

Missionaries and Cannibals

END TERM REPORT

by

S. No.	Name	Reg. No.	Roll No.	Section
1	Nitish Kumar Yadav	11803484	02	K18PA
2	Naman Jain	11803487	03	K18PA
3	Nitesh kr. Sao	11803482	01	K18PA



Department of Intelligent Systems
School of Computer Science Engineering
Lovely Professional University, Jalandhar
04 - 2020

Student Declaration

This is to declare that this report has been written by us. No part of the report is copied from other sources. All information included from other sources have been duly acknowledged. We aver that if any part of the report is found to be copied, we are shall take full responsibility for it.

<<Nitish Kr. Yadav>>

<<Roll number: A02>>

<<Naman Jain>>

<<Roll number: A03>>

<<Nitesh Kr. Sao>>

<<Roll number: A01>>

Jalandhar

05- 04- 2020

(A typical specimen of table of contents)

Table of contents

Title	Page
1. Acknowledgement	4
2. Introduction	5
3. What is Missionaries and Cannibals	5
4. Process to solve the problem	6
5. Introduction to python	7
6. Processing Code	8
7. Output	10

1. Acknowledgement

We would like to express my special thanks of gratitude to my teacher Ms. Jasleen Kaur who gave me the golden opportunity to do this wonderful project on the topic Missionaries and Cannibals, which also helped me in doing a lot of Research and we came to know about so many new things I am really thankful to them. Secondly we would also like to thank my parents and friends who helped me a lot in finalizing this project within the limited time frame.

2. Introduction

In this project we are working on missionaries and cannibals Problem. Missionaries and cannibals Problem is a game in which we have to help the missionaries to cross the river with the help of boat. But there is some conditions that we have keep in our mind. Let's discuss in details.

3. What is missionaries and cannibals Problem?

In the missionaries and cannibals problem, three missionaries and three cannibals must cross a river using a boat which can carry at most two people, under the constraint that, for both banks, if there are missionaries present on the bank, they cannot be outnumbered by cannibals (if they were, the cannibals would eat the missionaries). The boat cannot cross the river by itself with no people on board. And, in some variations, one of the cannibals has only one arm and cannot row.

4. Process to solve the problem

First let us consider that both the missionaries (M) and cannibals(C) are on the same side of the river.

Right Left

Initially the positions are: 3M, 3C (B) and 0M, 0C

Now let's send 2 Cannibals to right of bank: 3M, 1C and 0M, 2C (B)

Send one cannibal from right to left: 3M, 2C (B) and 0M, 1C

Now send the 2 remaining Cannibals to right: 3M, 0C and 0M, 3C (B)

Send 1 cannibal to the left: 3M, 1C (B) and 0M, 2C

Now send 2 missionaries to the right: 1M, 1C and 2M, 2C (B)

Send 1 missionary and 1 cannibal to left: 2M, 2C (B) and 1M, 1C

Send 2 missionaries to right: 0M, 2C and 3M, 1C (B)

Send 1 cannibal to left: 0M, 3C (B) and 3M, 0C

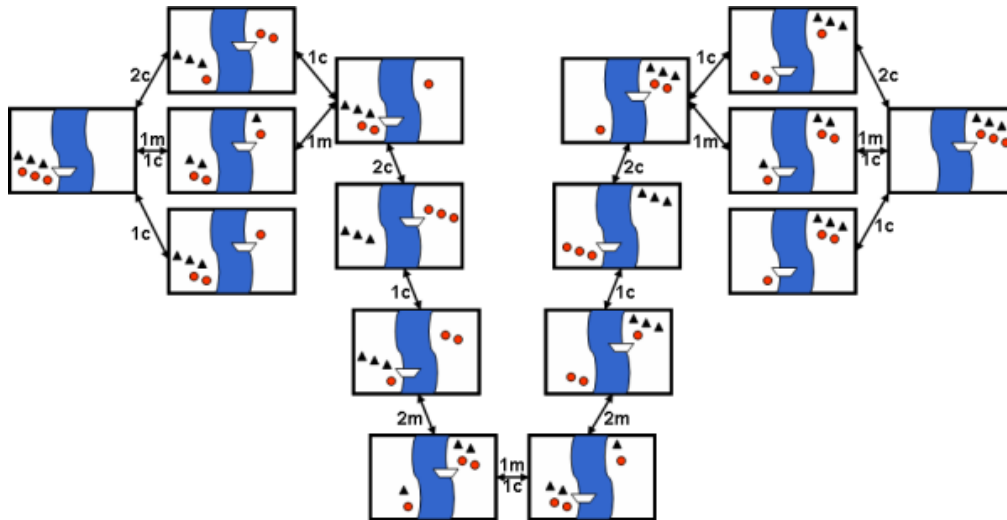
Send 2 cannibals to right: 0M, 1C and 3M, 2C (B)

Send 1 cannibal to left: 0M, 2C (B) and 3M, 1C

Send 2 cannibals to right: 0M, 0C and 3M, 3C (B)

Here (B) shows the position of the boat after the action is performed.

Therefore all the missionaries and cannibals have crossed the river safely.

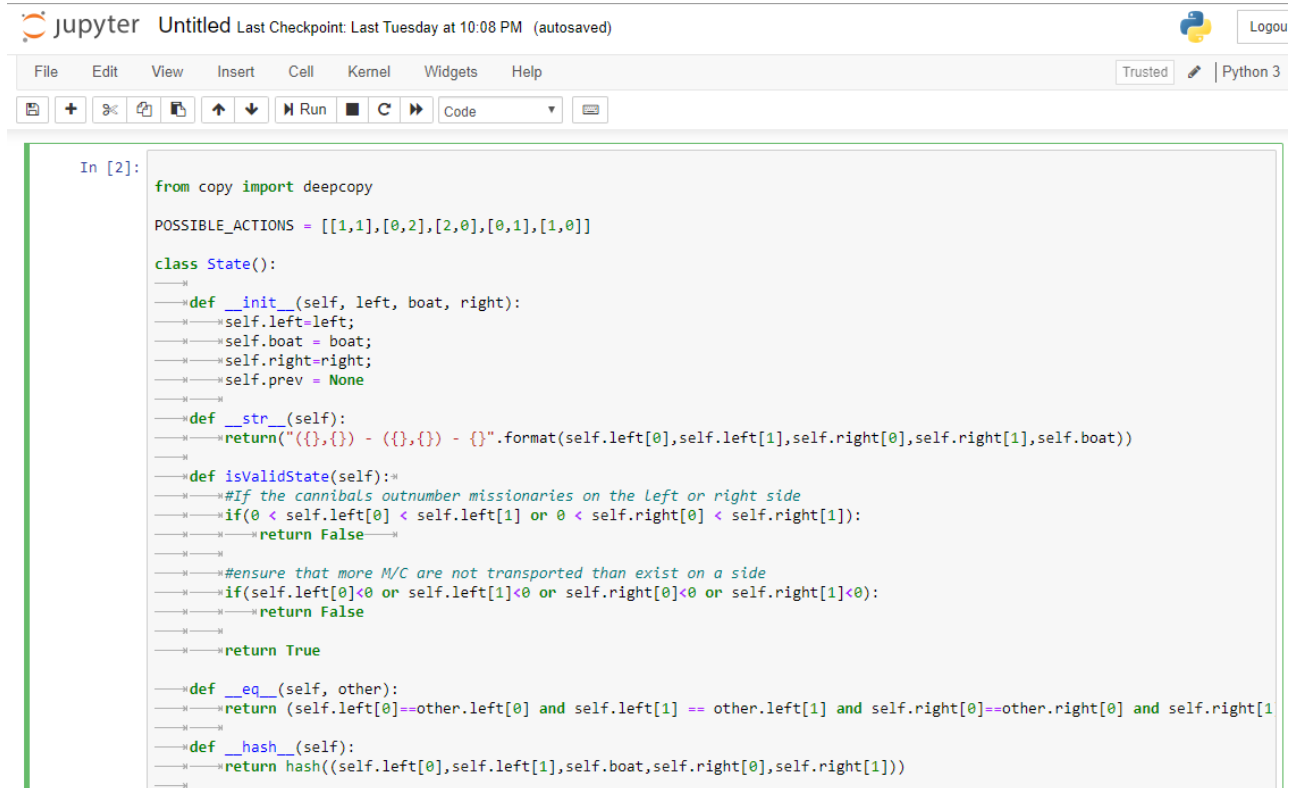


5. Introduction to Python

Python is a high level, dynamic programming language which is used for this thesis. Python3.4 version was used as it is a mature, versatile and robust programming language. It is an interpreted language which makes the testing and debugging extremely quickly as there is no compilation step. There are extensive open source libraries available for this version of python and a large community of users.

Python is simple yet powerful, interpreted and dynamic programming language, which is well known for its functionality of processing natural language data, i.e. spoken English using deepcopy.

6. Processing code



The image shows a Jupyter Notebook interface. The top bar includes the Jupyter logo, the text "Untitled", and a checkpoint status "Last Checkpoint: Last Tuesday at 10:08 PM (autosaved)". On the right, there is a "Logou" button and a Python 3 logo. Below the top bar is a menu bar with "File", "Edit", "View", "Insert", "Cell", "Kernel", "Widgets", and "Help". A "Trusted" status indicator and a "Python 3" label are also present. The main area contains a code cell labeled "In [2]:" with the following Python code:

```
from copy import deepcopy

POSSIBLE_ACTIONS = [[1,1],[0,2],[2,0],[0,1],[1,0]]

class State():
    def __init__(self, left, boat, right):
        self.left=left;
        self.boat = boat;
        self.right=right;
        self.prev = None

    def __str__(self):
        return "({},{}) - ({},{}) - {}".format(self.left[0],self.left[1],self.right[0],self.right[1],self.boat)

    def isValidState(self):
        #If the cannibals outnumber missionaries on the Left or right side
        if(0 < self.left[0] < self.left[1] or 0 < self.right[0] < self.right[1]):
            return False

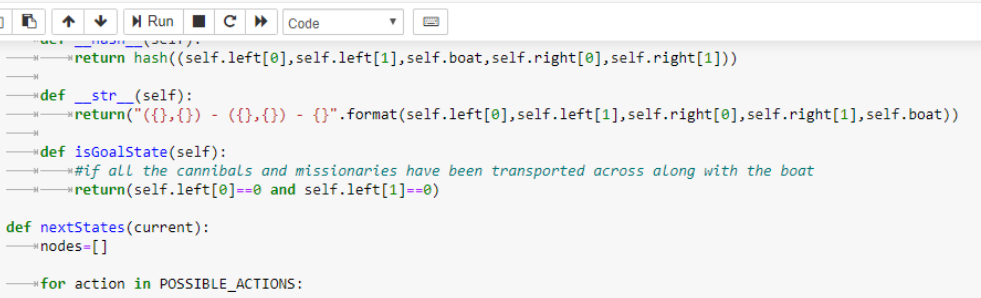
        #ensure that more M/C are not transported than exist on a side
        if(self.left[0]<0 or self.left[1]<0 or self.right[0]<0 or self.right[1]<0):
            return False

        return True

    def __eq__(self, other):
        return (self.left[0]==other.left[0] and self.left[1] == other.left[1] and self.right[0]==other.right[0] and self.right[1]

    def __hash__(self):
        return hash((self.left[0],self.left[1],self.boat,self.right[0],self.right[1]))
```


APPENDIX 9



```
def __hash__(self):
    return hash((self.left[0], self.left[1], self.boat, self.right[0], self.right[1]))

def __str__(self):
    return "({},{}) - ({},{}) - {}".format(self.left[0], self.left[1], self.right[0], self.right[1], self.boat)

def isGoalState(self):
    #if all the cannibals and missionaries have been transported across along with the boat
    return (self.left[0]==0 and self.left[1]==0)

def nextStates(current):
    nodes=[]

    for action in POSSIBLE_ACTIONS:
        nextState = deepcopy(current)
        nextState.prev=current

        #boat will be on the opposite side
        nextState.boat = 1-current.boat

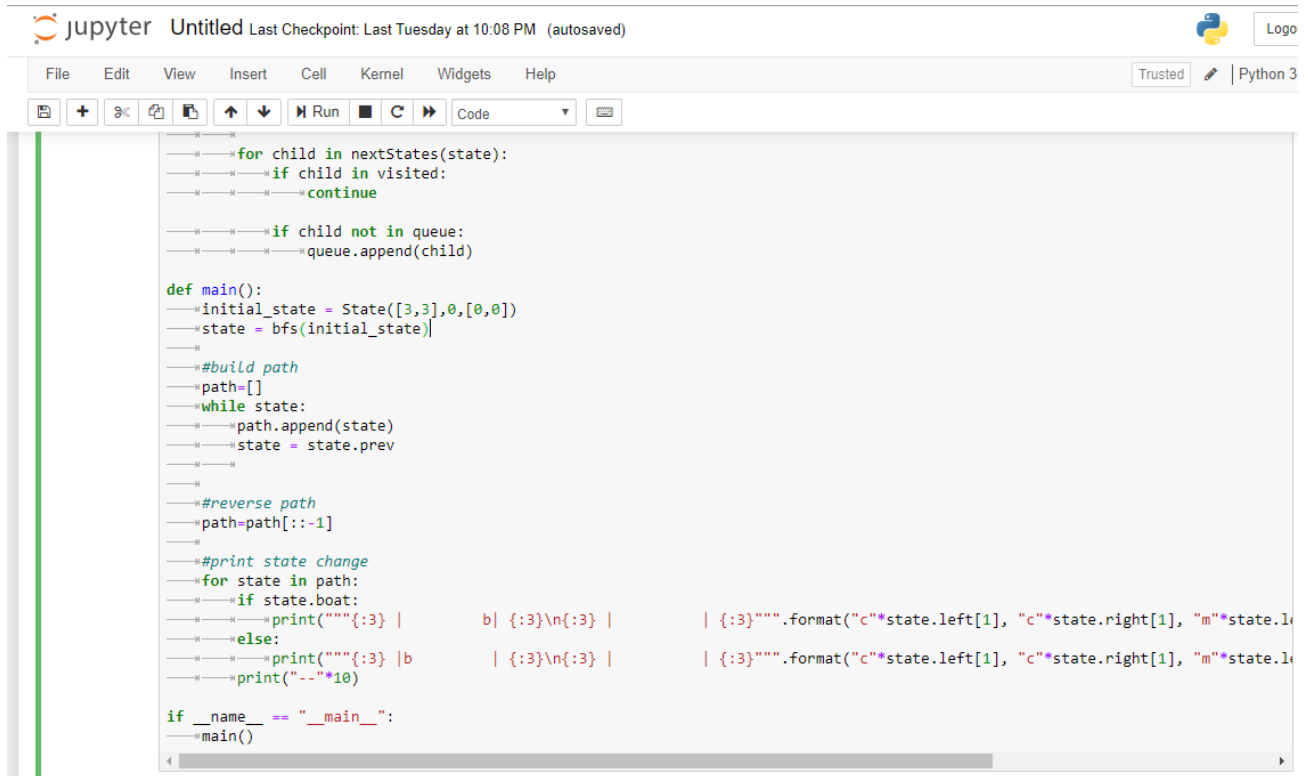
        #Moving from left to right
        if (current.boat==0):

            #Right population increases
            nextState.right[0]+=action[0]
            nextState.right[1]+=action[1]

            #Left population decreases
            nextState.left[0]-=action[0]
            nextState.left[1]-=action[1]

        #Moving from right to left
        elif (current.boat==1):
```

APPENDIX 10



The image shows a Jupyter Notebook interface with a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for file operations, running, and code execution. The notebook is titled "Untitled" and shows the last checkpoint from Tuesday at 10:08 PM. The code is written in Python 3 and implements a search algorithm, likely for a 3x3 puzzle. It includes a `nextStates` function, a `main` function, and a `__main__` guard. The code uses a `State` class and a `bfs` function. The output of the code is partially visible, showing a sequence of states and a path.

```

    for child in nextStates(state):
        if child in visited:
            continue

        if child not in queue:
            queue.append(child)

def main():
    initial_state = State([3,3],0,[0,0])
    state = bfs(initial_state)

    #build path
    path=[]
    while state:
        path.append(state)
        state = state.prev

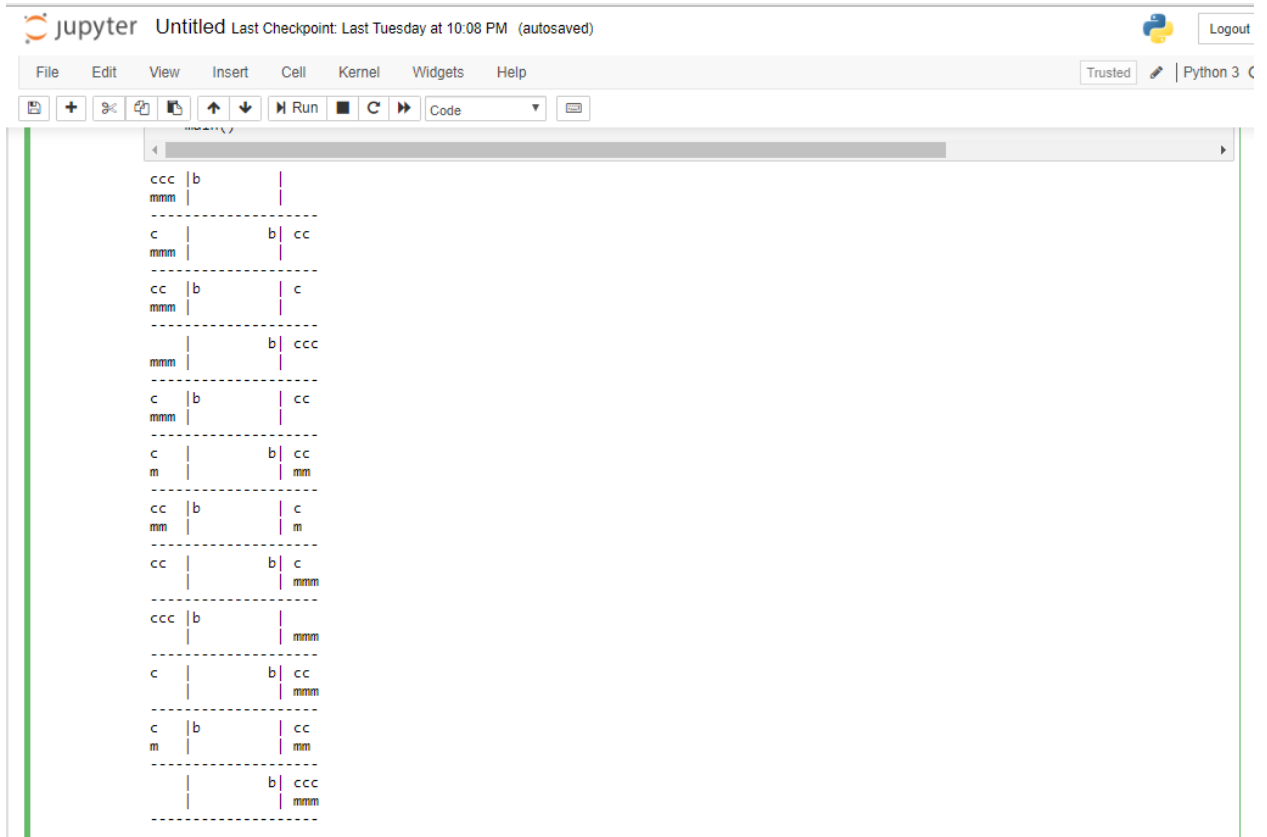
    #reverse path
    path=path[::-1]

    #print state change
    for state in path:
        if state.boat:
            print("{} | {} | {} | {}".format("c"*state.left[1], "c"*state.right[1], "m"*state.boat, "w"*state.water))
        else:
            print("{} | {} | {} | {}".format("c"*state.left[1], "c"*state.right[1], "m"*state.boat, "w"*state.water))
        print("--"*10)

if __name__ == "__main__":
    main()

```

7. Output



```

jupyter Untitled Last Checkpoint: Last Tuesday at 10:08 PM (autosaved)
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3
Run Code

ccc | b |
mmm |
-----
c | b | cc
mmm |
-----
cc | b | c
mmm |
-----
| b | ccc
mmm |
-----
c | b | cc
mmm |
-----
c | b | cc
m | | mm
-----
cc | b | c
mm | | m
-----
cc | b | c
| | mmm
-----
ccc | b |
| | mmm
-----
c | b | cc
| | mmm
-----
c | b | cc
m | | mm
-----
| b | ccc
| | mmm
-----

```

BONAFIDE CERTIFICATE

Certified that this project report “Missionaries and Cannibals” is the bonafide work of “Nitish Yadav, Naman Jain and Nitesh sao” who carried out the project work under my supervision.

Signature of the Supervisor

Jasleen Kaur

Ass. Professor

25340

Intelligent system

