1. The greedy algorithm taught in class with weight limit still set to , 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 that makes current weight exceed maximum allowed weight , include that item.

**Proof of correctness**: including the first item that makes our weight exceed does not make our weight exceed because no item is too big (i.e. ). Before including , the current weight is adding makes total weight . This new algorithm also makes our total value because as proved in lecture, including a fraction of item such that the total weight is equal to when running by sorted density makes our value equal to the optimal value , including the entirety of thus makes our total value .

**Runtime Analysis**: We learned that the greedy algorithm is . The only modification to the algorithm is not running the sort by value option, which actually makes the runtime smaller. It’s still .

1. Our algorithm:
2. Set weight limit to . Run the greedy algorithm on small items only (i.e. items with weight ).
3. Start over. Set weight limit to again. Put a big item in front, then run the greedy algorithm with small items only; run this for all big items.
4. Return the best run from the above two steps.

**Proof of correctness**: Observe that the optimal solution when the weight limit is has or big items, as a big item’s weight . If the optimal solution has big items, then step has a satisfying solution because it’s part a’s algorithm. If the optimal solution has big item, then there exists a satisfying solution from step because we exhaust all options of big items in the beginning. The big item’s weight , so running the greedy algorithm on small items only after putting a big item in the front (step ) is basically part a but with weight limit .

**Runtime Analysis**: step is because it’s the algorithm taught in class. Step is because the number of big items is . Overall, this algorithm is