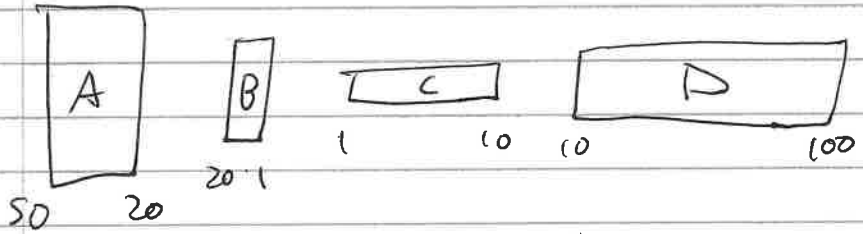
Chain Matrix Multiplication

This is a niche but important problem that could only have been solved using DP. It has nothing to do with faster matrix multiplication, but we want to reduce the number of operations to multiply several matrices. Depending on how we do them in order



	cost
A((BC)D)	120,200
((AB)C)D)	60,200
(AB)(CD)	7,000

choosing the correct association is important. Note that matrices do not commute, $AB \neq BA$, but they are associative $(AB)C = A(BC)$.

solution

array: $dp[1 \ldots n][1 \ldots n]$ given matrix dimensions (d_0, d_1), (d_1, d_2), (d_2, d_3), $\ldots (d_{n-1}, d_n)$ for $A_1, A_2, A_3, \ldots A_n$ so $n+1$ dimensions and n matrices. we want $dp[i,j]$ = the min cost to compute $A_i \ldots A_j$ for $1 \leq i \leq j \leq n$.

base cases: lets discuss our cost function. We may approximate the cost of computing an $(d_0 \times d_1) \cdot (d_1 \cdot d_2)$ matrix as if it takes $d_0 \cdot d_1 \cdot d_2$

$A((BC)D)$ costs $20 \cdot 1 \cdot 10$ (BC) $+ 20 \cdot 10 \cdot 100$ ((BC)D) $+ 50 \cdot 20 \cdot 100 = 120,200$

$(AB)(CD)$ costs $50 \cdot 20 \cdot 1$ (AB) $+ 1 \cdot 10 \cdot 100$ (CD) $+ 50 \cdot 1 \cdot 100$ ((AB)(CD)) $= 7000$

then $dp[i,i]$ = zero matrix mults so 0.

$dp[i,j] = 0$ if $j < i$, only fill up top half diagonally. The answer we will return is $dp[1,n]$ = min cost to compute $A_1 - A_n$

recurrence: lets take a second to guess this. Consider the last step. we can choose from a few ways:

$(A_1)(A_2\ A_3\ A_4 \ldots)$
$(A_1\ A_2)(A_3\ A_4 \ldots)$
$(A_1\ A_2\ A_3)(A_4 \ldots)$
$(A_1\ A_2\ A_3\ A_4)(\ldots)$

we want the minimum split. so $dp[1,n] =$

min cost to compute $A_1 \ldots A_k$ + min cost to compute $A_{k+1} \ldots A_n$ + cost to combine

we recursively

$$dp[1,n] = \min_{1 \le k < n} \left[dp[1,k] + dp[k+1,n] + d_0 \cdot d_k \cdot d_n \right]$$

this will be the last step. we may generalize it to intermediate steps:

$$dp[i,j] = \min_{i \le k < j} \left[dp[i,k] + dp[k+1,j] + d_{i-1} d_k d_j \right]$$

great! now we just need to ensure we fill in the table in the correct order, may want to fill in the table correctly:

```
def CMM(d_0 ... d_n)
    dp[1..n][1..n] all zeroes
    for i in 1..n  dp[i,i] = 0
    for s = 2 to n
        for i = 1 to n-s+1:
            j = i+s-1
            dp[i,j] = min_{i<=k<j} [ dp[i,k] + dp[k+1,j] + d_{i-1} d_k d_j ]
    return dp[1,n]
```

s = chain length

run time: we fill in $n^2/2$ cells, each takes the range of k, so $O(n^3)$ time. much better than brute force, The number of parenthetizations is the catalan number $\sim C(4^n)$

$C_n \sim \frac{4^n}{n^{3/2}\sqrt{\pi}}$

example: $d_0\ d_1\ d_2\ d_3\ d_4$

50 20 1 10 100

0	1000	1500	7000
	0	200	3000
		0	1000
			0

$dp[1,2] = 50 \cdot 1 \cdot 20 = 1000$

$dp[2,3] = 20 \cdot 1 \cdot 10 = 200$

$dp[3,4] = 10 \cdot 1 \cdot 1000 = 1000$

$$dp[1,3] = \min \begin{cases} dp[1,1] + dp[2,3] + d_0 d_1 d_3 & = 0 + 200 + 10{,}000 = 10{,}200 \\ dp[1,2] + dp[3,3] + d_0 d_2 d_3 & = 1000 + 0 + 500 = \boxed{1500} \end{cases}$$

$$dp[2,4] = \min \begin{cases} dp[2,2] + dp[3,4] + d_1 d_2 d_4 & = 0 + 1000 + 2000 = \boxed{3000} \\ dp[2,3] + dp[4,4] + d_1 d_3 d_4 & = 200 + 0 + 20{,}000 = 20{,}200 \end{cases}$$

$$dp[1,4] = \min \begin{cases} dp[1,1] + dp[2,4] + d_0 d_1 d_4 & = 0 + 3000 + 100K \\ dp[1,2] + dp[3,4] + d_0 d_2 d_4 & = 1000 + 1000 + 5000 = \boxed{7000} \\ dp[1,3] + dp[4,4] + d_0 d_3 d_4 & = 1500 + 0 + 50K \end{cases}$$

last choice of $k = 2$ implies top split was like $(A_1 A_2)(A_3 A_4)$