CAB203 Major Project
Complicated Wristwatch

Don Misura Minduwara Kaluarachchi

Introduction

My cousin has a complex digital wristwatch and he is having a bit trouble using it due to his tendency to favour old analogue watches. He explains to me that the wristwatch came with a user manual which was very unclear, but he understands that he can change the watch face by pressing the buttons **set**, **mode** or **long pressing set**. The watch faces are **Time, Time zone 2, Time set, Date set, Timer Set, Timer, Stopwatch, Alarm 1, Alarm 1 set, alarm 2, alarm 2 set.**

He has come to me in the hopes that I can help him navigate the wristwatch so he can go from the time mode to the date setting mode to set the date.

Instance Model

The problem of the complicated wristwatch can be modelled as a finite state automaton. It will contain an,

- Alphabet: The functions of the watch such as mode, set, long press set
- Set of states: The watch faces/ modes
- Starting state: Time
- The state change function: δ: S x $\sum$ -> S

Each state can be modelled as a vertex (u, v) e.g.: **(Time, Timer), (Timer, Stopwatch)**
Let W be the set of pairs of states of the watch. W can be given by

W = {(Time, TimeSet), (TimeSet, DateSet)}

The instance is modelled by (W, s, d) where s is the starting state and d is the end state.

Solution Model

In order to get from the starting state to the end state we can use the state change diagram that has been attached in Appendix A as a reference.
Using that diagram, we can see that once we can model the different modes/states as edges and the path will be drawing a path from the start to the end.
We can do this by doing a depth first traversal so that it goes from the “root” to the node the end. In this case it goes from the Time to the Date Set.

Problem Model

In an instance given by (W, s, d). We can model the watch states and transition as a graph G = (V, E).
The set of vertices is the states,

$$V = \{u : (u, v) \in W\} \cup \{v : (u, v) \in W\}$$

Meanwhile, E is the edge for each pair in W,

$$E = W \cup \{(v, u) : (u, v) \in W\}$$

The graph is connected according to the button presses. As mentioned by the cousin some states aren't affected by the button presses.
We can get from start to end by following the path in G. This will give us the shortest path across states

Solution Method

For the solution we will be using a depth first search to find the shortest path between start and end.
"Breadth first traversal tries to stay as close to u as possible, only moving further away when we've visited everything close by. Depth first traversal tries to go as far away from u as possible, and then backtracking when it can't go any further. Depth first search always moves between adjacent vertices, so it is good for applications like maze solving where you can't instantly move to another vertex elsewhere in the graph." – Lecture 9 Graphs

As explained above depth first search was used so that the best possible path was planned to get to the end state.

Let T be the path in the graph. This can be implemented later in python code. We start off with an initial vertex $V = V_0$.

Since we only want one path in the depth first search, we use $T = \{u\}$.
We then use a recursive function with the vertices, edges, u and the path T.

$$V_j = \frac{V_{j-1}}{D_{j-1}}$$

As we go through the vertices we have to keep changing the {u} to ensure we don't go to the same point again during the search to ensure shortest path. We have to also check if the path is completed and we have reach the vertex that we want to reach.

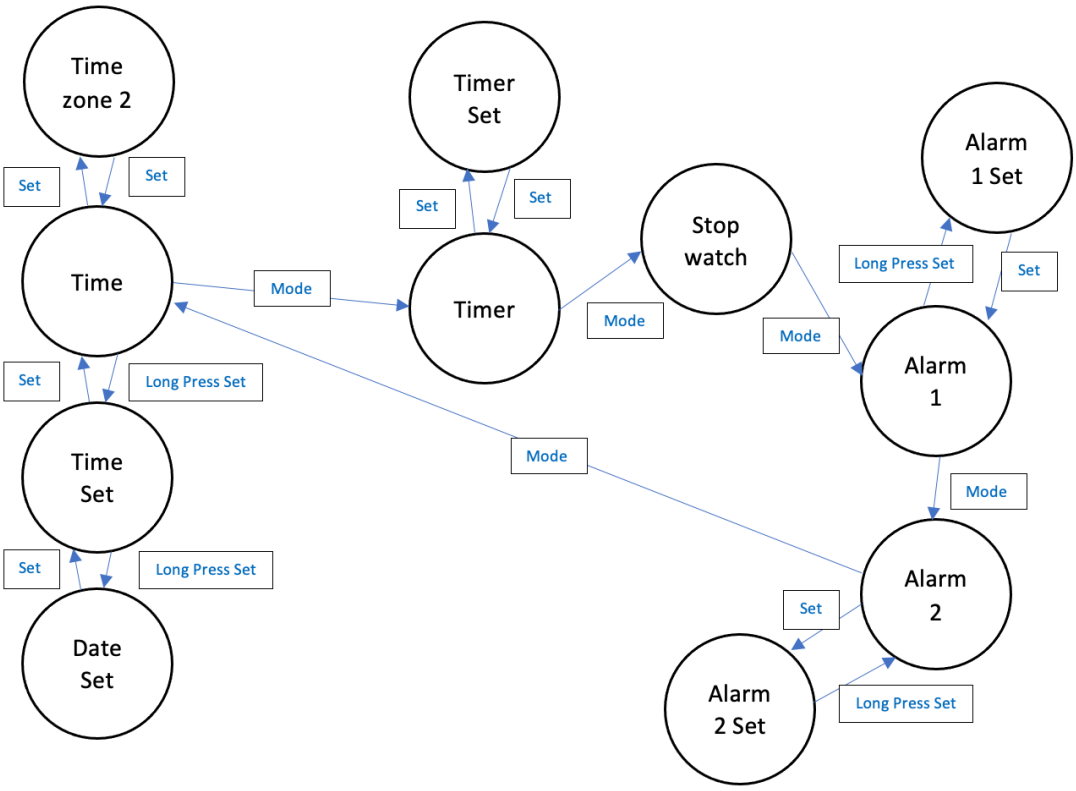
$$D_j = N_j(D_{j-1})$$

We then update the vertices in the branch. We now have the shortest path from the depth first search.
The path across states can then be used along with the vertices (u,v) to get the button presses required to get to those states.

References

1. Project Example: https://blackboard.qut.edu.au/bbcswebdav/pid-9270077-dt-content-rid-39011734_1/courses/CAB203_21se1/projectexample.pdf. Accessed 26th May 2021
2. Educative: Interactive Courses for Software Developers. 2021. *How to implement depth-first search in Python*. [online] Available at: <https://www.educative.io/edpresso/how-to-implement-depth-first-search-in-python> [Accessed 1 June 2021].
3. Cs.odu.edu. 2021. *Finite State Automata*. [online] Available at: <https://www.cs.odu.edu/~zeil/cs390/latest/Public/fsa/index.html#:~:text=%CE%B4%3AQ%C3%97%CE%A3%E2%86%92Q%20is%20the%20transition%20function.> [Accessed 4 June 2021].

Appendix A: State Change Diagram



Appendix B: Different test cases

Test Case 1: Time to DateSet

```
['Time', 'TimeSet', 'DateSet']
['LongPressSet', 'LongPressSet']
Dons-MacBook-Pro:CAB203-Code don$
```

Test Case 2: Alarm1Set to DateSet

```
['Alarm1Set', 'Alarm1', 'Alarm2', 'Time', 'TimeSet', 'DateSet']
['Set', 'Mode', 'Mode', 'LongPressSet', 'LongPressSet']
Dons-MacBook-Pro:CAB203-Code don$
```

Test Case 3: Timer to Alarm2Set

```
[Running] python -u "/Users/don/GitHub/CAB203-Code/almost.py"
()
['Timer', 'Stopwatch', 'Alarm1', 'Alarm2', 'Alarm2Set']
['Mode', 'Mode', 'Mode', 'Set']

[Done] exited with code=0 in 0.124 seconds
```