Analysis Report

1. The 5 different bin packing algorithms implemented are –
    a. Next Fit Algorithm.
    b. First Fit Algorithm.
    c. Best Fit Algorithm.
    d. First Fit Decreasing Algorithm.
    e. Best Fit Decreasing Algorithm.

Pseudocode summary –

- **Next Fit-**

```
Def nextFit(input []):
    Initialize binIndex to be zero and binSize to be 1
    Initialize an array of bins
    For i =0 to input.length
        If binSize – bins at current bin index is greater than or equal to the
        input
            Add the current element to the current binIndex in array of
            bins
        Else
            Increment the bin index i.e create a new bin
            Place the current element to the bin
Return binIndex+1 //The total no of bins
```

From the controller class which calls this function, calculate the wastage by subtracting the sum of input from total no. of bins used.

- **First Fit-**

```
Def firstFit(input []):
    Initialize binIndex to be zero and binSize to be 1
    Initialize an array of bins
```

```
        For i =0 to input.length
            Initialize the flag as false
                For j = 0 to the binIndex
                    If binSize – bins at current bin index is greater than or
                    equal to the input
                    Mark flag as true
                    Add the current element to the current binIndex in
                    array of bins
                    Break from loop
            If  flag is false
                Increment the bin index i.e create a new bin
                Place the current element to the bin
    Return binIndex+1 //The total no of bins
```

From the controller class which calls this function, calculate the wastage by subtracting the sum of input from total no. of bins used.

- **Best Fit-**

```
    Def bestFit(input []):
        Initialize binIndex to be zero and binSize to be 1
        Initialize bestIndex to be 0.
        Initialize an array of bins
        For i =0 to input.length
            Initialize minSpace to be binSize +1
                For j = 0 to the binIndex
                    Diff = binSize – bins at current bin index
                    If diff is greater than or equal to the input and diff is
                    less than minSpace

                    Mark bestIndex = j

                    Make minSpace = diff

            If  minSpace equal binSize+1

                Increment the bin index i.e create a new bin

                Place the current element to the bin

            Else

                Add the current element to the current binIndex in arryList of
                bins

                Update the minSpace by decrementing the current input



    Return binIndex+1 //The total no of bins
```

From the controller class which calls this function, calculate the wastage by subtracting the sum of input from total no. of bins used.

- **BFD** –
  - Sort the input array in decreasing order
  - Call the bestFit function above
- **FFD** –
  - Sort the input array in decreasing order
  - Call the firstFit function above

Design choices –

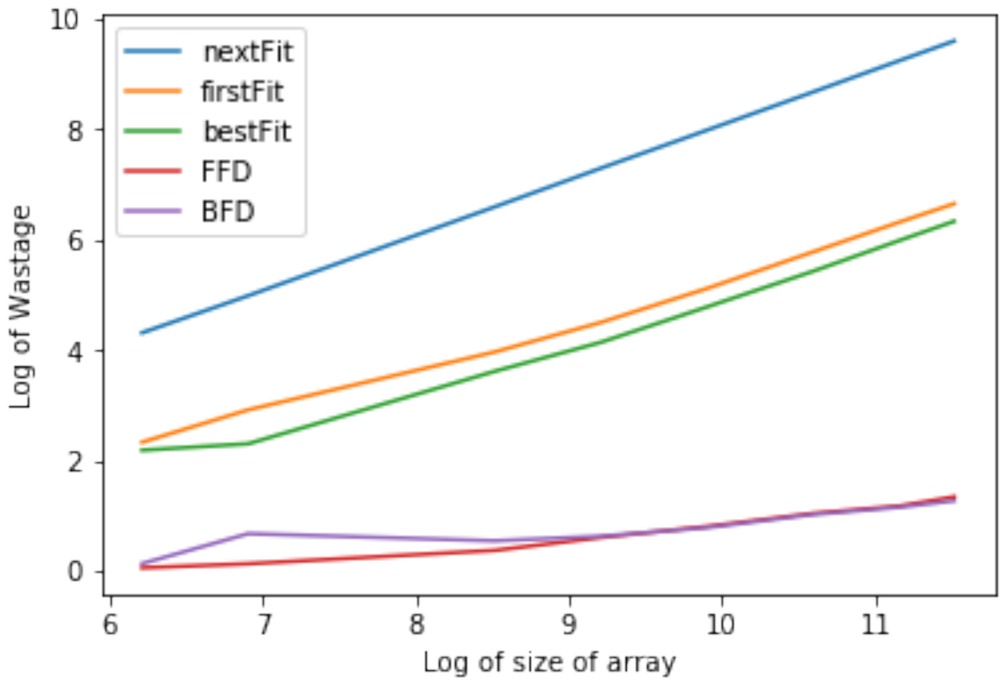
For all the algorithms described above, I have created an API which accepts the input array which is created by generating random numbers of different length and outputs the no of bin used to store them.

Comparative Empirical Analysis

1. **Random numbers**

   I have generated random numbers of different sizes starting from 500 to 100000 which acts as the input array.

For each array size I have considered 5 iterations and took the average of the wastage of bins. The visual plot for the 5 algorithms is –



The slope of the sequence are as follows –

```
1.0031242902707838
0.9469817966292692
0.9827904181865905
0.3251551566217007
0.2693447513851559
```

According to the slopes, the waste as a function of n for the above algorithms are –

a. Next Fit Algorithm – f(n) = n^1.00
b. First Fit Algorithm. – f(n) = n^0.94
c. Best Fit Algorithm. – f(n) = n^0.98
d. First Fit Decreasing Algorithm. – f(n) = n^0.32
e. Best Fit Decreasing Algorithm. – f(n) = n^0.26