1. Analytically determine the value of n0 that optimizes the running time of this algorithm in this model.

n: input matrix size

- standard matrix multiplication complexity

In order to speed up the multiplication operation, make matrix memory to be flatten in 1d array

And when iterate the row of A, col of B, use pointer operation aa, bb

As a result, operation is decreased from 7 \* n ^ 3 to 5 \* n \* 3

n : a \* b – multiply n times

n : sum += addition n times

index operation: 3 \* n (k++, aa++, bb += m)

S(n) = n \* n \* (5 \* n ) = 5 \* n ^ 3

- Strassen matrix multiplication complexity

Split Matrix to sub matrix –2 \* n^2

Generate s0 ~ s10: 10 \* 5 \* n^2 / 4

Generate p1 ~ p7: 7 \* T(n)

Reconstruct matrix: 15 \* n ^ 2 / 4

- Caluculating the cross-point

Let’s assume that standard matrix multiplication complexity S(n) = p \* n ^ 3

Strassen matrix multiplication complexity T(n) = 7 \* T(n/2) + q \* n^2

p = 5, q = 20

Generally p < q, because Strassen algorithm requires additional operation for recuristion

So for small n, S(n) < T(n), but while n is increased, S(n) > T(n)

We can identity this change point (cross point) using numerical anaylsis.

p = 5, q = 65

