University of West Attica (UniWA)
Dept. Of Informatics and Computer Engineering
Athens, Greece

Programming Assignment

Course: Artificial Intelligence
Professor: Katerina Georgouli
Author: Nikola Micic

26.12.2019.

Contents

Description of the Programming Assignment

1. Describe the nature of the problem
2. Describe its world (elements, transition operators, starting state)
3. Choose a convenient search algorithm to solve the problem and explain the reason of your choice
    a. Describe the structure of start state
    b. Give the search tree that the algorithm will produce
    c. Give the fridge and the queue that the algorithm will build in each step of search until to reach the goal
    d. Give the solution
4. Solve the save problem using an Expert System and describe the similarities and difference of this approach from the above algorithmic approach for the same problem
    a. Give the facts of the problem in a convenient structure
    b. Design the rules of the ES giving priorities
    c. Describe the content of the conflict set and the rule that fires in each running cycle
    d. Give the solution
5. Use pseudo-language wherever needed.

Description of the problem of building evacuation (Nature of the problem)

The elevator of a three-floor residential building is automatically turned on in case of evacuation needs ordered by the police for any reason of common security.

The building has a ground floor, 3 floors and an elevator with a capacity of 5 people to move the residents from the 3 floors to the ground floor. When the police send the signal for evacuation, the elevator is on the ground floor and is empty. On the first floor wait 2 residents, on the second 6 and on the third 4.

The elevator can move up and down, from any floor to any floor (but not to the ground floor) in condition that:

- There is still place for people to board (it contains less than 5 people) and
- There is at least one resident waiting to the floor.

The elevator moves from any floor to the ground floor only if it is full or alternatively if no other resident waits at a floor to board.

When the elevator reaches a floor, the door opens and it is allowed to let as many people to board as possible and the rest (if any) wait for the elevator to come again.

The aim of the problem is to successfully complete the evacuation, i.e. all the residents to be moved to the ground floor.

We are required to apply a searching technique in order to find the solution, i.e. steps will need to be taken to evacuate people from building.

Description of the problem's world

The building has a ground floor plus 3 floors and one elevator.
An elevator is unique by the amount of floors it manages and the maximum number of people in elevator in one moment.
Number of residents by floors:

- On the first floor wait 2 residents
- On the second floor wait 6 residents
- On the third floor wait 4 residents

List of the people will be represented in the form of {a,b,c} where a, b, c corresponding to the amount of people in the first, second and third floor respectively.

Problem's world is represented as follows:

- Contains number of floors, current floor and positioning the elevator on the appropriate floor.
- Contains maximum number of people in elevator.
- Contains number of people in the elevator waiting for their destination (ground floor) and number of people waiting for the elevator on each floor. Each floor can be represented by an element in the list and the amount of people waiting is represented by the number in the element.
- Contains number of people evacuated on the ground floor.

Transition operators (actions):

- **pickUpPeople:** this operator is used to move the elevator between floors and to pick up people into the elevator. The elevator can move up and down, from any floor to any floor (but not to the ground floor) in condition that there is still place for people to board and there is at least one resident waiting to the floor. During picking up people in the elevator we have to update number of people in the elevator and number of people who wait for the elevator to come again.

- **dropOffPeople:** this operator is used to move the elevator to the ground floor and to release people there. This operator will only be executed if elevator is full or if elevator is not full, but there are no other waiting people on the floors. During this action we have to update number of people in the elevator (number of people will be 0).

Starting state:

- numberOfFloors = 4 (ground floor + 3 floors)
- currentFloor = 0
- numberOfPeopleInElevator = 0
- listOfPeople[3] = {2,6,4}
- const maxNumberOfPeopleInElevator = 5

Search algorithm and solution of the problem

The idea of a solution: During evacuations in buildings, people at the highest floors are at greatest risk. Because of this reason, the evacuation strategy in my solution is to save the people from the top floors first.

As we know it, the elevator can move between floors (except ground floor) whenever it is not full. To determine which floor the elevator should be positioned on, we need a function to help us with this problem.
The function will be defined so that it returns the value corresponding to the highest floor in the building where people are. Parameter of this function is list with 3 elements (numbers), where is each floor represented by a number corresponding to the amount of people waiting on that floor.
The function is defined to check from the top floor to the lower floor, the return value will correspond to the first floor where people are. This function will return a value from 1 to 3 corresponding to the highest floor with people in building (number 1 – first floor, number 2 – second floor, number 3 – third floor). If all people from building are evacuated, then return value will be -1.
Declaration of this function:
int highestFloorWithPeople(listOfPeople);

Next, we need a function that will move elevator between floors. This movement depends on the return value of the previos function. Parameter of this function is number corresponding to that return value.
Declaration of this function:
void moveElevator(highestFloorWithPeople(listOfPeople));

During the evacuation, we must take care of the number of people in elevator, because elevator capacity is 5 people. When elevator is full, it has to go down to the ground floor and to release the people. We can use previous function for this action, but value of parameter will be number 0 corresponds to ground level =>
moveElevator(0);

Pseudocode of this solution:

```
while(true) {
        highestFloor = highestFloorWithPeople(listOfPeople);
        if (numberOfPeopleInElevator != 5 && highestFloor != -1){
                moveElevator(highestFloor);
                currentFloor = highestFloor;
                if (numberOfPeopleInElevator + listOfPeople[currentFloor] <= 5){
                        numberOfPeople+=listOfPeople[currentFloor];
                        listOfPeople[currentFloor] = 0;
                }
                else{
                        listOfPeople[currentFloor]  –= 5 – numberOfPeople;
                        numberOfPeople = 5;
                }
        }
        else{
                moveElevator(0); // releasing the people on the ground floor
                currentFloor=0;
                numberOfEscapedPeople += numberOfPeople;
                numberOfPeople = 0; // number of people in elevator is 0 again
        }
        if (numberOfEscapedPeople == 12){
                break; // Goal State -> all people are escaped!!!
        }
}
```

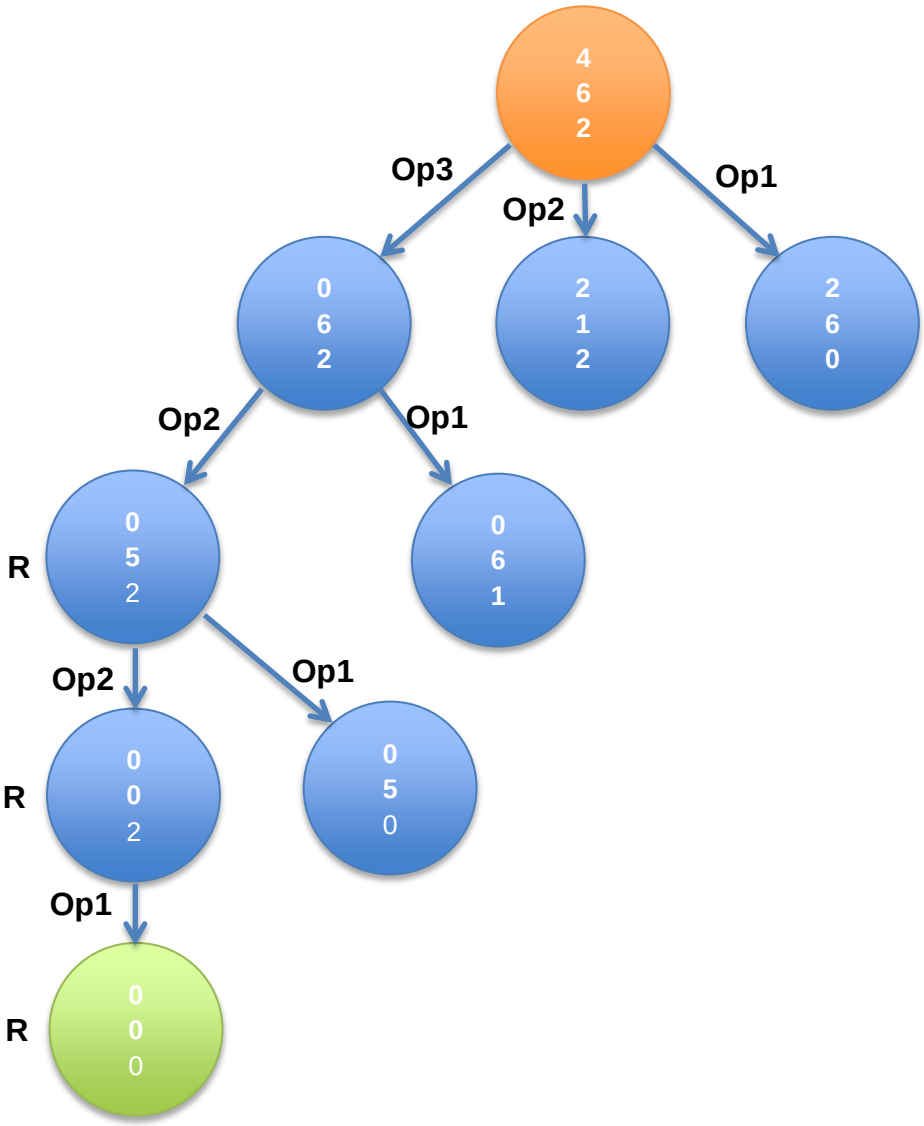
The Search algorithm that I chose is **DFS** (Depth First Search).

Search tree that the algorithm will produce:
Numbers inside circles correspond to number of people on each floor (the lowest number in the circle corresponds to the lowest floor).

Start state -> Goal state

Operations

Op1: Moving elevator from current floor to the first floor
Op2: Moving elevator from current floor to the second floor
Op3: Moving elevator from current floor to the third floor
R: Releasing people on the ground floor



The fridge and the queue that the algorithm will build in each step of search (DFS – Depth First Search) until to reach the goal:

1. {2, 6, 4}; queue: {2, 6, 4}
2. {2, 6, 0}; queue: {2, 6, 4}, {2, 6, 0}
3. {2, 5, 0}; queue: {2, 6, 4}, {2, 6, 0}, {2, 5, 0}
4. {2, 0, 0}; queue: {2, 6, 4}, {2, 6, 0}, {2, 5, 0}, {2, 0, 0}
5. {0, 0, 0}; queue: {2, 6, 4}, {2, 6, 0}, {2, 5, 0}, {2, 0, 0}, {0, 0, 0}

Solution:
On the fifth step, we achieved goal state! Solution for this problem with a series of states is: **{2, 6, 4}, {2, 6, 0}, {2, 5, 0}, {2, 0, 0}, {0, 0, 0}**