CSCI 314 – Fall 2012 Handout: asgn10
Algorithms
Michael H. Goldwasser
Saint Louis University Sunday, 2 December 2012

Homework #10: Approximation Algorithms
Due Date: Monday, 10 December 2012

Guidelines

Please make sure you adhere to the policies on collaboration and academic honesty as outlined in the syllabus.

Reading

Chapter 35

Problems

For this homework, we will focus on a classic optimization problem known as *bin packing*. Informally, this is a problem that is dual to the knapsack problem, as there are a set of items that have various weights and there are bins (akin to knapsacks) that have a fixed maximum capacity. In the knapsack problem, you have a single knapsack and the goal was to pick an optimal subset of items that fit into the knapsack. In the bin packing problem, you must take *all* items, and the goal is to use as few bins as possible.

Formally, the input to the problem consists of n items with weights w_1, w_2, ..., w_n, and a fixed bin capacity C. A single bin can hold any collection of items that have combined weight of C or less. You may assume that $\forall i : w_i \leq C$, so that each item is guaranteed to fit in a bin. The goal is to assign items to bins so as to minimize the number of bins that are used.

Problem A (25 points)

"You may discuss ideas with other students."
The PARTITION problem is a special case of SUBSETSUM, in which you are given a set of integers $\{x_1, x_2, \ldots x_n\}$, and the goal is to determine whether there exists a subset of those numbers with sum precisely $t = \frac{1}{2}\sum_{i-1}^{n} x_i$. This problem is known to be NP-complete.

i) Show that the decision version of BINPACKING, in which you must determine whether a given collection of items can be stored in m bins of capacity C, is NP-complete. (Hint: Show that PARTITION $\leq_P$ BINPACKING.)

ii) Show that unless P=NP, there cannot exist a k-approximation algorithm for bin packing with $k < 1.5$. (Hint: Show that such a black-box approximation algorithm can be used to exactly solve the NP-complete `Partition` problem.)

Problem B (25 points)

"You may discuss ideas with other students."

We will designate the following algorithm as FirstFit. We consider an arbitrary numbers of bins, $B_1, B_2, \ldots$, all initially empty. We then consider the items in their original (arbitrary) order, placing each into the lowest indexed bin in which the item fits. That is, we try to put the item in B_1, otherwise into B_2, and so on, using a previously empty bin if the item did not fit into any of the earlier bins. The final solution consists of only those bins that are nonempty.

i) What approximation ratio is achieved by FirstFit on the following problem instance. There are 18 items, with the first six having weight $\frac{1}{7}+\epsilon$, the second six having weight $\frac{1}{3}+\epsilon$, and the final six having weight $\frac{1}{2}+\epsilon$?

For the remaining parts, consider general instances of the problem (not the specific instance from part i).

ii) Argue that FirstFit leaves at most one bin less than half full.

iii) Argue that the number of bins used by FirstFit is at most $\lceil \frac{2W}{C} \rceil$

iv) Conclude that FirstFit produces a 2-approximation for bin packing.

Problem C (25 points)

"You may discuss ideas with other students."

The first algorithm we consider is FIFO-Packing. We again consider placing the elements in their original (arbitrary) order, but using only one active bin at a time. We place the currently considered element into that bin, if it fits. Otherwise, we seal that bin, never again to place an element into it, and we place the current item into a new bin, which becomes the active bin.

i) Demonstrate that FIFO-Packing may leave an arbitrary number of bins strictly less than half full.

ii) Despite the observation of part (i), argue that FIFO-Packing uses at most $\lceil \frac{2W}{C} \rceil$ bins. (And therefore by similar reasoning as Problem B.iv, that FIFO-Packing produces a 2-approximation for bin packing.)

iii) Demonstrate that there are instances for which FIFO-Packing achieves an approximation ratio arbitrarily close to 2. Specifically, show that for any integer $m \geq 1$, there exists a bin packing instance for which FIFO-Packing uses $2(m-1)$ bins even though m bins suffice for an optimal solution.

Problem D (25 points)

"You may discuss ideas with other students."

The final algorithm we consider is known as FirstFitDecreasing (FFD). It begins by reordering the original items so that, without loss of generality, we can assume $w_1 \geq w_2 \geq \ldots \geq w_n$. Then it runs the standard FirstFit using that order, so that a decision as to where to place a bigger item is made prior to a decision about placing a smaller item.

In this problem, we lead you through a proof that if the optimal solution for an instance uses m bins, that FFD uses at most $\left\lceil \frac{4m-1}{3} \right\rceil$ bins.

i) Prove that if $\forall i : w_i > \frac{C}{3}$, then FFD packs the items optimally. To show this, we recommend that you consider the following subgroups of items, how they are grouped by FFD, and how they might be grouped by the optimal packing.

$$\begin{aligned} \mathcal{X} &= \left\{ w_i : w_i > \frac{2C}{3} \right\} \\ \mathcal{Y} &= \left\{ w_i : \frac{C}{2} < w_i \leq \frac{2C}{3} \right\} \\ \mathcal{Z} &= \left\{ w_i : \frac{C}{3} < w_i \leq \frac{C}{2} \right\} \end{aligned}$$

Next, we consider the more general case where items may have $w_i \leq \frac{C}{3}$. Of course, since FFD places items from largest to smallest, all of those with $w_i > \frac{C}{3}$ must be placed in one of the first m bins, where m is the size of the optimal solution.

ii) Show that if m is the size of the optimal solution, there can be a total of at most $m - 1$ items placed in bins other than the first m. Hint: Prove a contradiction if m items were placed in those other bins, based on the size of those items, the amount of space that can be remaining in the first m bins, and knowledge of the fact that an optimal solution using m bins assures that $\sum_{i=1}^{n} w_i \leq mC$.

iii) Based on (ii), argue that FFD uses at most $\lceil \frac{m-1}{3} \rceil$ bins beyond the initial m bins, and therefore a total of $\lceil \frac{4m-1}{3} \rceil$ bins.

iv) The quality of FFD's performance, when combined with the result of Problem A.ii earlier in this homework, might seem to imply that P = NP. Why isn't that the case?

Problem E **(EXTRA CREDIT – 10 points)**

"You may discuss ideas with other students."

The book presents a proof that the Greedy algorithm produces an (H_k)-approximation for the `SetCover` problem, where k is the maximum cardinality of any set $S \in \mathcal{F}$.

In the `WeightedSetCover` problem, each $S_i \in \mathcal{F}$ has a weight w_i that you must pay if you use S_i in a solution. The cost of a solution is the sum of the weights for those sets that are chosen in order to cover the universe of elements. The original `SetCover` problem is equivalent to having all weights equal to one.

Show that the greedy algorithm, and its analysis, can be generalized to provide an (H_k)-approximation.