**Tents and Trees puzzle**

This is a project to implement a "Tents and Trees" puzzle in Python.

The Tents (also known as "Tents and Trees") is a popular logic puzzle.

According to https://wpcunofficial.miraheze.org/wiki/Tents, the puzzle was first published in 1989 by Léon Balmaekers under the name "Alen Verzamelen" in Dutch.

The rules of the puzzle are as follows:

* Find all of the tents in the forest.
* Every tent is attached to exactly one tree, and every tree is attached to exactly one tent.
* The number of tents is the same as the number of trees.
* The clues tell you how many tents are in that row or column.
* A tent can only be found horizontally or vertically next to a tree.
* Tents are never next to each other, neither vertically, horizontally, or diagonally.
* A tree might be next to two tents but is only connected to one.

**About Algorithm**

I solved this problem using a step-by-step, iterative method.

Each cells of grid may be 4 states : "tree", "tent", "grass", or "not-tested".

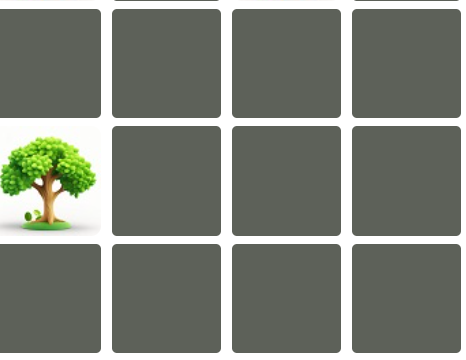
The steps of algorithm are as follows:

* Check all hints in rows and columns. If you find a row or column with a "0" hint, you can set all cells except for the trees in that row and column to "grass".
* And the rest of the steps are repeated until all tents are found.

1. Each identified "tree" and "tent" pair is excluded from the iteration. (Assume it is grass.)



1. Cells with no trees around them (top, bottom, left, right) are identified as "grass".



1. If there is only one 'not-tested' cell around the 'tree', this is the 'tent'.



1. All cells that exist around the cell identified as "tent" are set to "grass".
2. Check each hint and its rows and columns to determine the location and number of possible tents. Based on this information, make the identified cells into "tents".



2



2

1. Make the cells that exist around the new 'tent'(up, down, left, right) the 'grass'.

Also, if the current number of 'tents' in any row or column is equal to the hint, all remaining 'not-tested' cells become 'grass'.

1. See Hints and special location cases, cells that can't be 'tents' are set to 'grass'.

Like this:



1. check all the corners(meaning in subset) and "grass" the impossible ones.

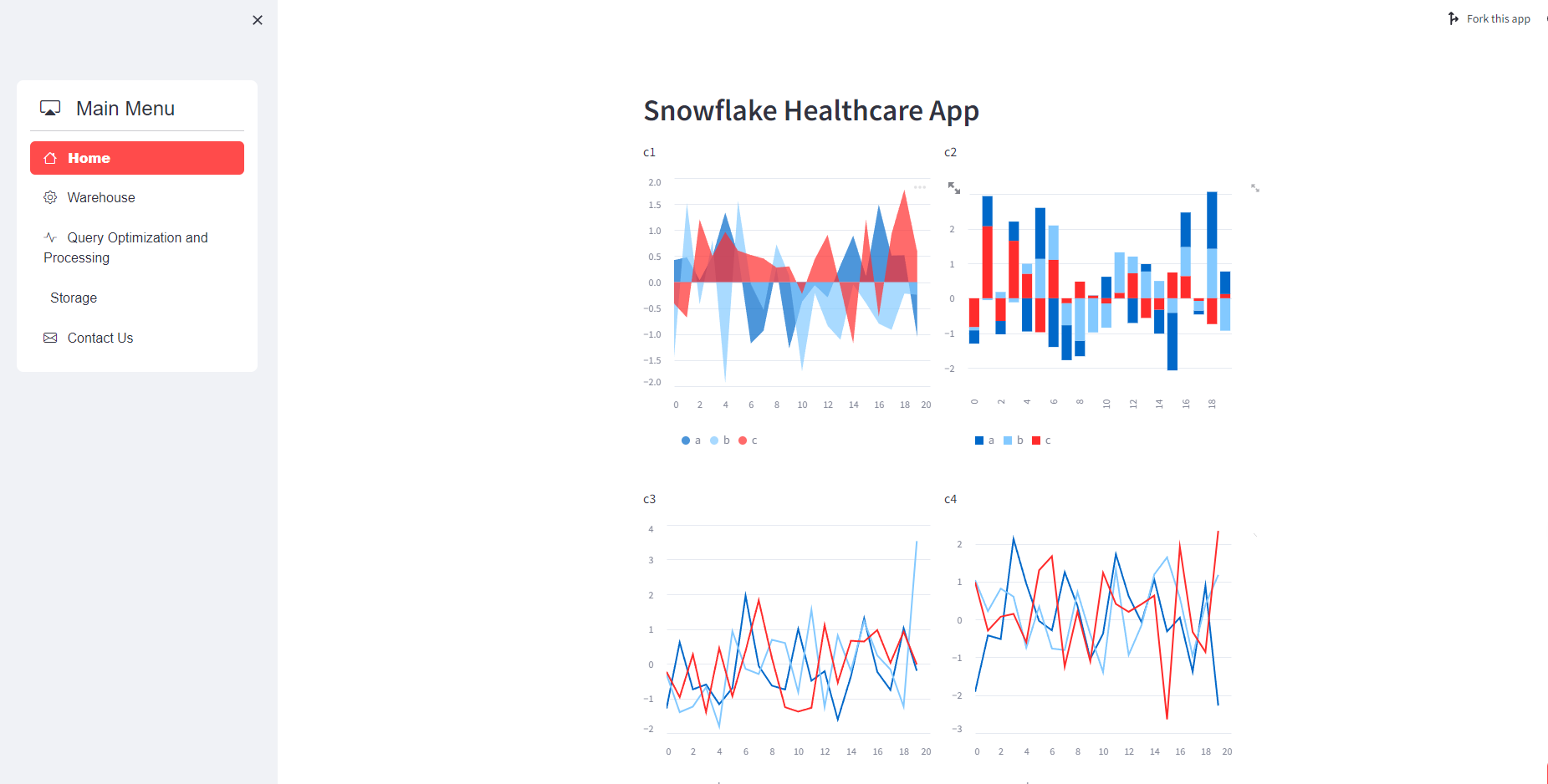


When you repeat this step, if the algorithm is satisfied with one step, it runs it again, starting from the first step. This is iterative method.

I don't think any one of breadth-first search, depth-first search, or others can solve this sensitive problem.

**User Interface**

And the UI is built with streamlit. Streamlit is a powerful tool to create web UI with python for data science and AI.



**Requirements**

This app can run on Python 3.10.10.

You need to install this version of Python to run this app.

Virtual environments are suitable for this app's installation package.

Follow this steps for python virtual environment installation:

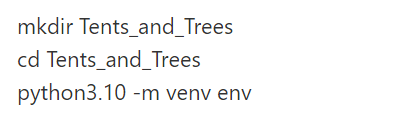
(This is allow for both VS code or Pycharm)

* Run this command in the terminal of VS code or Pycharm.



* To use venv in your project, in your terminal, create a new project folder, cd to the project folder in your terminal, and run the following command:

Like so:

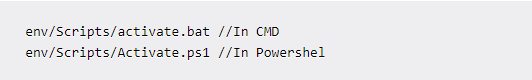


* Now that you have created the virtual environment, you will need to activate it before you can use it in your project. On a mac, to activate your virtual environment, run the code below:



This will activate your virtual environment. Immediately, you will notice that your terminal path includes env, signifying an activated virtual environment.

Note that to activate your virtual environment on Widows, you will need to run the following code below (See this link to fully understand the differences between platforms):



* And you can install packages on virtual environment.

Run this:



Alternatively, if you can create a requirements.txt file, write the following command in the text and run it to install all the requirements packages for this app:



* After package installation, you can run the app with this:



In my app, [streamlit app name] is Streamlit\_app.py.

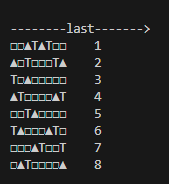
After that, you should see the app appear on your website.

**Structure of this app**

This app consists of four files : “Solver.py”, “main.py”, “Streamlit\_app.py” and “Images.py”.

* "Solver.py" contains all the functions in the algorithm that can perform all the steps in the algorithm.
* "main.py" processes the values from "solve.py" and outputs the results for step-by-step viewing in a terminal.

You can see this image:

H Where "▲" stands for "tent" and "T" stands for "tree", "\_" for "uninspected", and "□" for "grass".

(Since this is the last result, there is no '\_' to indicate 'untested').

* Also, "Streamlit\_app.py" is required for the web UI.