1. In the Knapsack problem we are given a set $U$ of $n$ items where each item $i$ has size $s_i$ and profit $p_i$. The goal is to find a maximum profit subset of items that can fit in a knapsack of size $B$.

In the class, we learned that the algorithm that guesses the $\lceil 1/\epsilon \rceil$ most profitable items in the optimal solution and runs the greedy algorithm on the remaining Knapsack problem yields a PTAS. We would like to design another PTAS – first guess the set $X$ of $\lceil 1/\epsilon \rceil$ largest items in the optimal solution and runs the greedy algorithm on the remaining Knapsack problem by packing items $i$ in $U \setminus X$ in non-increasing order of $p_i/s_i$ until the knapsack runs out of space. Show that this algorithm is a PTAS for the Knapsack problem.

2. In the Bin Packing problem, we are given as input a set of $n$ items where item $i$ has size $s_i \in (0, 1]$. Our goal is to pack in the minimum number of bins of size 1.

In the class, we learned a PTAS where we partitioned items into groups of an equal size (except the last one), and each item in $B_k$ is rounded to the size of the smallest item in $B_{k-1}$, for all $k \geq 2$. Instead of this, partition items of exponentially increasing sizes as we did for the min-makespan problem on identical machines – more precisely, if an item has size $s_i \in (\epsilon(1 + \epsilon)^j, \epsilon(1 + \epsilon)^{j+1}]$, then it is rounded to size $\epsilon(1 + \epsilon)^{j+1}$. This rounding also reduces the number of distinct item sizes. However, this rounding does not lead to a PTAS. Explain why.

3. We consider the minimum makespan scheduling problem when given $n$ jobs of sizes $p_1, p_2, \ldots, p_j, \ldots, p_n$ for related machines where machines have different speeds, $s_1 \geq s_2 \geq s_3 \geq \ldots \geq s_m$. Machine $i$ with speed $s_i$ finishes workload $L_i$ at time $L_i/s_i$. The goal is to assign jobs to machines such that all jobs are finished as early as possible, i.e. $\max_i L_i/s_i$ is minimized. Give a $O(1)$-approximation for this problem.

Hints:

(a) Consider the following algorithm. Guess the optimal value, OPT. Assign jobs in decreasing order of their sizes. When considering a job $j$, assign $j$ to the slowest machine $i$ where the current load of the machine plus job $j$ does not exceed $s_i \cdot c \cdot \text{OPT}$, where $c$ is some constant you can fix. That is, the makespan of the machine will not exceed $c \cdot \text{OPT}$ if you place job $j$ on the machine.

(b) Each job $j$ cannot be assigned to machine $i$ such that $\text{OPT} < p_j/s_i$ in the optimal solution. Why? This will define a subset of machines that can possibly schedule the job.

(c) Think how to modify the 2-approximation we learned in the class to this setting where machines have different speeds.