Files distribution problem

Problem Description.

Let we have files with corresponding sizes (in MBs) . Our goal is to store these files onto disks, , that have corresponding storages amounts . One file can not be spread across multiple disks. In this problem, the goal is to minimize the amount of storage that is not used on each disk (that is used). This should also minimize the total number of disks being used.

First-fit algorithm.

This is a very straightforward greedy approximation algorithm. The algorithm processes the files in arbitrary order. For each file, it attempts to place the file in the first disk that can accommodate the file. If no disk is found, it opens a new disk and puts the file within the new disk.

It is rather simple to show this algorithm achieves an approximation factor of , that is, the number of disks used by this algorithm is no more than twice the optimal number of disks. In other words, it is impossible for disks to be at most half full because such a possibility implies that at some point, exactly one disk was at most half full and a new one was opened to accommodate a file of size at most . But since the first one has at least a space of , the algorithm will not open a new disk for any file whose size is at most . Only after the disk fills with more than or if a file with a size larger than arrives, the algorithm may open a new disk.

Pseudo-code and complexity.

```
sort All files descending by sizes
sort All disks ascending storage amounts

for All files i = 1, 2, . . . , n do
     for All disks j = 1, 2, . . . do
          if file i fits in disk j then
               Pack file i in disk j.
               Break the loop and pack the next file.
          end if
      end for
     if file i did not fit in any available disk then
```

```
            open new disk and pack file i.
        end if
    end for
```

In the worst-case, a new disk has to be opened each time a new file is inserted. Thus, there are executions of the inner loop, which yields an asymptotical factor of .

Brute force solution

The main purpose of the brute force is trying all possible solutions and pick the most appropriate. To check all possible solutions for the files distribution problem, a recursive approach can be used: Iterate over all disks, try to write the current file onto the disk and if it fits - call the same method with the next file.

Pseudo-code for the brute force method:

```
recurse(int fileId)
  if pastLastFile(fileId)
    if betterThanBestSolution
      bestSolution = currentAssignment
    return
  for each disk i:
    writeOnFile(fileId, i)
    recurse(fileId+1)
    removeFromFile(fileId, i)
```

The brute force approach complexity is , where is the number of files and is the number of disks. This is very slow irrespective of the implementation.