DeJuan Anderson May 9, 2014

6.046 Problem 5-4

Collaborators: *Jake Barnwell* Recitation: Kang Zhang, 12:00

Give an example that demonstrates that the greedy algorithm does not necessarily nd an optimal solution. That is, give a set of specific file sizes, show the solution found by the greedy algorithm, and then the optimal solution. Let L = 8. Let our file sizes be 3, 3, 4, and 5.

The greedy algorithm will place 3 and 3 on the first disk for a total space consumption of 6. It will try to place 4, notice that the first disk no longer has space to store 4, and will place 4 on the second disk. At this point, neither disk has enough space left to store a file of size 5, and so the greedy algorithm returns the following:

File sizes to be placed on Disk 1: 3, 3

File sizes to be placed on Disk 2: 4

Total number of files placeable: 3

However, if we put the two files of size 3 on different disks, we can fit all the files. Thus, the optimal solution would be:

File sizes to be placed on Disk 1: 3, 4

File sizes to be placed on Disk 2: 3, 5

Total number of files placeable: 4.

This is an example where the greedy algorithm does not find the optimal solution.

State the decision problem corresponding to this optimization problem. Then prove that the decision problem is NP-complete. The decision problem is as follows:

Does there exist a distribution of files between the two disks such that we store at least k files in total, if the individual disks have a storage limit of L?

1 Prove that the problem is in NP.

Given a certificate of this problem, we can simply check that there are at least k files on each disk and that the total capacity of each disk is not exceeded; simply check all of the files placed. There are N files, and if all files are placed, it would take us O(N) time to check the files and check the sum of their sizes. We could just update the sum whenever we check a file, since it would still be constant time to check file size and update the summation.

2 Prove that an existing NP-complete problem can be reduced to this problem in polynomial time.

In this instance, we will reduce from the Partition problem. The reduction is as follows. In the partition problem, the two sets are the two disks. We want each set to sum to the same file size as the other set. Our set S would simply be the set of file sizes. Let F be the total sum of all file sizes l. In this case, we want each partition to contain files whose sizes sum to F/2.

Note here a point which will be helpful in completing the proof: Partition is actually a special instance of our problem, where our L = F/2 and k = n.

The reduction is just making the set S the sizes we need to reduce, and setting the appropriate variables, which can be done in polynomial time in n, assuming we must actually construct the set S.

2.1 Show if "yes" to partition instance, then "yes" to Disk Storage Optimization

Henceforth, Disk Storage Optimization will be referred to as DSO.

This aspect of the reduction is fairly trivial. If we have a yes to the partition problem, where our set S was the set of file sizes, we have divided the total sum of file sizes equally between the two sets. By construction, since F is the total sum of the file sizes, for these partitions to be equal, they each have a selection of files whose sizes sum to F/2. We have also stored all files if the partition instance is "yes", so there are n files in storage.

If we set L = F/2 and k = n in our Disk Storage Optimization instance, the distribution obtained from our solution to the Partition problem is precisely the solution for our DSO instance.

2.2 Show if "yes" to DSO instance, then "yes" to partition

As noted earlier in this proof, the Partition problem is actually a special instance of our problem, where L = F/2 and k = n. If we set the parameters to those values and find a "yes" to this decision version of the problem, then the solution is once again precisely the solution to the Partition question given the same set S.

Prove that the greedy algorithm is an approximation algorithm that finds solutions that are at most 1 smaller than optimal. We want to prove the following:

$$OPT - 1 \leq ALG \leq OPT \tag{1}$$

Let us assume that for a given sequence L = $l_1, l_2, \ldots, l_n$, that when our greedy algorithm reaches some file l_i, i < n, that the file we wish to place next will not fit on disk one. Then we leave some amount of space, let us call it X_1. Then, later, at some file l_j, j $\leq$ n, we cannot place this file on disk 2, and we leave some amount of space X_2 on disk 2.

We claim that the OPT algorithm put put at most one file in the gaps X_1 and X_2

We will prove this claim by contradiction.

Assume that ALG $\leq$ OPT -1.

Then ALG is at most OPT - 2.

Then there exist two files, call them x and y, such that x + y < X_1 + X_2.

By the constraint on the file size list, we know that the list was given in non-decreasing order of file sizes. Further, since X_1 was the gap left on disk 1 when we found we could not fit file l_i, we know that the file l_i must have had size greater than X_1, and the same must be true for the gap X_2 and the file l_j. We can substitute these into the previous inequality relating x and y to X_1 and X_2:

x + y < l_i + l_j

Here lies our contradiction. Note that in our algorithm, we did not include the files X and Y. If these files existed, by the constraint of a non-decreasing order on the list of file sizes, we would have seen X and Y before seeing l_i and l_j. Since we did not include X and Y, these files must not exist, and so there is at most one file that can fit in the sum of the gaps created by our greedy placement.