COL 106: Assignment 3

Due Date : March 5, 2019

Implementation of a Bin Packing Algorithm

The bin packing problem can be described as follows -- there are n **bins** , say B[1], ..., B[n]. Each bin B[i] has a capacity cap[i], which is an integer. We are also given a set of **objects** , say O[1], ..., O[m]. Each object O[j] has a size s[j], which is an integer. The problem is to *pack* these objects into bins, i.e., we should assign each object to a unique bin such that the total sizes of the objects assigned to a bin does not exceed its capacity.

Let us look at an example. Suppose there are 3 bins with capacities 10,20,15. Suppose there are 5 objects with sizes 6,8,9,2,8. One way of packing the objects is to assign 2nd object to the first bin, 1st and 5th objects to the second bin, and 3rd and 4th objects to the third bin. Of course, there are other ways of doing this packing as well. In fact given an instance of this problem, it might happen that it is not possible to pack the objects into bins.

We will not worry about finding a good algorithm for bin packing. In fact, we will work with the well known **Best Fit Algorithm** for bin backing. The best fit algorithm for bin packing can be described as follows. When we need to add an object to one of the bins, we will pick the bin which has the largest **remaining capacity** . Let us go back to the example above. We will add the objects in the order 6,8,9,2,8. When we need to add the first object, the remaining capacities of the bins are 10,20,15 respectively. So it goes to the second bin. The remaining capacities are now 10,14,15. So the 2nd object goes to the third bin. The remaining capacities are now 10,14,7. The 3rd object goes to the 2nd bin. The remaining capacities are 10,5,7. The last 2 objects now go the the first bin.

In this assignment, you will develop an implementation of the best fit algorithm. For sake of implementation, assume that each bin has a unique id, which can be any integer. Similarly, each object has a unique id, which can be any integer. Of course, two bins can have the same capacity or two objects can have the same size.

The program should implement the following operations:

- **AddBin (id, c)** : In this method, we want to add a new bin to the system, whose id is given as a parameter, and c is it's capacity.
- **AddObject(id, s)** : Here, we want to add a new object to the system, whose id and size are parameters. You should use the best fit algorithm to assign the object to a bin. If the object can not fit into any bin, the program should print the object is too big, otherwise, it should print the id of the bin into which the object has been assigned.
- **DeleteObject (id)** : Delete the object whose id is given.
- **PrintBin(id)** : print the ids and sizes of objects in this bin.

If there are n bins in the system and m objects, the first three operations should take *O(log n + log m)* time, and the last operation should take *O(log n + S)* time where S is the number of objects in the particular bin. The total space used by the program should be *O(n + m)*.

The input to the program will be given in a file, where each line of the file will look like
option_no parameters

where the option_no is 1,2,3 4 depending on which if the above 4 methods are invoked. The java implementation should have the following class definition:

```
// Requirement : Use Java 8

class BestFit{
    public BestFit(){
            // Fill this
        }
    public void add_bin(int bin_id, int capacity){
            // Fill this
        }

    public int add_object(int obj_id, int size){
            // Fill this
            // return the bin id to which this object is added
       }

    public int delete_object(int obj_id){
            // Remove the object with id "id" from the bin it is placed in.
           // Fill this
```

```
            // Return the bin_id from which this object was removed
        }

     public List> contents(int bin_id){
    // Return the list of current objects in the bin with id "bin_id" : list of pairs  for each object in the bin.
    }
 }
```