a. The greedy algorithm taught in class with weight limit still set to W, but only run the sort by density option instead of both sort by density and sort by value; and instead of excluding the first item j that makes current weight exceed maximum allowed weight W, include that item.
Proof of correctness: including the first item j that makes our weight exceed W does not make our weight exceed $\frac{3}{2}W$ because no item is too big (i.e. $w_i \leq \frac{W}{2}\ \forall\ i$). Before including j, the current weight is $\leq W$, adding $w_j \leq \frac{W}{2}$ makes total weight $\leq \frac{3}{2}W$. This new algorithm also makes our total value $\geq V^*$ because as proved in lecture, including a fraction of item j such that the total weight is equal to W when running by sorted density makes our value equal to the optimal value V^*, including the entirety of j thus makes our total value $\geq V^*$.
Runtime Analysis: We learned that the greedy algorithm is $O(n \log n)$. The only modification to the algorithm is not running the sort by value option, which actually makes the runtime smaller. It's still $\boldsymbol{O(n \log n)}$.

b. Obtain the value of V^* by running the greedy algorithm taught in class using only density sorting and with weight limit W and taking a fraction of the value of the first item that doesn't fit. Then, run the greedy algorithm (with both sorting schemes now) on small items only (i.e. items with weight $w_i \leq \frac{1}{2}W$). If the value of that is less than V^*, then put a big item in the beginning, then run the greedy algorithm with small items only; run this for all big items. Return the run with the highest value.
Proof of correctness: Observe that the optimal solution when the weight limit is W has 0 or 1 big items, as a big item's weight $w_i > \frac{W}{2}$. If we get a total value $V < V^*$ after running the greedy algorithm on small items only at first (0 big items), then that means the optimal solution has 1 big item. We don't know which big item is included, so we exhaust all options of big items in the beginning then run the greedy algorithm on small items.
Runtime Analysis: Obtaining V^* is just running the greedy algorithm taught in class, which is $O(n \log n)$; running it on small items once is $O(n \log n)$ again; running it on all possible 1-big-item-collections is $O(n) * O(n \log n)$, as the number of big items is $O(n)$. Overall, this algorithm is $O(n^2 \log n)$.