Hsiang Lo

Homework Assignment #8

Note

Items of different weights must be packed into a finite number of bins each with capacity C in a way to minimizes the number of bins used. Since the goal is to minimize the number of bins used, this is an optimization problem.

And since the decision version of the bin packing problem is a NP-complete problem, there is no known polynomial time algorithm to solve the optimization version of the bin packing problem.

Greedy approximation algorithms to solve this problem

1. 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
2. First-Fit-Decreasing- First sort the items in decreasing order by size, then use First-Fit on the resulting list

Part A)

First-Fit's running time is O(n^2)

```
Function firstFIt(capacity, weight)

    bin[] //Set a bin of array
    bin[0] = capacity

    for each items the array
        boolean doesFit = 0 //0 for false, 1 for true
        for every bins
            if the weight of item < space available
                add item to such bin
                bins space – weight of item
                set doesFit = 1 //true
                break
            else
                add a bin
                add such item to such bin
                bin space – weight of item
```

First-Fit-Decreasing's running time is O(n^2)

```
Function firstFitDecreasing(capacity, weight)
    mergeRev(weight, 0, weight size – 1)
    ifFindFit(capacity, weight)

function mergeRev(list, weight,newsize)
    if weight < newsize
        find midpoint
        call left recursively
        call right recursively
        mergeReverse(list, weight,  new size)

function mergeRevser(list, weight, new size)
    index = newSize – weigh + 1
    mid = newsize – weight

    A[index] //such that it is an array
```

```
        B[mid] //such that it is an array

        //merge everything back while compare value
        //return recombined array
```

c) Summarize your results on randomly generated sample input, which algorithm is better? How often?

While both algorithms runs and performs in O(n^2), it would appear that first fit decreasing function has a smaller delay when it comes to recursive calls for sorting the data. This is especially evident when increase in the number of sample data. So if minimization of time is the goal here, the First-Fit algorithm would be preferred. However, if space complexity is more important, First-Fit-Decreasing algorithm is better for that case as it takes less bins to fit all data when compared the two different algorithms.

First-Fit Running Time

N	Runtime	Bins
0	0.0001^-06	1
1000	0.002	521
5000	0.049	2592
10000	0.195	5121

First-Fit-Decreasing Running Time

N	Runtime	Bins
0	0.0001^-06	1
1000	0.002	499
5000	0.055	2540
10000	0.214	4520