Samar Ibrahim Antar

Homework#1: Matrix Multiplication

- **the time complexity of a naïve approach is :**

✓ there are 3 for loops.

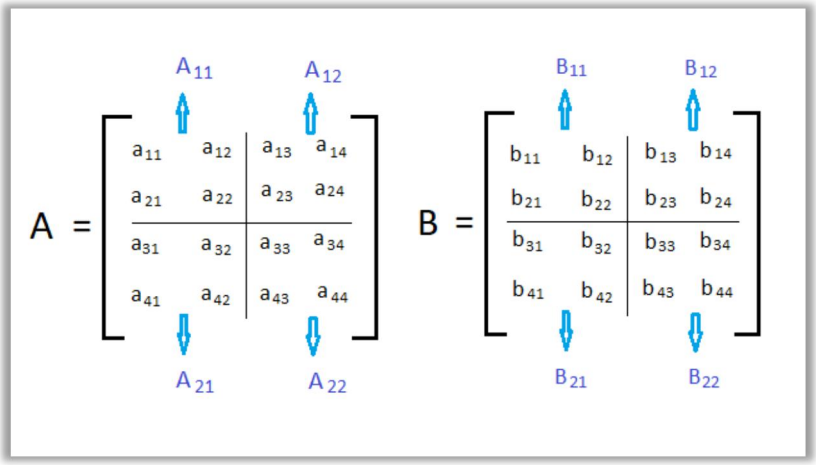
✓ It needs three for loops; For accessing all the elements of any matrix we need two for loops. But for finding the product, it requires one additional for loop. That's how it is taking 3 for loops.

✓ So, Time complexity: **O(n^3)**

- **the time complexity of the used divide-and-conquer approach:**

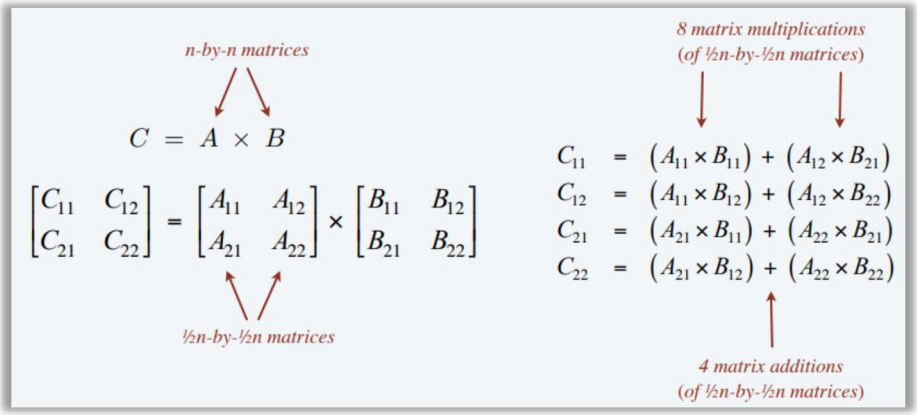
✓ To multiply two n-by-n matrices A and B:

>>Divide: partition A and B into ½n-by-½n blocks.



>>Conquer: multiply 8 pairs of ½n-by-½n matrices, recursively.

>>Combine: add appropriate products using 4 matrix additions.



✓ So, the time complexity:

#	work
1	kn
8	$8K(n/2)= K4n$
⋮	⋮
8^i	$8^iK(n/2^i)=K4^in$
⋮	⋮
$8^{\log_2 n}$	$K8^{\log_2 n} = Kn^3$

So, total= $\sum_{i=0}^{\log_2 n} 8^iK(n/2^i) = \Theta(n^3)$

✓ Thus, a naïve divide and conquer algorithm runs in **O(N^3)**.

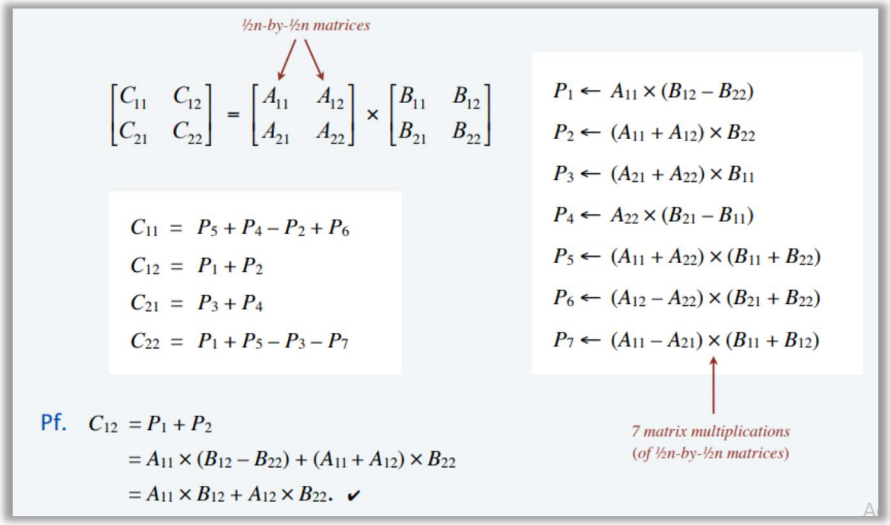
✓ The passed matrices must be nxn matrices (square matrices), where n is a power of 2

✓ parameter n is the size of matrices.

● **Optimized divide and conquer:**

✓ Strassen's trick:

>>Key idea: Can multiply two n-by-n matrices via 7 (½n-by-½n matrix) multiplications (plus 11 additions and 7 subtractions).



✓ So, the time complexity:

#	work
1	kn
7	$7K(n/2)=K(7/2)$
⋮	⋮
7^i	$7^iK(n/2^i)=K(7/2)^in$
⋮	⋮
$7^{\log_2 n}$	$K7^{\log_2 n}=Kn^{\log_2 7}$

So, total= $\sum_{i=0}^{\log_2 n} 7^iK(n/2^i) = O(n^{\log_2 7}) = O(n^{2.8074})$

✓ Thus, a divide and conquer(Strassen Algorithm) which runs in $\mathbf{O(N^{Log_2\,7})}$, that is almost equal to $\mathbf{O(N^{2.8074})}$.

- **Output of naïve.cpp ,divide_conquer.cpp & strassen_algorithm.cpp** : are the same which showing in the following figure.

```
Enter n: dimension of matrices n x n (n is a power of 2)
4
Enter elements of first matrix
1 1 1 1
2 2 2 2
3 3 3 3
4 4 4 4
Enter elements of second matrix
1 1 1 1
2 2 2 2
3 3 3 3
4 4 4 4
Result matrix is
10 10 10 10
20 20 20 20
30 30 30 30
40 40 40 40

Process returned 0 (0x0)   execution time : 44.813 s
Press any key to continue.
```

- **Note:** I used pointers in **strassen_algorithm.cpp** and vectors in **divide_conquer.cpp** to make functions accept any dimension of square matrices(nxn).

  - why I didn't use pointers again in **divide_conquer.cpp?** Just for remembering C++ and learning more about it.