Aurora Weiden
CS325 - Homework 8
08/16/19

<u>Pseudocode:</u>

First-fit/first-fit-decreasing:

```
first_fit(arr, C):
      bins = [0] #there will be at least one bin
      for each element in arr:
            index = 0
            for each bin:
                  if element+bin[index] <= bin[index]: #bin found
                        bin[index] = element+bin[index]
                        break #no need to check the other bins
                  index+=1
            if index == len(bins): #went through all the bins w/out
                                          finding a free one
                  bins.append(element) #add element to new bin
      return len(bins)
```

The runtime for first_fit is O(n^2). For first-fit-decreasing I will use python's sort function, which has a time complexity of O(n), so the time complexity for first-fit-decreasing is O(n^3)

Best-fit:

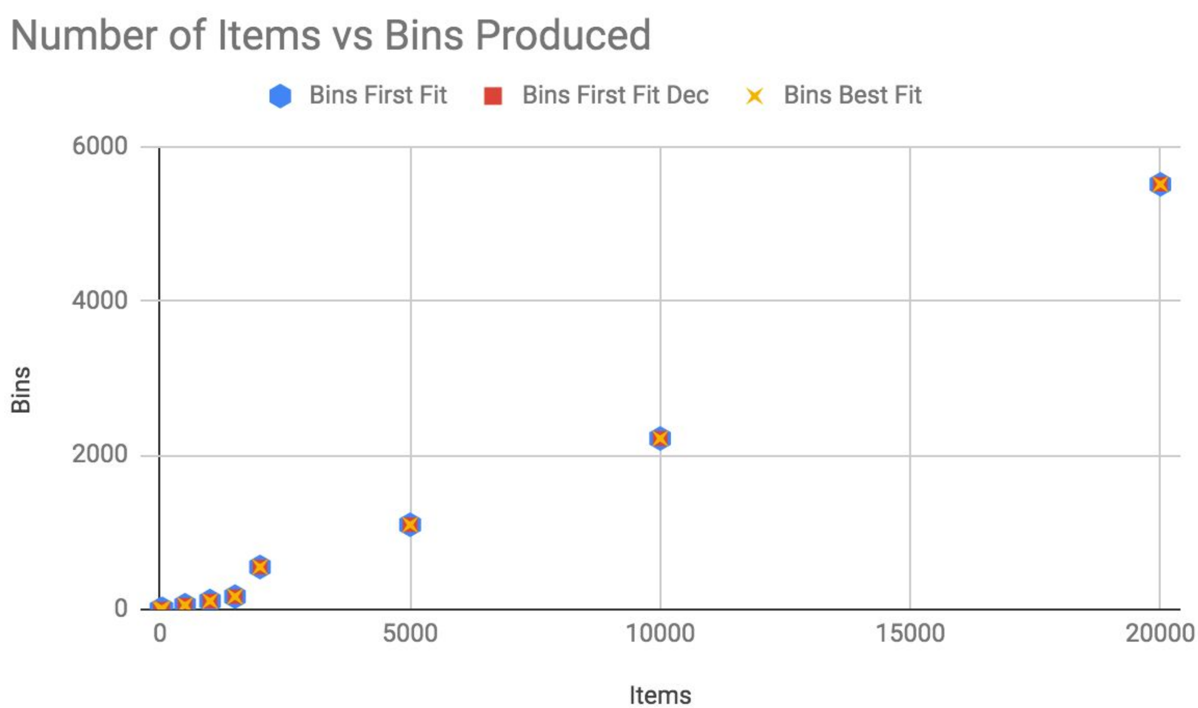
```
best_fit(arr, C):
      bins = [0] #there will be at least one bin
      for each element in arr:
            index = 0
            weights = [0]*len(arr)
            for each bin:
                  if element+bin[index] > bin[index]: #hit C
                        weights[index] = -1
                  else:
                        weights[index] = bin's weight
                  index+=1
            ind_max = index of max(weights)
            if ind_max == len(bins): #no free bin
                  bins.append(element) #add element to new bin
            else:
                  bins[ind_max] += element
      return len(bins)
```

The runtime for best_fit is O(n^2).

Aurora Weiden
CS325 - Homework 8
08/16/19

To generate 20 arrays of items of varying size I made a function that took in the size of an array and returned an array of that size, filled with random numbers between 1 and 10. I ran this function for arrays ranging from size 10 to size 20000 (so 10 to 20000 items). I then set the capacity to 50 and ran my three algorithms, timing each. Interestingly, I found that (i) each of the three algorithms performed equally as well finding bins for varying sizes (Figure 1), but (ii) the first-fit-decreasing algorithm was significantly slower than the other two algorithms (Figure 2). The best-fit algorithm was the fastest of the three, but only marginally faster than first-fit.

Figure 1



(Figure 2 on following page)

Aurora Weiden
CS325 - Homework 8
08/16/19

Figure 2