MIE-PAA
Paul Rondeau
rondepau
October 13, 2014

Homework 1

In this assignment, the task was to solve the common problem: Knapsack Problem. Two algorithms were created and compared for accuracy and timing. The first was to be a brute force algorithm designed to test every possible instance of the problem. The second was an algorithm designed to compare the cost/weight ratio of each item and make selections based on this value. The brute force algorithm can only be completed in exponential time, $O(2^n)$, since it must test every combination of elements. The greedy cost/weight algorithm can be completed in much less time.

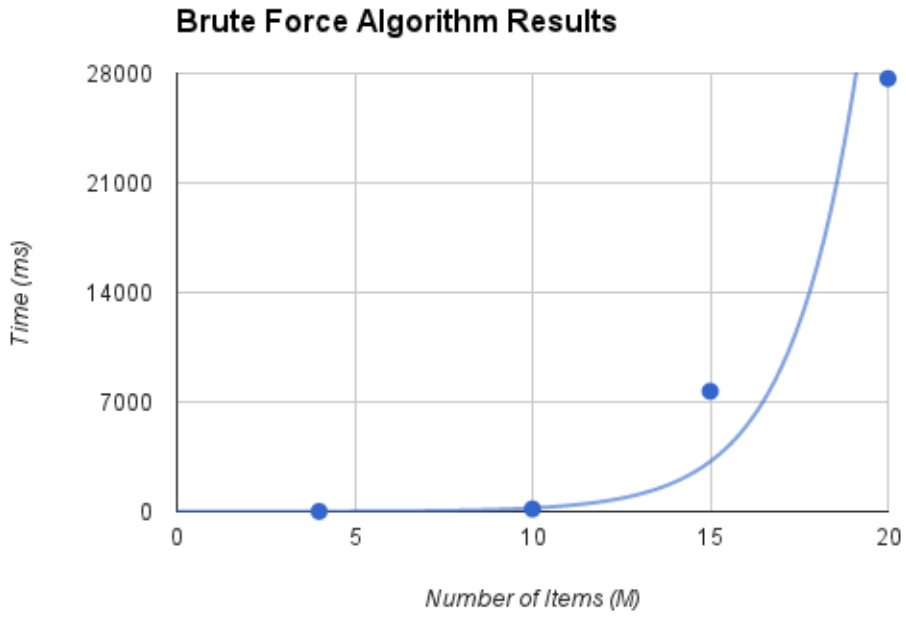
The brute force algorithm would use a binary counter to iterate over all possibilities of the knapsack filling. Each increment of the counter would represent a new instance of the problem. The counter was then used to index into the array wherever there was a '1'. This would imply the item was to be added to the knapsack. This was repeated for all values, calculating the potential of the weight and cost values while checking if it was a valid solution. Pseudo code to describe this algorithm:

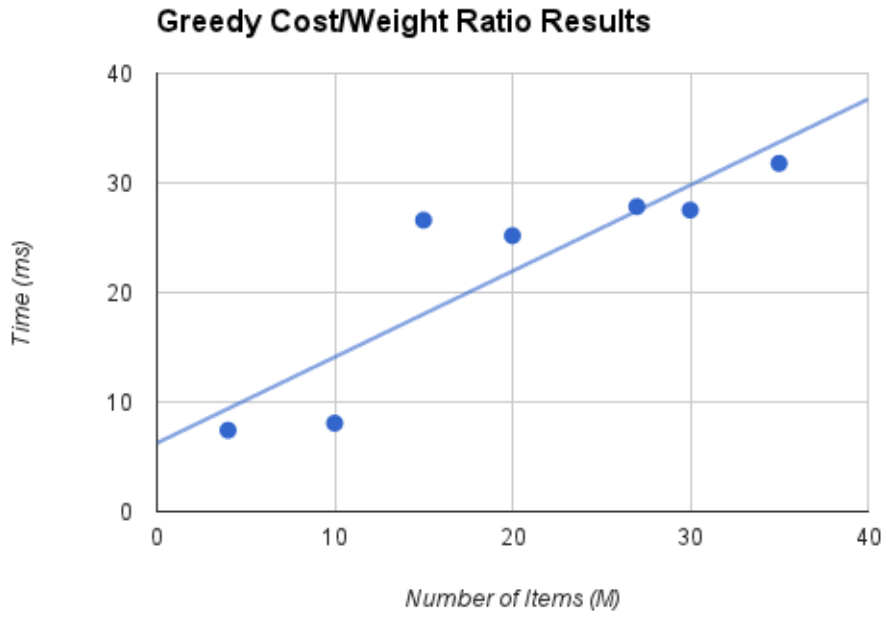
```
count = 0
maxCost = 0
while count < 2**n:
        binCount = bin(count)
        for '1' in binCount:
                weight += W[index]
                cost     += C[index]
        if cost > maxCost && weight <= M
                maxCost = cost
        count++
```

The greedy cost/weight algorithm used a simple approach, calculate the cost/weight ratio for each element, insert that value into an array, find the max element in this new array and add it to the knapsack (remove it from the array). This is outlined in this pseudo code:

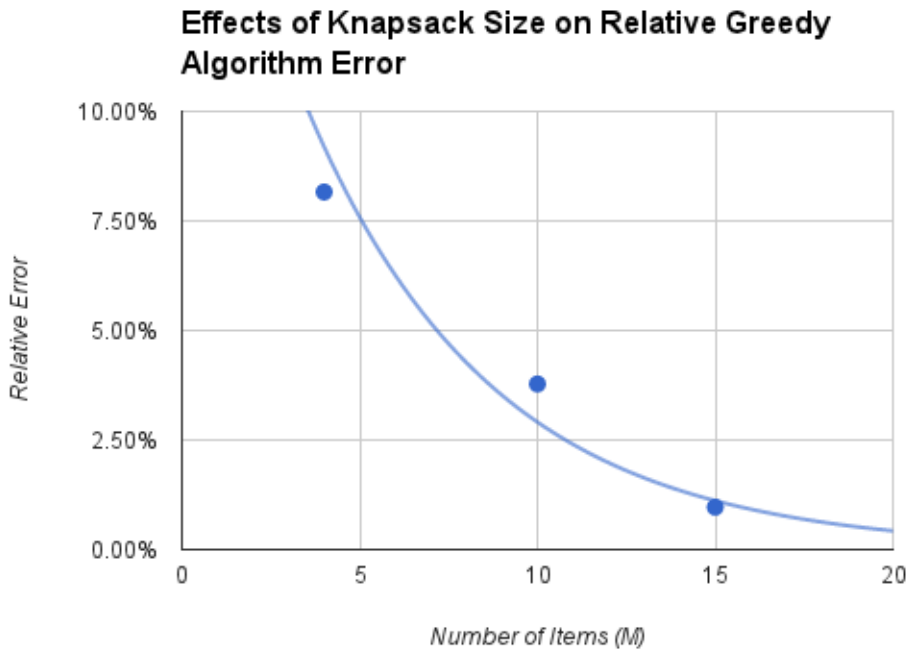
```
for element in W:
        ratio.append(C[index]/W[index])
maxIndex = max(ratio).index
while len(ratio) > 0 && weight + W[maxIndex] < M:
        weight += W[maxIndex]
        cost     += C[maxIndex]
        ratio.remove(maxIndex)
        maxRatio = max(ratio).index
```

Results from both algorithms are as follows:





As expected, the greedy algorithm outperforms the brute force by far and does much better at computing large instance sizes, as the increase seems to be linear, whereas, the brute force is clearly an exponential increase.
As for the error associated with the brute force algorithm, this graph illustrates the effects of the increasing knapsack size on relative error.



It can be seen that as the number of items increase, the relative error becomes less.

All calculations included in this report have come from a bash script labeled “run.sh” The values were taken from the average of several trials, and then divided by 50 since there were that many instances in the test files.

In conclusion, for large sets of data, where complete accuracy is not required, the greedy algorithm performs quite well with limited resources. Overall the greedy provides a quick and easy way to find the correct answer most of the time.

Link to Source Code:
https://edux.fit.cvut.cz/courses/MI-PAA/_media/en/student/rondepau/assignment1.zip

https://github.com/parondeau/school/tree/master/3B/alg/assignment1