|  |  |
| --- | --- |
| **Complexity Analysis** | **25 total (each question is worth 5)** |

Code sample 1:

As this solution has one for loop, and only run one function iterating over the *num in numbers* the time complexity is O(N) which is linear time. And the space complexity is O (1) as the input value are integers which makes this constant.

Code sample 2:

As this code runs depth first it has a time complexity of O(N^m), the n is the branching factor in this case and m would be the depth that the algorithm goes to. For space complexity it would be O(N) where n would be the number of levels to the tree.

Code sample 3:

The time complexity for this would be O(n) as it is iterating through the string twice. The space complexity would be O(n) as it is dependent on the length of the string.

Code sample 4:

The time of this would be O(n/2) for time complexity which reduces to O(n) and the space is O(n/2) also which is O(n).

Algorithms 1:

The big O and time complexity of this are O(n) and O (1) respectively. This code is efficient as it takes the array and the sequence as arguments for the functions. It iterates over the sequence for the length of the sequence increasing the index of the search as it finds each match. If there are no matches the function returns false. If it iterates over the length of the subsequence, then the for loop breaks and returns true. I initially wrote this code with multiple for loops for the arrays and sequences, individually but the code in the .py file works.

Algorithms 2:

Initially I coded a solution to this question using two different types of sorting methods, the code is still in the .py file. The first one had a time complexity of O(n^2) and a space complexity of O(1). I then wrote a second answer using the python’s inbuilt method. sort, and this had a time complexity of O(n) and a time complexity of O(1) also.

My third attempt for this question checks to see if the value in the array is larger than the first index which enters it which is 0. After it does this it places it within [x,y,z] positions in a separate list once these three slots have been filled with the largest numbers across the array it returns them. The time complexity for this is O(n^2) and I am not sure of a more efficient way to execute this without sorting as done in my previous examples. The space complexity of this is O(1) as it is accepting integer values the same as the two other versions of my attempts to answer this question.

Concept generation and prototype design:

This music app is designed for athletes and people who are very active and like to listen to music during their workouts or exercise and tend to spend a lot of time picking music. This slows down their workout time and can interrupt workouts, looking for your phone to change the songs etc. This app is designed to be simple and have pre-curated playlists and easy access to skipping so that the user can use the app quickly and efficiently.

This app has 3 versions, mobile, desktop and on smart watch. Below are examples of the smartphone version and the smart watch version. These are the two I would anticipate to be used the most. The desktop version is purely for functionality, the core concept is that for whichever sport you are doing you can simply click a sport and a generated playlist would appear for you to listen to. The core of these playlists does require user input on the desktop app. Prompts such as “select 5 artists” or select a sport or “pick an athlete”, if there was expansion where athletes had playlists that the user could also select etc. The idea would be after user input, a playlist is then preloaded for the user and they can’t preview what has been curated for them. And after each use the playlists would refresh which keeps the app engaging and up to date with current music.

The white arrows on the smart watch icons indicate the direction for skipping songs or rewinding them. The play button would turn into a pause one upon clicking. The idea behind minimal buttons and functionality is that there is little to do and therefore the users can focus on their workout.

The colour palette for this is bright to reflect the activeness of the user demographic, as activewear usually tends to be quite bright and fluorescent sometimes. There are two ideas where either the exercises themselves have a colour palette or depending on the intensity of the workout it would increase in brightness. The concept is that you can also warm up and cool down and those have calmer colours like blue.

Future considerations would be to add further functionality on desktop where the user could have more control on the desktop version of the app to add key song they like in the playlists and share their playlists with others, as they are based on the users tastes initially. It would be interesting to expand into working playlists or for activities such as cooking where again no input is required it just generates your music for you based on predictions.

If it had voice recognition, for instance the user states the exercise, and it instantly begins playing their running playlist from the word run.

Storytelling through yearly sumups where the app also measures your exercise, or integration with other apps like strava to have then display a similar thing would be a fun add on. For each exercise animated versions of the equipment used in each space would be a way to make the page more interactive such as rolling yoga mats if that option was selected.

In term of navigation, I would like the app to snap on the user’s desktop wherever they like it to work with instant adaptation to the resolution of screens like macs application bar.

Graphical user interface

Description automatically generatedA picture containing text, outdoor, sign, vector graphics

Description automatically generated