A black sign with white text

Description automatically generated

Artificial Intelligence Project Report(INT-404)

Project Title: System for Family-Tree

Group 9:

* SHADAB KHAN (11801321 Roll no-26)
* SUDARSHAN SUKLA (11801708 Roll no-27)
* RAVIKANT PARETA (11801723 Roll no-28)

Submitted to: Mr. Ishan Kumar

Table of Contents

Packages used in making Family Tree

* Python
* Json
* LogPy

Concepts

Algorithm

Working

Output

I have to make family tree so I decided to make my own family tree. This is my family tree diagram and also I make this project based on these tree diagram.

A picture containing road

Description automatically generated

Concepts

* **Knowledge Representation and Reasoning.**

**Knowledge representation and reasoning** (**KR²**, **KR&R**) is the field of AI where we represent information about the world in a form that can solve complex problems. Knowledge representation and reasoning is also used for findings from logic to automate various kinds of reasoning, such as the application of rules or the relations of sets and subsets. In this family tree I’m using LogPy that enables the expression of relation and then it search for values which satisfy the problem. I’m attaching the working of my family tree.

Algorithm

This algorithm is O(n \* (1 + generations)), and will work for any dataset. For Example data this is O(n).

It will run through all records in family tree and generate objects representing people which include links to parents, and links to children, and several more uninitialized fields and we will not tell everyone blood relation with everyone.

Go through all people and recursively find and store Sultan. If you call the person again, return the memorized record. For each person you can encounter the person (needing to calculate it), and can generate 2 more calls to each parent the first time you calculate it. This gives a total of O(n) work to initialize this data.

Go through all people and recursively generate a record of when they first added a generation. These records only need go to the maximum of when the person or their last ancestor. It is O(1) to calculate when you had 0 generations. Then for each recursive call to a child you need to do O(generations) work to merge that child's data in to yours. Each person gets called when you encounter them in the data structure, and can be called once from each parent for O(n) calls and total expense O(n \* (generations + 1)).

Go through all people and figure out how many generation an. This is again O(n \* (generations + 1)) if implemented with a linear scan.

The sum total of all of these operations is O(n \* (generations + 1)).

Working

We are making Family Tree to store close blood relation of ours and system will tell us relation between any two person. That’s why I decided to make my own family diagram.

A picture containing road

Description automatically generatedAbove given diagram is my family diagram where I represented my each family member.

My grandfather Sultan and my grandmother Masluma have three sons - Sarfaraj, Shamsad, and Ibraj. My father name is Sarfaraj and the wives of Sarfaraj, Shamsad, and Ibraj are Pravin, Najia, and Nusrat respectively. That means Sarfaraj is my father and Pravin is my mother, and I also have a brother. From above diagram we can clearly see that my father Sarfaraj and my mother Pravin have two children - Shadab and Sahil. My uncle Shamsad and my aunty Najia have five children - Akbar, Asgar, Ashraf, Noor, and Kamal. My uncle Ibraj and aunty Nusrat have one child - Ibrahim.

Based on these facts and above diagram, I am creating a program that can tell us the name of Akbar's grandfather or  Ibrahim's uncles are. Even though I have not explicitly specified anything about the grandparent or uncle relationships, logic programming can infer them.

These relationships are specified in a file called relationships.json I have attached this file in ZIP Folder. you can check this file in Project Folder.

The file looks like the following:

{

"father":

[

{"Sultan": "Sarfaraj"},

{"Sultan": "Shamsad"},

{"Sultan": "Ibraj"},

{"Sarfaraj": "Shadab"},

{"Sarfaraj": "Sahil"},

{"Shamsad": "Akbar"},

{"Shamsad": "Asgar"},

{"Shamsad": "Ashraf"},

{"Shamsad": "Noor"},

{"Shamsad": "Kamal"},

{"Ibraj": "Ibrahim"}

],

"mother":

[

{"Masluma": "Sarfaraj"},

{"Masluma": "Shamsad"},

{"Masluma": "Ibraj"},

{"Pravin": "Sahil"},

{"Pravin": "Shadab"},

{"Najia": "Asgar"},

{"Najia": "Ashraf"},

{"Najia": "Noor"},

{"Najia": "Kamal"},

{"Nusrat": "Ibrahim"}

]

}

It is a simple json file that specifies only the father and mother relationships w. Note that we haven't specified anything about husband and wife, grandparents, or uncles.

Then I create a new Python file with name of family.py and imported the following packages:

import json

from logpy import Relation, facts, run, conde, var, eq

Define a function to check if x is the parent of y. We will use the logic that if x is the parent of y, then x is either the father or the mother. We have already defined "father" and "mother" in our fact base:

# Check if 'x' is the parent of 'y'

def parent(x, y):

return conde([father(x, y)], [mother(x, y)])

Define a function to check if x is the grandparent of y. We will use the logic that if x is the grandparent of y, then the offspring of x will be the parent of y:

Copy

# Check if 'x' is the grandparent of 'y'

def grandparent(x, y):

temp = var()

return conde((parent(x, temp), parent(temp, y)))

Define a function to check if x is the sibling of y. We will use the logic that if x is the sibling of y, then x and y will have the same parents. Notice that there is a slight modification needed here because when we list out all the siblings of x, x will be listed as well because x satisfies these conditions. So when we print the output, we will have to remove x from the list. We will discuss this in the main function:

Copy

# Check for sibling relationship between 'a' and 'b'

def sibling(x, y):

temp = var()

return conde((parent(temp, x), parent(temp, y)))

Define a function to check if x is y's uncle. We will use the logic that if x is y's uncle, then x grandparents will be the same as y's parents. Notice that there is a slight modification needed here because when we list out all the uncles of x, x's father will be listed as well because x's father satisfies these conditions. So when we print the output, we will have to remove x's father from the list. We will discuss this in the main function:

# Check if x is y's uncle

def uncle(x, y):

temp = var()

return conde((father(temp, x), grandparent(temp, y)))

Define the main function and initialize the relations father and mother:

if \_\_name\_\_=='\_\_main\_\_':

father = Relation()

mother = Relation()

Load the data from the relationships.json file:

with open('relationships.json') as f:

d = json.loads(f.read())

Read the data and add them to our fact base:

for item in d['father']:

facts(father, (list(item.keys())[0], list(item.values())[0]))

for item in d['mother']:

facts(mother, (list(item.keys())[0], list(item.values())[0]))

Define the variable x:

x = var()

We are now ready to ask some questions and see if our solver can come up with the right answers. Let's ask who Sultan's children are:

# Sultan's children

name = 'Sultan'

output = run(0, x, father(name, x))

print("\nList of " + name + "'s children:")

for item in output:

print(item)

Who is Sarfaraj's mother?

# Sarfaraj's mother

name = 'Sarfaraj'

output = run(0, x, mother(x, name))[0]

print("\n" + name + "'s mother:\n" + output)

Who are Ibraj's parents?

# Ibraj's parents

name = 'Ibraj'

output = run(0, x, parent(x, name))

print("\nList of " + name + "'s parents:")

for item in output:

print(item)

Who are Akbar's grandparents?

# Akbar's grandparents

name = 'Akbar'

output = run(0, x, grandparent(x, name))

print("\nList of " + name + "'s grandparents:")

for item in output:

print(item)

Who are Masluma's grandchildren?

# Masluma's grandchildren

name = 'Masluma'

output = run(0, x, grandparent(name, x))

print("\nList of " + name + "'s grandchildren:")

for item in output:

print(item)

Who are Shamsad's siblings?

# Shamsad's siblings

name = 'Shamsad'

output = run(0, x, sibling(x, name))

siblings = [x for x in output if x != name]

print("\nList of " + name + "'s siblings:")

for item in siblings:

print(item)

Who are Asgar's uncles?

# Asgar's uncles

name = 'Asgar'

name\_father = run(0, x, father(x, name))[0]

output = run(0, x, uncle(x, name))

output = [x for x in output if x != name\_father]

print("\nList of " + name + "'s uncles:")

for item in output:

print(item)

List out all the spouses in the family:

# All spouses

a, b, c = var(), var(), var()

output = run(0, (a, b), (father, a, c), (mother, b, c))

print("\nList of all spouses:")

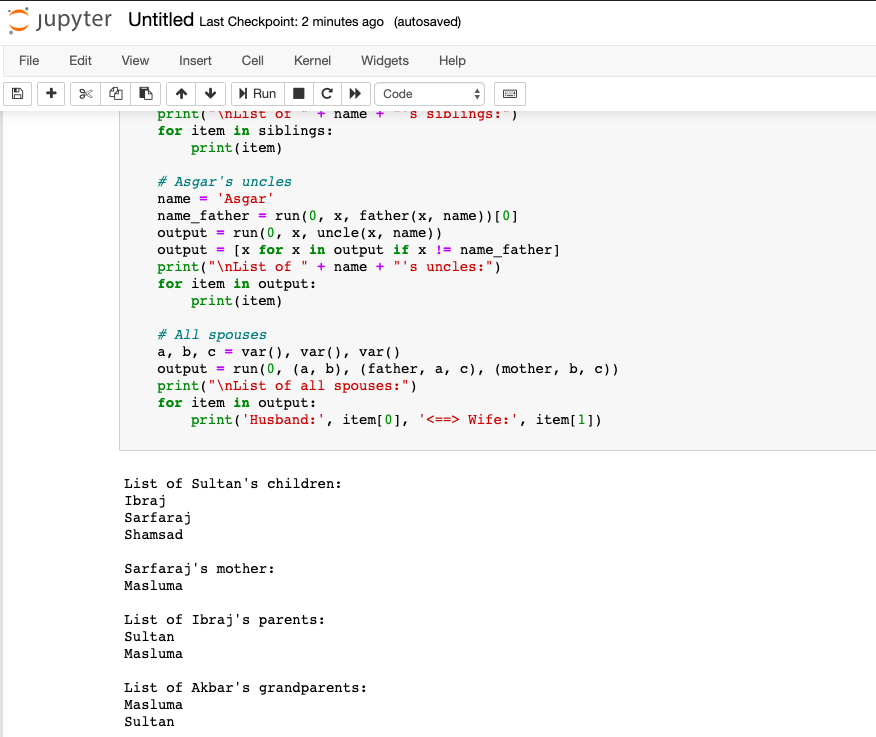
for item in output:

print('Husband:', item[0], '<==> Wife:', item[1])

The full code is given in Zip folder  family.py. If you run the code, you will see many things on Terminal or in PyCharm.

Output

The first half of output will looks like the following you can check by running the programme:



The second half looks like the following:

A screenshot of a cell phone

Description automatically generated

You can compare the outputs with the family tree to ensure that the answers are indeed correct. And I’m also attaching the Jupiter file for checking the output.