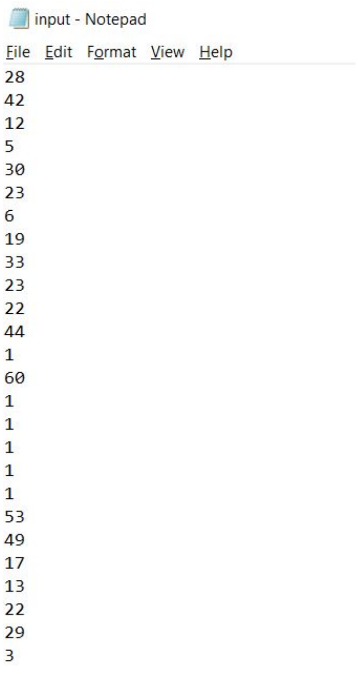
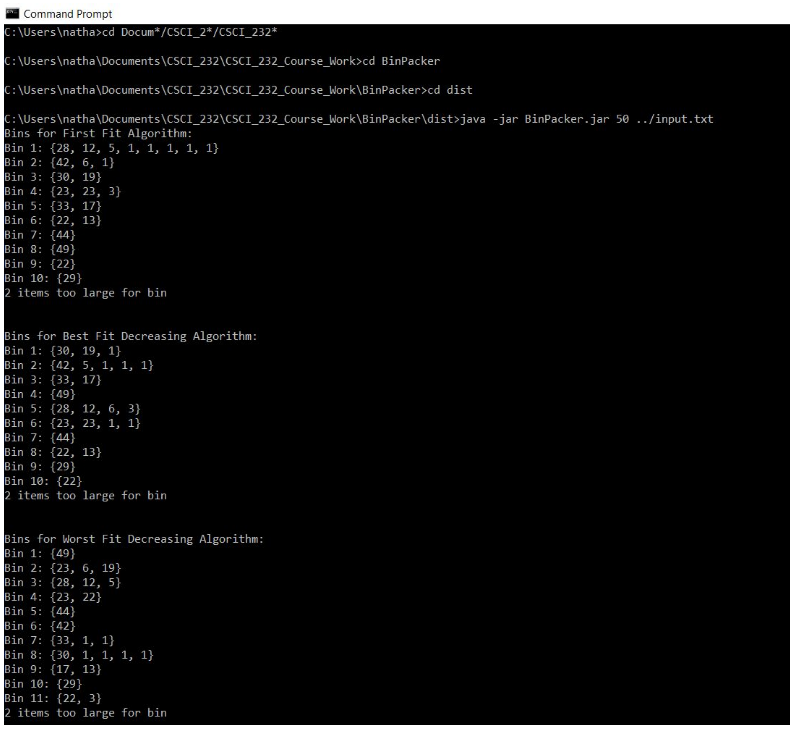
Nathan Stouffer
Lab 2 Response

In the different input tests that I performed, the best fit and first fit algorithms seem to use approximately the same number of bins while, on average, the worst fit algorithm seems to use a few more bins. Granted, my sample sizes did not get large by any means. One sample is shown below.



```
28
42
12
5
30
23
6
19
33
23
22
44
1
60
1
1
1
1
1
53
49
17
13
22
29
3
```



```
C:\Users\natha>cd Docum*/CSCI_2*/CSCI_232*

C:\Users\natha\Documents\CSCI_232\CSCI_232_Course_Work>cd BinPacker

C:\Users\natha\Documents\CSCI_232\CSCI_232_Course_Work\BinPacker>cd dist

C:\Users\natha\Documents\CSCI_232\CSCI_232_Course_Work\BinPacker\dist>java -jar BinPacker.jar 50 ../input.txt
Bins for First Fit Algorithm:
Bin 1: {28, 12, 5, 1, 1, 1, 1, 1}
Bin 2: {42, 6, 1}
Bin 3: {30, 19}
Bin 4: {23, 23, 3}
Bin 5: {33, 17}
Bin 6: {22, 13}
Bin 7: {44}
Bin 8: {49}
Bin 9: {22}
Bin 10: {29}
2 items too large for bin


Bins for Best Fit Decreasing Algorithm:
Bin 1: {30, 19, 1}
Bin 2: {42, 5, 1, 1, 1}
Bin 3: {33, 17}
Bin 4: {49}
Bin 5: {28, 12, 6, 3}
Bin 6: {23, 23, 1, 1}
Bin 7: {44}
Bin 8: {22, 13}
Bin 9: {29}
Bin 10: {22}
2 items too large for bin


Bins for Worst Fit Decreasing Algorithm:
Bin 1: {49}
Bin 2: {23, 6, 19}
Bin 3: {28, 12, 5}
Bin 4: {23, 22}
Bin 5: {44}
Bin 6: {42}
Bin 7: {33, 1, 1}
Bin 8: {30, 1, 1, 1, 1}
Bin 9: {17, 13}
Bin 10: {29}
Bin 11: {22, 3}
2 items too large for bin
```

Additionally, none of the algorithms will perform well with input that is sorted in ascending order. This will result in early bins failing to be fully filled and end up putting large items in bins by themselves with no chance of filling them in later. Sorting in descending order has the most chance of benefiting the best fit algorithm because the bins are attempted to be maximum filled and the only remaining inputs are guaranteed to be smaller than previous inputs. This means that the small holes left over might be filled by later inputs.