Assignment 1: Parallel Knapsack

In this assignment you will be writing a parallelized solution to the knapsack problem given a serial solution. The knapsack problem is as follows.

The Knapsack Problem:

Given n items with weights w[i] and values v[i] and capacity c of your sack, you would like to put items into your sack maximizing your value while not exceeding your capacity. That is, you are choosing a subset of the items so that the sum of the chosen items weights is not larger than c and that the sum of the values is larger than any other subsets would be.

Explanation of the serial solution:

The serial solution uses an approach called dynamic programming. Essentially it keeps track of previous information to prevent itself from trying the same thing many times. It follows the following strategy: start with the first item and consider the two cases, either adding it to your sack or leaving it. If you added it, you now have value v[i] and capacity c - w[i]. If you left it your value and capacity are unchanged.

We define our optimal value as follows: opt[curr][c] is the highest possible value only considering items curr to n with capacity c. Thus, we can have a recurrence relation for opt[curr][c] as

opt[curr][c] = opt[curr + 1][c] (don’t take the item)
Or, if w[curr] <= c, = opt[curr + 1][c - w[curr]] + v[curr] (add the item)

And we simply take the max of those two possible values.

Your Assignment:

You will be writing a parallel version of knapsack. Feel free to use the example program provided, but if you find that writing your own solution would be easier, feel free to as well, just make sure input is taken in the same manner (take inputs n and c, then w[i], v[i]). You should also take the number of threads to run in parallel as a command line parameter.

Once you have done this, test your code on the provided examples (except for test0, which should just be used for debugging). Run it at least 10 times to make sure that it always gives the same output as the serial version (if not, then the parallel version isn't correct). If your version is correct, time your execution at least 10 times for 1 thread, 2 threads, 4 threads, 8 threads, and 16 threads and report the **average execution time for each on each input**. Also record the average execution time of the provided serial version on each of the inputs.

458 Additional Work:

For each input, find the optimal number of threads to reduce run time. This might be a different number for each input. You might have to increase the number of threads beyond 16, or you might have to try with a number between 4 and 8, for instance. Find this number where the run time is minimized (and speedup is maximized). If there are several values that have a similar result, pick the one with the lowest number of threads.

Once you have done this, plot your parallel version's speedup vs number of threads (y axis is speedup, x axis is number of threads used) for each input using the data you collected from above.

Summary of Submission:

Your submission should include:
Your parallel version of knapsack
A readme explaining your code
A text file called averages.txt that gives the average execution time of your parallel version on 1, 2, 4, 8, and 16 threads as well as the provided serial version on each input.
(458) Your values of the number of threads needed to achieve the maximum speedup for each input, as well as the graphs of the speedup vs the number of threads on each input.