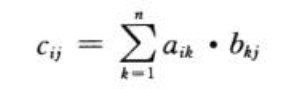
常规的矩阵计算

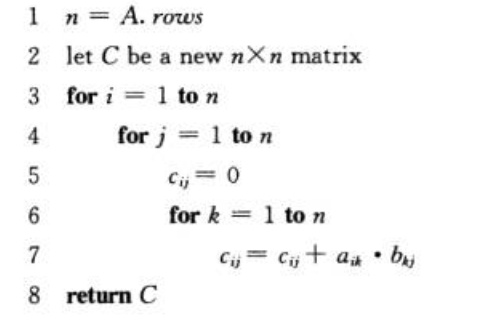
假设A与B是两个n\*n的矩阵

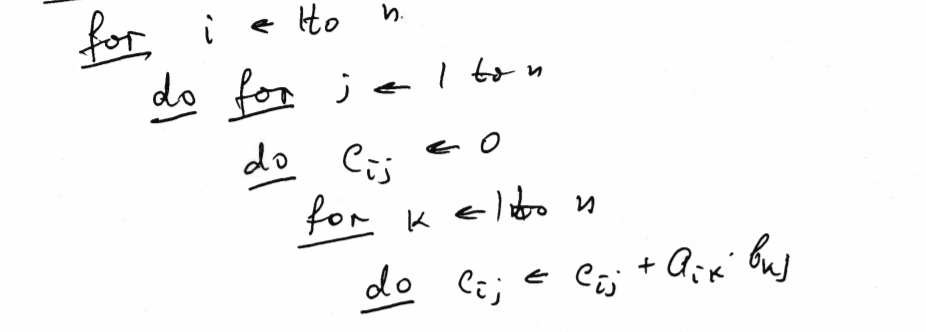


那么相乘们每个元素Cij为

,k从1到n

那么标准暴力解为

就很好理解，ij两个大循环，然后cy就是当前循环的和，然后开始loopk,这样就能算到cij的数

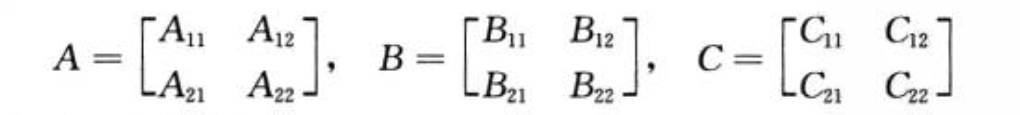


三个for,运行时间是n^3

Divide and conquer法

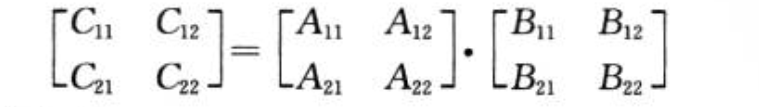
把大的问题分成小的

matrix A\*B=C

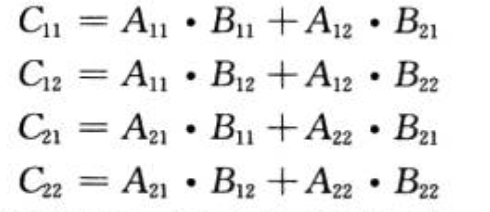


我们可以加上辅助线，把这些矩阵四等切割

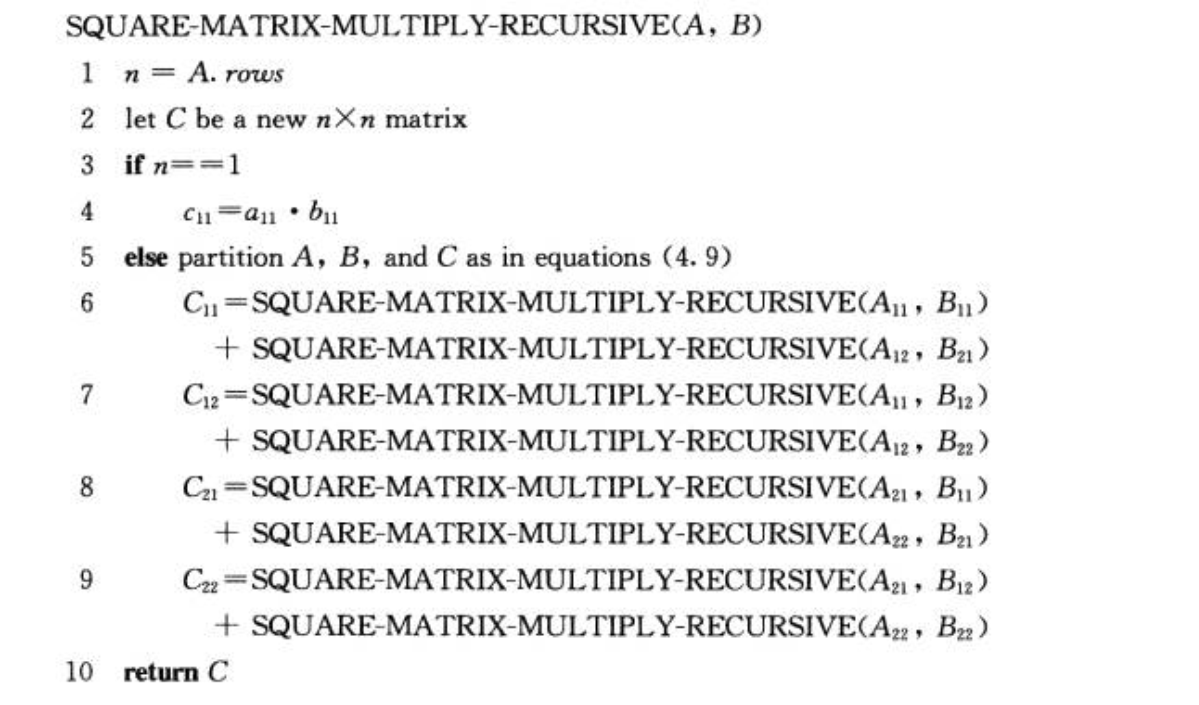
那么



等价于

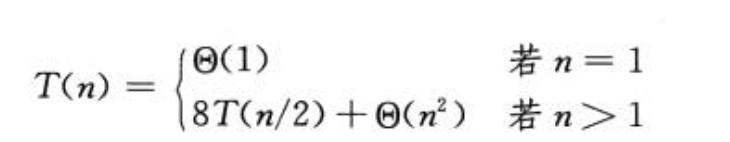


因此可以设计一个递归的conquer and divide



n=1就不说了，conquer and divide 必备

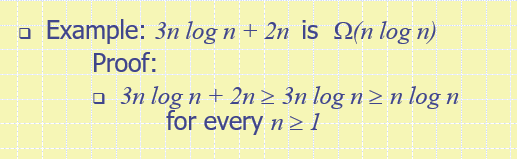
然后直接套公式

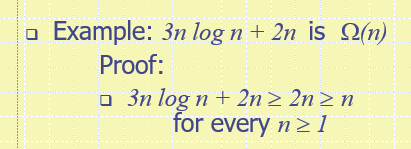


s1-w1

Big-O代表着最坏情况，Big Omega代表着最好情况







注意O可以无穷大，fN如果小于N，必然就小于NLOGN。。。

OMEGA可以无穷小，他如果大于N，必然就大于LOOGN。

都是TRUE，但是不一定是good好的。不是最优解

只有O和OMEGA交界，得到的big-theta才是最优解

算法只是理论上的，并不代表exact运行时间，所以我们用渐进符号asymptotic notation表示run-time

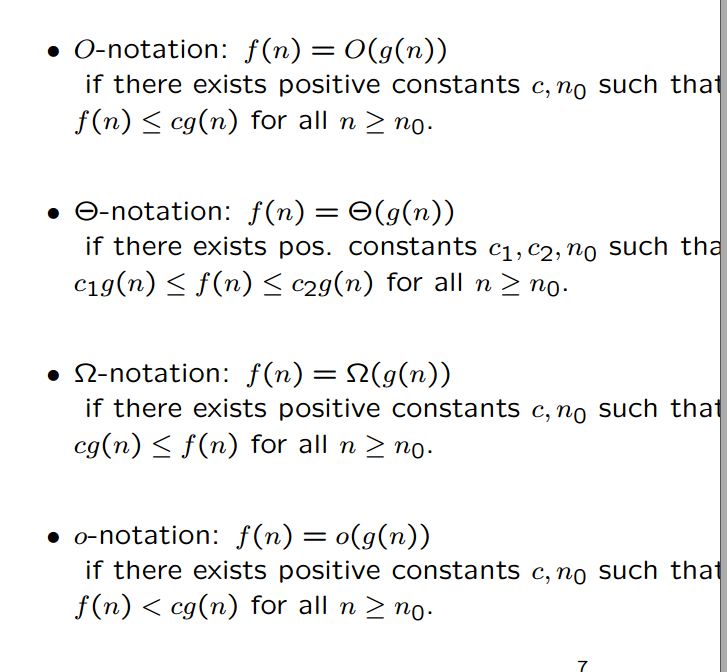
g(n)指 这个asymptotic notation理论所用时间，fn代表实际时间

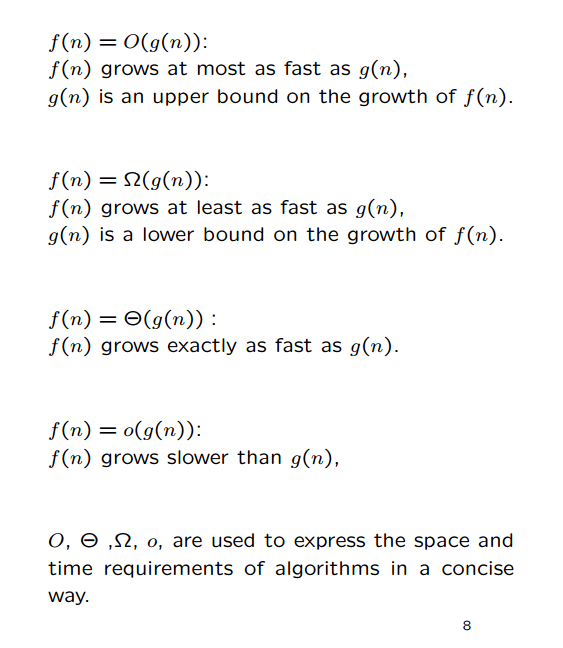
O最坏情况，实际用时f(n)必然小于O里的时间

OMEGA最好情况， 实际必然大于

theta最常用，f(n)必然落在之间

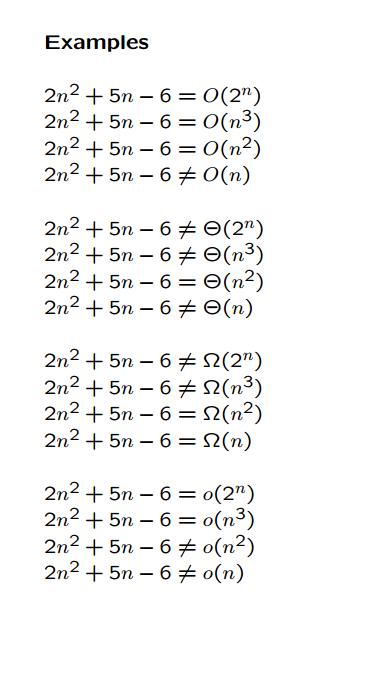
小o与大O唯一区别就是不能等于

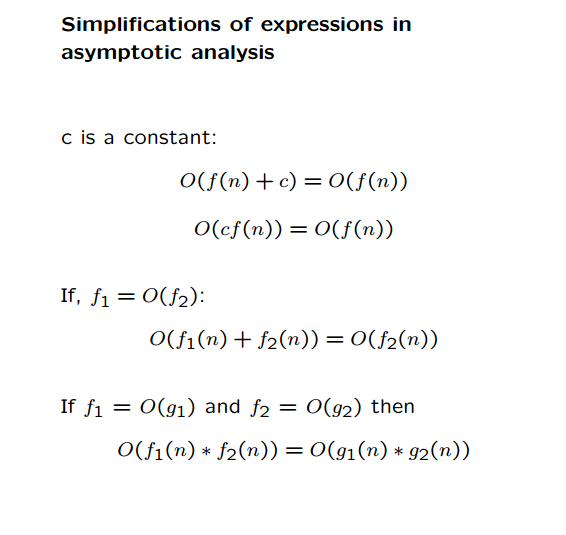




还是上面说的，f(n)实际，g(n)理论

打O小o唯一区别就在于一个能等于，一个不能等于





然后省略常数加系数

如果其中一个函数是另外一个函数的O，可以省略

