightZombies + action.zombieTransfer;
        rightHuman = state.rightHuman + action.humanTransfer;
        leftZombies = state.leftZombies - action.zombieTransfer;
        leftHuman = state.leftHuman - action.humanTransfer;
    }
    else {
        rightZombies = state.rightZombies - action.zombieTransfer;
        rightHuman = state.rightHuman - action.humanTransfer;
        leftZombies = state.leftZombies + action.zombieTransfer;
        leftHuman = state.leftHuman + action.humanTransfer;
    }
    return new StateZombies(leftZombies, leftHuman, rightZombies, rightHuman, !state.boatIsLeft);
}
```

ب)
~ علامت قایق است

خروجی Bfs

numOfObservedNodes: 15

numOfExtendedNodes: 13

max memory: 14

pathStates : (z=3 h=3 ~ , z=0 h=0) (z=2 h=2 , z=1 h=1 ~) (z=2 h=3 ~ , z=1 h=0) (z=0 h=3 , z=3 h=0 ~) (z=1 h=3 ~ , z=2 h=0) (z=1 h=1 , z=2 h=2 ~) (z=2 h=2 ~ , z=1 h=1) (z=2 h=0 , z=1 h=3 ~) (z=3 h=0 ~ , z=0 h=3) (z=1 h=0 , z=2 h=3 ~) (z=1 h=1 ~ , z=2 h=2) (z=0 h=0 , z=3 h=3 ~)

pathActions : (z=1 , h=1) (z=0 , h=1) (z=2 , h=0) (z=1 , h=0) (z=0 , h=2) (z=1 , h=1) (z=0 , h=2) (z=1 , h=0) (z=2 , h=0) (z=0 , h=1) (z=1 , h=1)

pathCost : 11.00.0

خروجی Dfs depth incremental

numOfObservedNodes: 100

numOfExtendedNodes: 85

max memory: 14

pathStates : (z=3 h=3 ~ , z=0 h=0) (z=1 h=3 , z=2 h=0 ~) (z=2 h=3 ~ , z=1 h=0) (z=0 h=3 , z=3 h=0 ~) (z=1 h=3 ~ , z=2 h=0) (z=1 h=1 , z=2 h=2 ~) (z=2 h=2 ~ , z=1 h=1) (z=2 h=0 , z=1 h=3 ~) (z=3 h=0 ~ , z=0 h=3) (z=1 h=0 , z=2 h=3 ~) (z=2 h=0 ~ , z=1 h=3) (z=0 h=0 , z=3 h=3 ~)

pathActions : (z=2 , h=0) (z=1 , h=0) (z=2 , h=0) (z=1 , h=0) (z=0 , h=2) (z=1 , h=1) (z=0 , h=2) (z=1 , h=0) (z=2 , h=0) (z=1 , h=0) (z=2 , h=0)

pathCost : 11.00.0

خروجی Bidirectional

numOfObservedNodes: 17

numOfExtendedNodes: 15

max memory: 16

pathStates : (z=3 h=3 ~ , z=0 h=0) (z=2 h=2 , z=1 h=1 ~) (z=2 h=3 ~ , z=1 h=0) (z=0 h=3 , z=3 h=0 ~) (z=1 h=3 ~ , z=2 h=0) (z=1 h=1 , z=2 h=2 ~) (z=2 h=2 ~ , z=1 h=1) (z=2 h=0 , z=1 h=3 ~) (z=3 h=0 ~ , z=0 h=3) (z=1 h=0 , z=2 h=3 ~) (z=1 h=1 ~ , z=2 h=2) (z=0 h=0 , z=3 h=3 ~)

pathActions : (z=1 , h=1) (z=0 , h=1) (z=2 , h=0) (z=1 , h=0) (z=0 , h=2) (z=1 , h=1) (z=0 , h=2) (z=1 , h=0) (z=2 , h=0) (z=0 , h=1) (z=1 , h=1)

pathCost : 11

یک نمونه ی کامل از خروجی های خواسته شده :

```
( z=3 h=3 ~ , z=0 h=0) : removed from f array and added to e array(expand p)
check goal test for state : ( z=2 h=3 , z=1 h=0 ~ )
( z=2 h=3 , z=1 h=0 ~ ) : added to f array
check goal test for state : ( z=2 h=2 , z=1 h=1 ~ )
( z=2 h=2 , z=1 h=1 ~ ) : added to f array
check goal test for state : ( z=1 h=3 , z=2 h=0 ~ )
( z=1 h=3 , z=2 h=0 ~ ) : added to f array
( z=2 h=3 , z=1 h=0 ~ ) : removed from f array and added to e array(expand p)
( z=2 h=2 , z=1 h=1 ~ ) : removed from f array and added to e array(expand p)
check goal test for state : ( z=2 h=3 ~ , z=1 h=0)
( z=2 h=3 ~ , z=1 h=0) : added to f array
( z=1 h=3 , z=2 h=0 ~ ) : removed from f array and added to e array(expand p)
check goal test for state : ( z=2 h=3 ~ , z=1 h=0)
( z=2 h=3 ~ , z=1 h=0) : removed from f array and added to e array(expand p)
check goal test for state : ( z=0 h=3 , z=3 h=0 ~ )
( z=0 h=3 , z=3 h=0 ~ ) : added to f array
( z=0 h=3 , z=3 h=0 ~ ) : removed from f array and added to e array(expand p)
check goal test for state : ( z=1 h=3 ~ , z=2 h=0)
( z=1 h=3 ~ , z=2 h=0) : added to f array
( z=1 h=3 ~ , z=2 h=0) : removed from f array and added to e array(expand p)
check goal test for state : ( z=1 h=1 , z=2 h=2 ~ )
( z=1 h=1 , z=2 h=2 ~ ) : added to f array
( z=1 h=1 , z=2 h=2 ~ ) : removed from f array and added to e array(expand p)
check goal test for state : ( z=2 h=2 ~ , z=1 h=1)
( z=2 h=2 ~ , z=1 h=1) : added to f array
( z=2 h=2 ~ , z=1 h=1) : removed from f array and added to e array(expand p)
check goal test for state : ( z=2 h=0 , z=1 h=3 ~ )
( z=2 h=0 , z=1 h=3 ~ ) : added to f array
( z=2 h=0 , z=1 h=3 ~ ) : removed from f array and added to e array(expand p)
check goal test for state : ( z=3 h=0 ~ , z=0 h=3)
( z=3 h=0 ~ , z=0 h=3) : added to f array
( z=3 h=0 ~ , z=0 h=3) : removed from f array and added to e array(expand p)
check goal test for state : ( z=1 h=0 , z=2 h=3 ~ )
( z=1 h=0 , z=2 h=3 ~ ) : added to f array
( z=1 h=0 , z=2 h=3 ~ ) : removed from f array and added to e array(expand p)
check goal test for state : ( z=1 h=1 ~ , z=2 h=2)
( z=1 h=1 ~ , z=2 h=2) : added to f array
check goal test for state : ( z=2 h=0 ~ , z=1 h=3)
```

(z=2 h=0 ~ , z=1 h=3) : added to f array
(z=1 h=1 ~ , z=2 h=2) : removed from f array and added to e array(expand p)
check goal test for state : (z=0 h=0 , z=3 h=3 ~)
found final node!
numOfObservedNodes: 15
numOfExtendedNodes: 13
max memory: 14
pathStates : (z=3 h=3 ~ , z=0 h=0) (z=2 h=2 , z=1 h=1 ~) (z=2 h=3 ~ , z=1 h=0) (z=0 h=3 , z=3 h=0 ~) (z=1 h=3 ~ , z=2 h=0) (z=1 h=1 , z=2 h=2 ~) (z=2 h=2 ~ , z=1 h=1) (z=2 h=0 , z=1 h=3 ~) (z=3 h=0 ~ , z=0 h=3) (z=1 h=0 , z=2 h=3 ~) (z=1 h=1 ~ , z=2 h=2) (z=0 h=0 , z=3 h=3 ~)
pathActions : (z=1 , h=1) (z=0 , h=1) (z=2 , h=0) (z=1 , h=0) (z=0 , h=2) (z=1 , h=1) (z=0 , h=2) (z=1 , h=0) (z=2 , h=0) (z=0 , h=1) (z=1 , h=1)
pathCost : 11.00.0