UTAH STATE UNIVERSITY, COMPUTER SCIENCE DEPARTMENT

CS 7910 Computational Complexity Assignment 8

Gopal Menon

April 2, 2016

1. In this exercise, we consider the *knapsack packing* problem of minimizing the number of knapsacks to pack all items. The problem is formally defined as follows.

   We are given n items of sizes $a_1, a_2, ..., a_n$. We are provided with many knapsacks and the size of each knapsack is M, and it holds that $a_i \leq M$ for any $1 \leq i \leq n$. The goal is to use a minimum number of knapsacks to pack all items such that the total size of all items in the same knapsack is at most M and each item is packed in one and only one knapsack.

   The problem is equivalent to partitioning the numbers $a_1, a_2, ..., a_n$. into a minimum number of subsets such that the total sum of the numbers in each subset is at most M. As an application, consider the scenario when you are moving. You have many boxes of the same size (for example, you can buy them from Walmart) and you want to use a minimum number of these boxes to pack all your stuff.

   Consider the following polynomial-time algorithm to solve this problem. We consider the items in the order of $a_1, a_2, ..., a_n$. Starting with an empty knapsack, we keep packing the next item into the knapsack as long as the knapsack is not "overloaded". If it is overloaded, then we finish this knapsack and use another empty knapsack. We do this until the last item an is packed in a knapsack, and then we return all non-empty knapsacks as our solution.

   a) **(10 points)** Prove that the decision version of the knapsack packing problem is NP-Complete. (Hint: by a reduction from the Partition problem, i.e., the one in Assignment 5)

      In order to show that the decision version of the knapsack problem is NP-Complete, we can reduce it from the partition problem. Let (A) be an instance of the partition problem, where $A = a_1, a_2, ..., a_n$, and the *partition problem* is to decide whether A can be partitioned into two subsets A_1 and A_2 (i.e., $A = A_1 \cup A_2$ and $A_1 \cap A_2 = \phi$) such that $\sum A_1 = \sum A2$. Let (A, M, m) be an instance of the knapsack problem, where A is the set of n items as shown above, M is the size of each knapsack and m is the number of knapsacks. The decision knapsack problem is to decide whether given such an instance, it is possible divide the set A, into M subsets such that the sum of each subset is at most M.

Given a certificate of the knapsack problem, which will be the set A, the knapsack size M and the m sets, we can verify in polynomial time that the union of the m sets is the set A, the intersection of the m sets is the empty set and the sum of the items in each of the m sets is at most M. This means that the knapsack problem is in NP.

Consider a special case of the knapsack problem (A, M, m) where $M = \frac{1}{2}\sum A$ and $m = 2$. We can see the following:

i. We can clearly construct the special instance of the knapsack from from the partition problem in $O(n)$ time where $n = |A|$. So the construction can be done in polynomial time.

ii. Consider the case where the instance of the partition problem is true, where it is possible to partition the set A into two parts so that the sum of each part is equal. That means that we can pack all the n items into 2 knapsacks of size $M = \frac{1}{2}\sum A$. Consider the reverse case where we know that we can pack the items in set A into 2 knapsacks of size $M = \frac{1}{2}\sum A$. This must mean that the partition problem instance is true.

From i and ii above, we can conclude that the knapsack problem is in NP-Hard since it can be reduced in polynomial time from a known NP-Complete problem. Given that the knapsack problem is also in NP, we can conclude that the knapsack problem in in NP-Complete.

b) **(5 points)** Give an actual example where the above algorithm does not find an optimal solution for the knapsack packing problem.

c) **(15 points)** Prove that the approximation ratio of the above algorithm is 2.

d) **(10 points)** Give an actual example for which C is strictly larger than $\frac{3}{2} \cdot OPT$, where C is the number of knapsacks used by the above algorithm and OPT is the number of knapsacks used in an optimal solution.