Some notes, problems, and exercises on the class on knapsack and bin packing problems.

Problem 1 We discussed the strongly NP-hard 3-Partition problem in class. Given $3n$ items with sizes $a_1, a_2, \ldots, a_{3n}$ such that $\sum_i a_i = nB$, is there a partition of the items into n sets $S_1, S_2, \ldots, S_n$ of 3 items each such that for $1 \leq j \leq n$, $\sum_{i \in S_j} s_i = B$?

- Show that if we insist on exactly 2 items in each set instead of 3, the problem can be solved in polynomial time. *Hint:* Use matchings.
- Argue that we don't need to insist on exactly 3 items in each set.

Problem 2 *Multiple knapsack problem (MKP)*: we are given n items with item i having profit p_i and size s_i. Unlike the single knapsack problem we are given m different knapsacks with capacities $B_1, B_2, \ldots, B_m$. The goal remains the same: find a subset of items of maximum profit such that they can be packed into the given set of knapsacks.

- Consider the 2 knapsack case. Extend the pseudo-polynomial time algorithm for the single knapsack problem to obtain an exact algorithm with a running time of $O(nB_1B_2)$.
- Extend the above algorithm for the case of m knapsacks. What is the running time of your algorithm?
- For $m = 2$, can you get a pseudo-polynomial time algorithm with a running time that is polynomial in n and $P = \sum_i p_i$? Why not? *Hint:* Show that the Partition problem can be solved if such an algorithm exists.
- From above argue that there is no FPTAS for the 2 knapsack problem. Conclude that for existence of a pseudo-polynomial time algorithm does not imply existence of an FPTAS.

Problem 3 In class we discussed the greedy algorithm for the knapsack problem. Items are ordered such that $p_1/s_1 \geq p_2/s_2 \geq \ldots \geq p_n/s_n$. We put items in the knapsack in the above order, skipping items if they do not fit. Let k be the index of the first item that we skip over. Argue that $\sum_{i=1}^{k} p_k \geq \text{OPT}$. Since k does not fit into the knapsack, the fraction of k that can put into the knapsack is $\alpha = (B - \sum_{i=1}^{k-1} s_i)/s_k$. Argue that $\sum_{i=1}^{k-1} p_i + \alpha p_k \geq \text{OPT}$. From this argue that a 2 approximation follows if we pick either k or items 1 to $k-1$. What happens if you run greedy on the MKP problem with 2 knapsacks?

Problem 4 Here we develop a PTAS for the knapsack problem that finds applications in other problems. Consider the following algorithm. We guess the ℓ most profitable items that are picked in some optimum solution. Then we run the greedy algorithm on the remaining capacity with the remaining items. Argue that this gives a $\ell/(\ell-1)$ approximation. Develop a PTAS for the knapsack problem based on this idea. What is the running time as a function of n and $1/\epsilon$? Develop a PTAS for the 2 knapsack problem using these ideas. What happens if you have m knapsacks where m is a fixed constant? What is the running time of your scheme as a function of n, m and $1/\epsilon$?

Problem 5 We briefly talked about the *maximum coverage* problem. This problem is closely related to the set cover problem, they have the same decision version. In the set cover problem we are given m subsets of $\{1, 2, \ldots, n\}$ and the goal is to find the minimum number of sets that can cover all the elements. In the maximum coverage problem we are given an additional parameter k and the goal is to cover as many elements as possible while using at most k sets.

- The greedy algorithm for the set cover problem can be easily adapted for the maximum coverage problem: pick sets in the greedy order and stop when k sets are picked. Clearly this produces a feasible solution. Show that the greedy algorithm has an approximation ratio of $(1 - (1 - 1/k)^k)$ which tends to $(1 - 1/e)$ as $k \to \infty$.

- Suppose set i has cost c_i and we would like the total cost of sets picked to be at most some budget B. This is adding a knapsack like constraint to maximum coverage. The greedy algorithm can be modified where the goodness of a set is defined to be the ratio of its cost to the number of elements covered. Consider the following algorithm. Guess the ℓ most important sets from some optimum solution, then run the greedy algorithm while the budget is not exceeded. Do you see the similarity with the algorithm for the knapsack problem? What can you say about the approximation ratio of this algorithm? Argue that for $\ell \geq 3$ that the approximation ratio is $(1 - 1/e)$.

Problem 6 We mentioned greedy online algorithms for bin packing like First Fit (FF), Best Fit (BF), Worst Fit (WF) and so on. The FF algorithm is the following: put the next item in the first bin (in the order of opening) that accommodates it. If no bin accommodates the item we open a new bin for it. These algorithms work better in the offline setting if the items are sorted. First Fit decreasing (FFd) is the the algorithm that sorts items in decreasing order of size and executes the First Fit algorithm on this order. Show that FFd opens at most $3\text{OPT}/2 + 1$ bins. *Hint:* Among bins that are less than $2/3$ full look at the first such to be opened. Can you improve the bound? $11/9$ is the known bound.

Problem 7 We quickly covered an asymptotic PTAS for bin packing. Here are details to work out the result for yourself. For simplicity we shall assume that $1/\epsilon$ is an integer, the assumption can be removed easily. We started with two special cases.

- Small items. Suppose all items have size at most $\delta < 1$. Argue that any greedy algorithm that opens a bin only if necessary uses at most $(1 + \delta)\text{OPT} + 1$ bins. Hence if δ is small we can obtain a good approximation.

- Few distinct sizes. Let k be the number of *distinct* sizes in the item set. Give an $O(mn^k)$ algorithm to decide if m bins suffice to pack all the items. Conclude that the optimum solution for the instance can be found in $O(mn^{k+1})$ time. Hence if k is small we are doing well.

We will make use of the above two special cases to obtain a PTAS for the general case. Define an item to be *small* if its size is less than ϵ, otherwise it is *large*. Let n' be number of large items. Assume that $n' > 1/\epsilon^2$.

- Argue that $\text{OPT} \geq \epsilon n'$.

We sort the large items and partition them into k sets $G_1, G_2, \ldots, G_k$. The properties we want of these sets are the following.

- $|G_1| = |G_2| = \ldots |G_{k-1}| \geq |G_k|$.
- For $1 \leq j < k$, every item in G_j is at least as big as every item in G_{j+1}.

Figure out the following.

- What value of k should be chosen such that $|G_1| \leq \epsilon^2 n'$?
- How many bins are required to pack G_1?

- For $2 \leq j \leq k$ we round up the size of every item in G_j to the size of the *smallest* item in G_{j-1}. Argue that the items in $G_2, \ldots, G_k$, after the sizes have been rounded up, can be packed into OPT number of bins.
- How many *distinct* sizes are in the rounded up items?

Putting together. Given $\epsilon > 0$.

- Partition items into small and large using ϵ.
- Group items into k groups for sufficient large k.
- Pack items in G_1 into individual bins.
- Round up sizes of items in $G_2, \ldots, G_k$ and use the algorithm for few distinct sizes to find an optimal solution for them.
- Pack the small items greedily on top of the bins used for the large items and using extra bins if necessary.
- Argue that the total number of bins used is at most $(1+\epsilon)\text{OPT} + 1$.
- What is the running time for the algorithm?
- Can you handle the case when $n' \leq 1/\epsilon^2$?

Conclude that there is an asymptotic PTAS for bin packing.