Naive Analysis

Given a chain of n matrices, the naive method will traverse the entire $n+1$ dimension chain, and calculate a cost $n-1$ times. It does this by multiplying the dimensions from left to right. The equation to calculate cost is $cost = cost + p_{i-1} * p_i * p_{i+1}$. So there will be $n-1$ additions and $2n-2$ multiplications. This is a linear number of arithmetic operations. The overall runtime is linear in the number of matrices $O(n)$.

Greedy Analysis

In the greedy algorithm the input is an array of $n+1$ dimensions where n is the number of matrices, and a range of the array to loop through. The algorithm consists of a loop that executes $(n+1)-2 = n-1$ times to determine the minimum element k, then the function calls itself twice, first with a range of size k and second with a range of size $n+1-k$. Therefore a recurrence relation for the greedy algorithm is $T(n) = T(k)+T(n+1-k)+n+1$. In the worst case, $k = 1$ every time so the function is called n times, each time looping through n elements for a runtime of $O(n^2)$

DP Analysis

In the DP algorithm, the input is an array of $n+1$ dimensions where n is the number of matrices. The algorithm uses a 2D array to store the the answers to each subproblem which costs $O(2n)$ to initialize. Then it iterates through 3 nested for loops, each of which will scan nearly the whole array at some point. Therefore the runtime is $O(n^3)$