TV schedule planner

Martin Reinsalu

Idea and problem

Website kava.ee is probably one of the most popular sources to get information about TV series and shows. Quick access to a certain date and selecting a channel is made there rather convenient and the website itself is fast. However, the search option hasn't worked for a while. That means users can't find out when a certain show or serial was on TV. As television service providers offer rewatching option (mostly in the range of past two weeks), then in this case it would be an annoying activity to click through date by date just to search a show or serial.

The idea of this project and my solution for this problem was to use web scraping to get the previous data of two weeks from kava.ee website and implement an algorithm, which communicates with the user. This determines, which shows or series they want to watch, when by the latest they need to watch them and can they even watch these by considering the free time they have.

Brief introduction of the algorithm

The program is written in Python and it a local database is created by using sqlite3 module. After the data has been "crawled", a new table will be created (if not exist already) and the data of each show (unique by name and/or season and episode) inserted into the table grouped by three sections of channels (Estonian channels, series, sports). After every execution of the program it checks if some of the data is out of date (can't be rewatched anymore) and therefore those lines will be deleted.

Next the program's algorithm moves forward by asking the user step by step choices. First, they have to determine if they are interested in any channel of a certain section. If they aren't, then same about the next section will be asked. Otherwise, they will choose the channels by ID-s and then shows or series of each channel, also by entering the appropriate ID. Finally, the program asks to set the priority of choices and proposes options.

Measuring time of program's main parts

Obviously, the time of communication part between the program and user depends how fast the user is by entering their choices or ideally, when having a GUI, clicking the suitable options. But for curiosity I measured the time of the program ten times when only making requests and getting the data and then adding also database processing part.

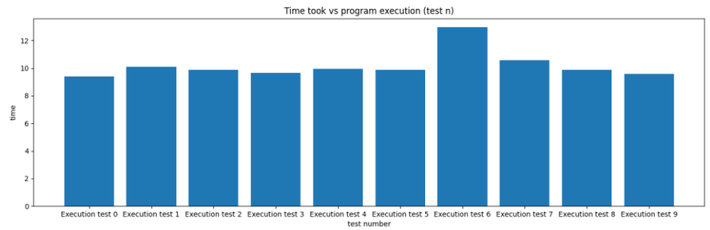

Figure 1.

The results in Figure 1 show that the average time of requests and database operations took 8-10 seconds. Although, one execution experiment was even 12-13 seconds. Number of lines added to database was between 9000 – 10 000 depending of the dates in past two weeks.

As expected, the measurements of only requests (crawls) were faster, seen in Figure 2. But there was a small anomaly of the first request, which took significantly more time, although there's more likely a chance of coincidence of the server response time.

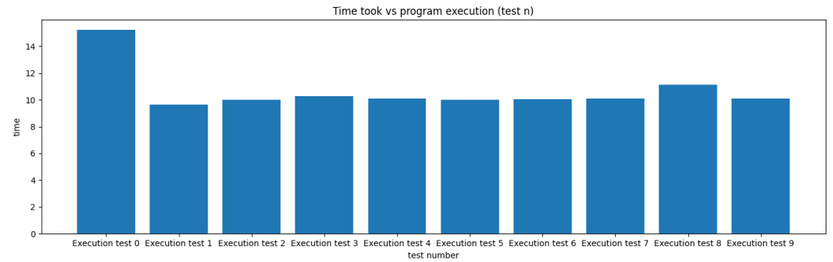

Figure 2.

Conclusion

Prioritizing the user's choices of this solution needs improvement, more testing overall should be done and also it would be nice that the user doesn't need to make that many steps. It's a question of further analyzing, especially the planner part. The goal is to use priority search tree, which would make the whole algorithm faster. Currently there are many loops, which iterate over a certain list more than once.

```
The shows/serials you are interested in are the following:
Kuldvillak (season 0, episode 256, date: 09-01-2022) [0]
Castle (season 4, episode 18, date: 09-01-2022) [1]
Football:Football: DFB Pokal 2021/22 - Hannover 96-Borussia Mönchengladbach (season 0, episode 17, date: 19-01-2022) [2]
Enter the integers followed by each selected show/serial in order of your priority (the lowest the more important):
2
1
0
Enter the free time in seconds:
25200

Considering the time you have, you should watch these shows/serials today:

Football:Football: DFB Pokal 2021/22 - Hannover 96-Borussia Mönchengladbach (season 0, episode 17, date: 19-01-2022)
Castle (season 4, episode 18, date: 09-01-2022)
Kuldvillak (season 0, episode 256, date: 09-01-2022)
```

Figure 3. Example output of the program's final part.

Link to GitHub: https://github.com/MartinUT/algorithmics_project