My Healthcare Device: Technical Annex

Phase 1:

The myHealthcare device was created using a 2D array (matrix) structure and values were generated as listed in the specification. The index of each individual list corresponds to either the timestamp (index [0]) or a vital sign (Temperature [1], Heart rate [2], Pulse [3], Blood pressure [4], Respiratory rate [5], Oxygen saturation [6], pH [7]).

Phase 2:

The abnormalSignAnalytics function first asks the user for input. The options are either 'Pulse' or 'Blood Pressure' and once the user inputs their selection, the program will return the count of abnormal values, and the timestamp and abnormal value as a list of tuples. In addition, the program returns a histogram which represents this data accurately.

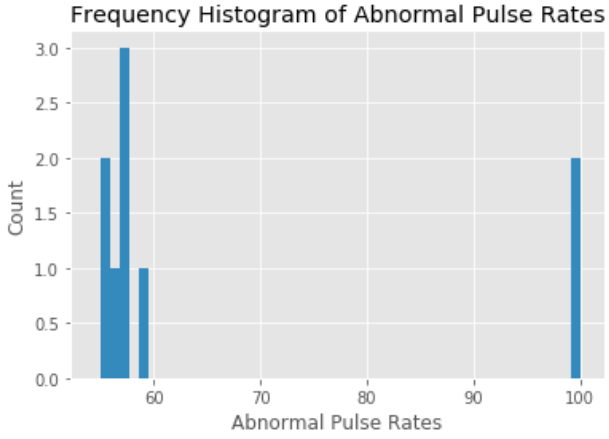
The histograms for 2a show the count of the abnormal pulse (Graph 1) or blood pressure (Graph 2) rates for a small sample of 50 records. The y-axis shows the count, and the x-axis the abnormal pulse or blood pressure value. For blood pressure, there is only ever 1 abnormal value, which is 121.

The computational complexity of the abnormalSignAnalytics function is linear O(N), as the program goes through the list of records (for either Pulse or Blood Pressure) and compares each record to the abnormal values individually.

The abnormalSignAnalyticsbin function, attempts to mimic binary search, however its nested-loop implementation results in polynomial $O(N^2)$ computational complexity. Hence, the abnormalSignAnalytics function, which is linear, is the better algorithm to use.

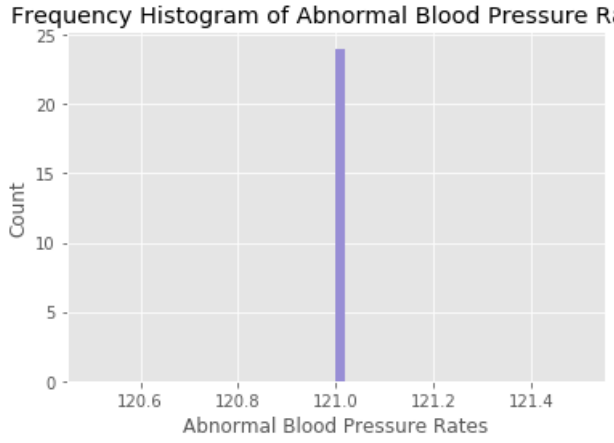
The frequency analytics function uses a dictionary data structure to count repeated values in a list. It then returns the count, and a histogram of the data. The computational complexity of the algorithm is O(N), as the program goes through a list of values, appends these to a dictionary, and then for every repeated value increments the count by 1.

The frequency histogram of pulse-rate values (Graph 3) shows the count on the y-axis, and the pulse-rate values on the x-axis for a small sample of 50 records. There is a good spread of pulse-rate values, which does not ever surpass a count of 3.



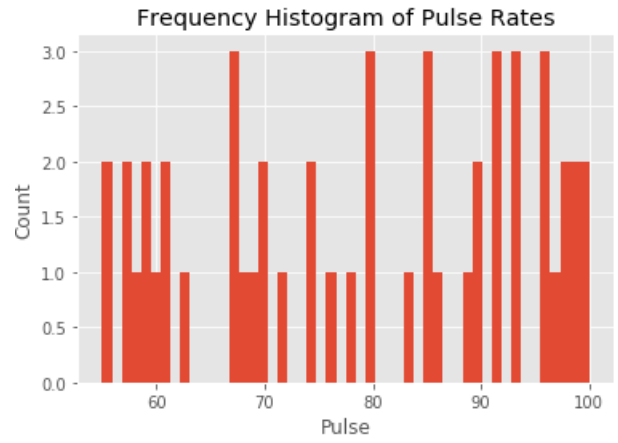
Graph 1:

Frequency histogram of abnormal pulse rate values for myHealthcare device.



Graph 2:

Frequency histogram of abnormal blood pressure rates for myHealthcare device.



Graph 3:

Frequency histogram of pulse rates for myHealthcare device.

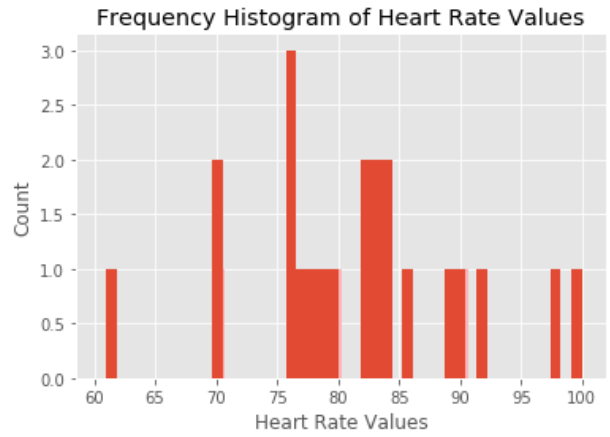
Phase 3:

The HealthAnalyser function searches for a pulse sign value in a list, and once found, appends the entire record to a new list. The function also returns a histogram with heartrate values (Graph 4) for pulse rate ‘56’ as mentioned in the specification. The computational complexity of the algorithm is O(N), as the query mechanism goes through each pulse value in each list individually.

There were several limitations with implementing binary search and interpolation search for the myHealthcare device. For example, the value at each index of the myHealthcare 2D array is linked to a particular vital sign. As the array needs to be sorted before binary search is implemented, this will re-order the data in each individual list, making it more difficult to identify which value corresponds to which vital sign. Moreover, as the values for pulse and heart rate could both potentially be 56, it is difficult to identify which vital sign corresponds to this value without maintaining the original indexed 2D array structure.

Furthermore, for both binary and interpolation search, the values of the 2D array could not be accessed without the use of a nested-loop. Therefore, the benefit of lower computational complexity of these methods is nullified as the program structurally can only access these

values with a nested-loop which is of polynomial complexity O(N^2). The best algorithm to use is the HealthAnalyser function as the computational complexity is lower at O(N). Moreover, the HealthAnalyser program can specifically access only pulse rate values of 56 (ignoring heart-rate values of 56) giving more accurate results.

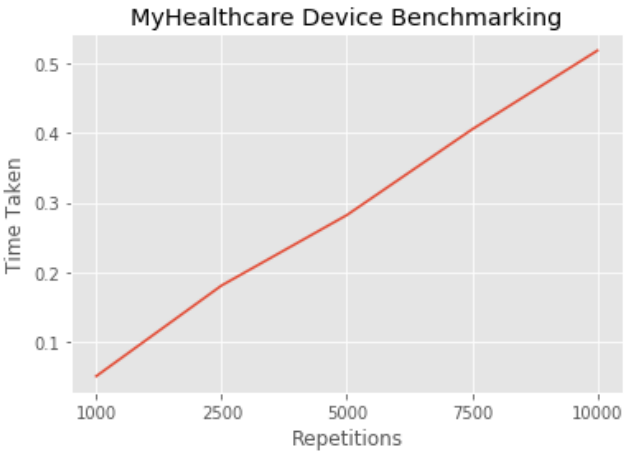


Graph 4:

Frequency histogram of heart rate values when pulse rate = 56 from myHealthcare device.

Phase 4:

The benchmarking function records how long it takes to run the myHealthcare function from phase 1. A loop is run for the values of n mentioned in the specification and a line-graph of the results is produced (Graph 5). Graph 5 shows that as the input value of n increases (repetitions) so does the time-taken in seconds for the program to run and output the generated records.



Graph 5:

Line-graph for myHealthcare device benchmarking, showing time-taken (seconds) on y-axis of n-repetitions on x-axis.