CS170 Project

Phase II Writeup

Kenneth Kao - 25676363
Christine Munar - 25579673

In determining the input files for Phase I, we created a script that generated large, randomized files. In this script, we selected a random P, M, and N in the range of higher value numbers. We chose this strategy because we thought increasing the amount of items and having the lower bound still be a large number would increase the complexity and make it slower for algorithms with huge runtimes. We also randomized the classes and constraints. With each iteration up to C, we randomly selected, with replacement, from our list of classes.

When creating the algorithm, we decided to not stick to one algorithm alone. Instead, we created four greedy algorithms, ran them all with parallelization, took the max total resale value of all four algorithms, and returned the set of items that chosen algorithm returned.

With the first greedy algorithm, we took the remaining maximum net value (cost - resale). With each iteration of obtaining the max, we ensured its class did not conflict with the classes already in the set. We filled up the sack as much as possible, until it reached P and M as close as possible.

In the second greedy algorithm, we took the remaining minimum cost item. With this algorithm, we can fill up the sack with as many items as possible. Again, we kept in mind all the constraints of classes, P, and M.

For the third greedy algorithm it basically chooses the items with highest value/weight ratio. This is kind of a mixture of the first two because it weeds out some values that are really valuable but very heavy and some things that are very weighty but not very valuable.

For the last greedy algorithm it chooses the most valuable items until the sack is 70%. Then it fills it with the lightest items. This is a “hybrid” algorithm to try to combine the two, as both versions are pretty solid and legit greedy strategies.

For runtime, we create lists and a dictionary in the beginning by going through every item once. This is O(items) for initialization. Then for each constraint the runtime for creating the sets of conflicting categories is O(constraints). Then for each of the greedy algorithms they all have similar runtimes. They too go through all the items. So it is O(items) again. So basically our runtime in total is linear: it is O(items) + O(constraints).