CS 325- Homework Assignment 8

Problem 1: (24 pts) In the bin packing problem, items of different weights (or sizes) must be packed into a finite number of bins each with the capacity C in a way that minimizes the number of bins used. The decision version of the bin packing problem (deciding if objects will fit into <= k bins) is NP-complete. There is no known polynomial time algorithm to solve the optimization version of the bin packing problem. In this homework you will be examining three greedy approximation algorithms to solve the bin packing problem.

- First-Fit: Put each item as you come to it into the first (earliest opened) bin into which it fits. If there is no available bin then open a new bin.

- First-Fit-Decreasing: First sort the items in decreasing order by size, then use First-Fit on the resulting list.

- Best Fit: Place the items in the order in which they arrive. Place the next item into the bin which will leave the least room left over after the item is placed in the bin. If it does not fit in any bin, start a new bin.

a) Give pseudo code and the running time for each of the approximation algorithms.

b) Implement the bin packing approximation algorithms in Python, C++ or C. Your program named binpack should read in a text file named bin.txt with multiple test cases as explained below and output to the terminal the number of bins each algorithm calculated for each test case. Your code should also collect and output the running time of each algorithm. Submit a README file and your program to TEACH.

Example bin.txt: The first line is the number of test cases, followed by the capacity of bins for that test case, the number of items and then the weight of each item. You can assume that the weight of an item does not exceed the capacity of a bin for that problem.

```
3
10
6
5 10 2 5 4 4
10
20
4 4 4 4 4 4 4 4 4 4 6 6 6 6 6 6 6 6 6 6
10
4
3 8 2 7
```

Sample output:

Test Case 1 First Fit: 4, time. First Fit Decreasing: 3, time. Best Fit: 4, time.
Test Case 2 First Fit: 15, time. First Fit Decreasing: 10, time. Best Fit: 15, time.
Test Case 2 First Fit: 3, time. First Fit Decreasing: 2, time. Best Fit: 2, time.

Note: time is the running time measured in the units of your choice.

c) Randomly generate at least 20 bin packing instances of varying sizes (number of items). Submit a description of how the inputs were generated not the code used to produce the random inputs.

i. Create a graph or chart that shows the number of bins used by each algorithm with instances of different sizes. Which algorithm performs better if the metric is number of bins?

ii. Create a graph or chart that shows the running time each algorithm as a function of numbers of items. Which algorithm performs better if the metric is running time?

Problem 2: *(6 pts)* An exact solution to the bin packing optimization problem can be found using 0-1 integer programming (IP) see the format on the Wikipedia page.

Write an integer program for each of the following instances of bin packing and solve with the software of your choice. Submit a copy of the code and interpret the results.

a) Six items S = { 4, 4, 4, 6, 6, 6} and bin capacity of 10

b) Five items S = { 20, 10, 15, 10, 5} and bin capacity of 20

Note: The version of LINDO that you have access to on the OSU server has a limit of 50 integer variables. Therefore, LINDO will only be able to solve problems with at most 6 items.