<u>Stock Your Bookshelf</u>

Jonathan Wenger

1. I devised this algorithm very similar to the "subset-sum" algorithm we discussed in class, as they are very similar. Some differences include that while in subset-sum we can choose to include a given item or not, in this algorithm we must include one Sefer from each of the C classes of seforim, and we have to see within each class which Sefer is "most optimal". I also sorted the seforim in descending order, so when I get the solution List, I get them in backwards order, which will be in ascending order. Another factor to consider is that if it is impossible to find a solution using M dollars and c classes (where $0 \leq c \leq C$), then there will be no possible solution for M dollars and C classes (as choosing the cheapest Sefer from each class will still be more money than her budget allows). The way I checked that is I had a 2D boolean array that checked that the solution to the subproblem is "acceptable", and if it was, I marked that index in the array as false. Otherwise, I marked it as true. To get the solution List, I backtracked from the 2D array to get the seforim that were used in the solution.

2. To compute OPT(C, M), we need the optimal value for subproblems consisting of the first c items (where $0 \leq c \leq C$) for every money value $0 \leq m \leq M$. The optimal value of these subproblems is OPT(c, m).

3. Given that we need at least one Sefer from each category, the recurrence is as follows:

OPT (c, m) = {0 **if c=0**
{no solution **if (T_i>m), or (c-1!=0 and m-T_i≤0) (T_i for all of the i seforim in class c)**
{max {T_i+ OPT(c-1, m-T_i) for all of the i seforim in class c **otherwise**

Additionally, for any value of c where $0 \leq c \leq C$, if OPT (c, M) = 0, then there is no possible solution for the first c classes, which means there will certainly not be a solution for the total C classes (which I kept track of using the 2D boolean array).

4. Algorithm performance by method:
    - **maxAmountThatCanBeSpent**
        - For this method, I:
            - Sorted the C classes in descending order, which is (C log C)
            - Iterated through the C classes. Within each class I iterated through all the money values between 0 and M, and within each money value I iterated through all of the T_i seforim in that class. That totals ($C*W*T_i$).
        - Therefore, the total runtime of this algorithm is the maximum between C log C and $C*W*T_i$.
    - **Solution**
        - For this method, I:
            - Iterated through the T_i seforim at that level of c from every c between C and 0, while also checking the value of the weights from 0 to W (worst case)
        - Therefore, the total runtime of this algorithm is $C*W*T_i$.