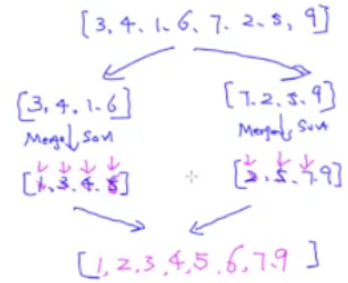
## 1. merge sort 归并排序

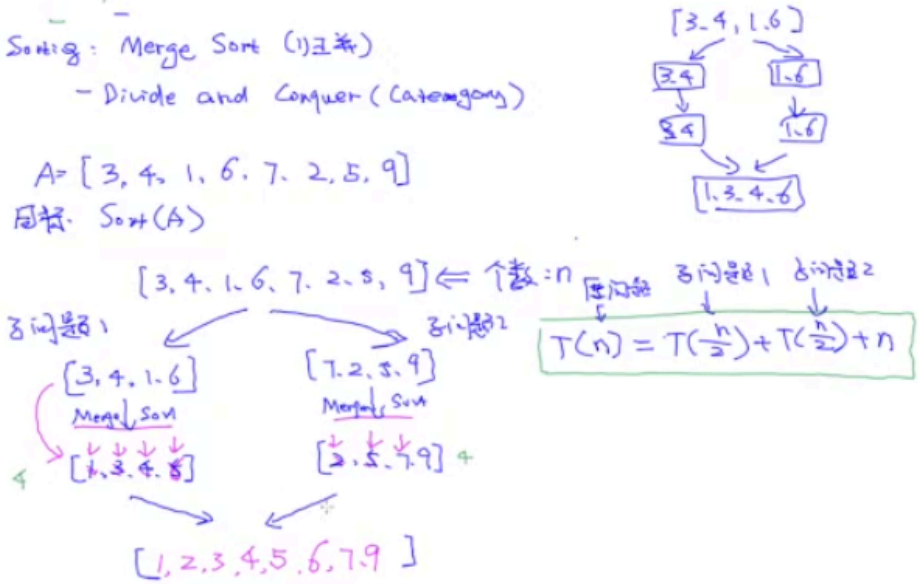
Sorting: Merge Sort (归并)

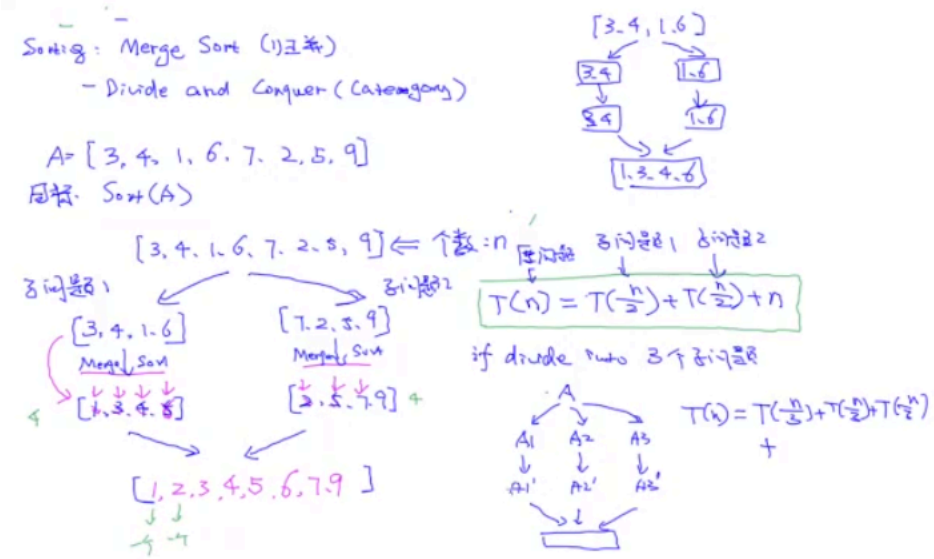
- Divide and Conquer (Catemgory)

A = [3, 4, 1, 6, 7, 2, 5, 9]

目标: Sort(A)





T(n) = O(n*logn)

## 2. Master Theorem

一个问题写成比它更简单的小问题

The Master Theorem applies to recurrences of the following form:

$$T(n) = aT(n/b) + f(n)$$

where $a \geq 1$ and $b > 1$ are constants and $f(n)$ is an asymptotically positive function.

There are 3 cases:

1. If $f(n) = O(n^{\log_b a - \epsilon})$ for some constant $\epsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$.
2. If $f(n) = \Theta(n^{\log_b a} \log^k n)$ with[1] $k \geq 0$, then $T(n) = \Theta(n^{\log_b a} \log^{k+1} n)$.
3. If $f(n) = \Omega(n^{\log_b a + \epsilon})$ with $\epsilon > 0$, and $f(n)$ satisfies the regularity condition, then $T(n) = \Theta(f(n))$.
   Regularity condition: $af(n/b) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large $n$.

时间复杂度 https://www.jianshu.com/p/f4cca5ce055a

Master http://people.csail.mit.edu/thies/6.046-web/master.pdf

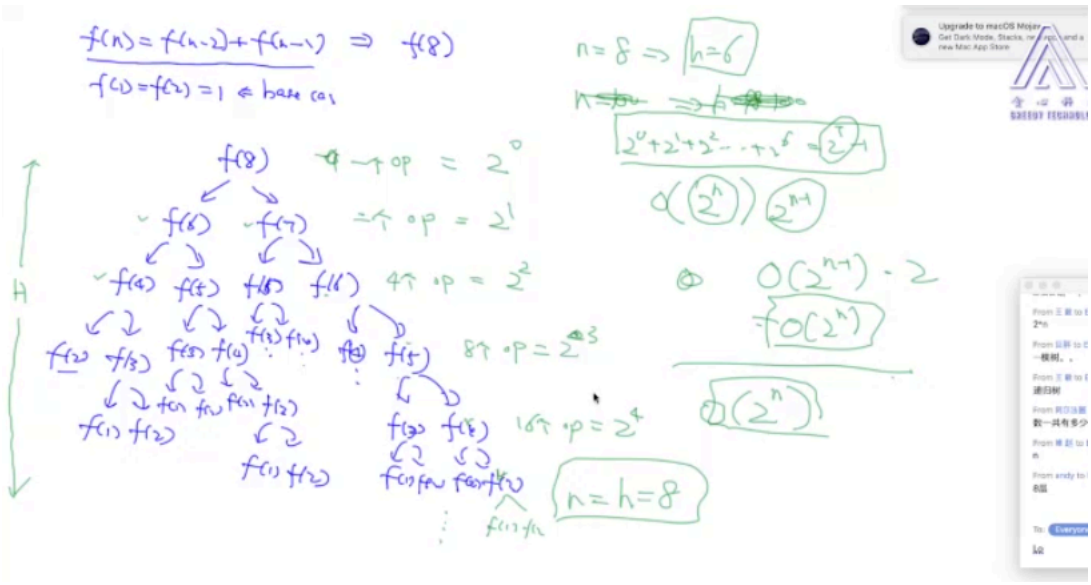
## 3. Fibonanci Number

见 jupter

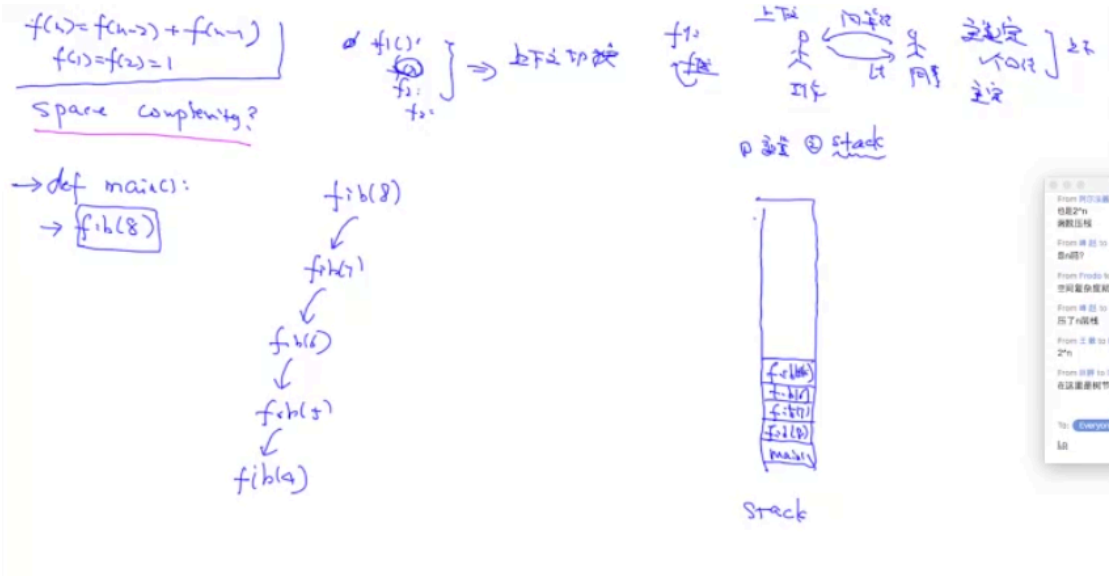
时间复杂度：

T(n) = T(n-2) + T(n-1)

T(n) = 2^n



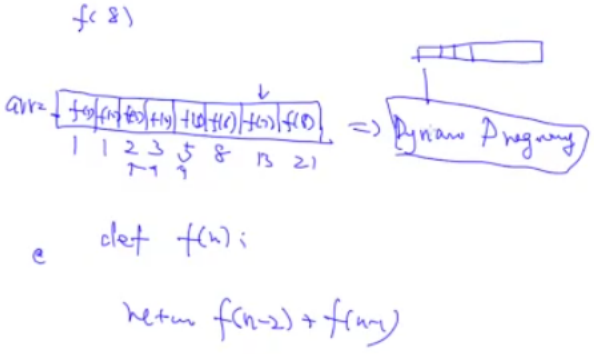
空间复杂度（针对一些递归算法）
上下文切换：变量、stack（操作系统知识）
Fib8→fib7 就是一次上下文切换，stack 会多一层



到 fib2 会出栈，因为我们知道它的数值是多少。
求 fib3 需要知道 fib2 和 fib1，而两者已知（=1），所以可以求得 fib3=2，这样 fib3 就可以出栈。
求 fib4 需要知道 fib3 和 fib2，对于 fib2 进栈，但我们知道 fib2=1，所以用 1 来替代 fib2，然后 fib2 出栈。
从 fib4 到 fib5，此时求解 fib5 需要知道 fib3，fib3 需要知道 fib1 和 fib2，然后 fib1 和 fib2 分别依次进栈出栈，求得 fib3。Fib3 再与 fib4 一起求得 fib5.
直到求解完 fib8，它出栈，栈内只剩 main 函数。
所以求解 fib8 时，我们最多用到 8 个内存空间。
空间复杂度 O(n).

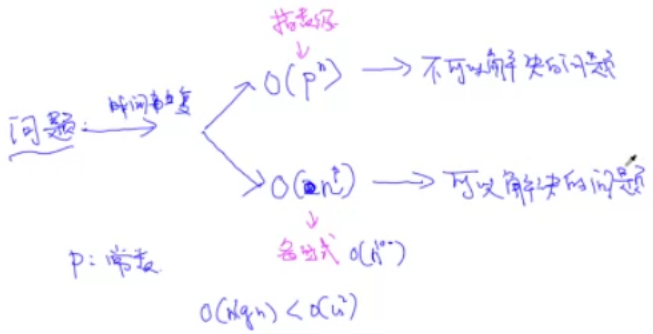
## 4. dynamic program

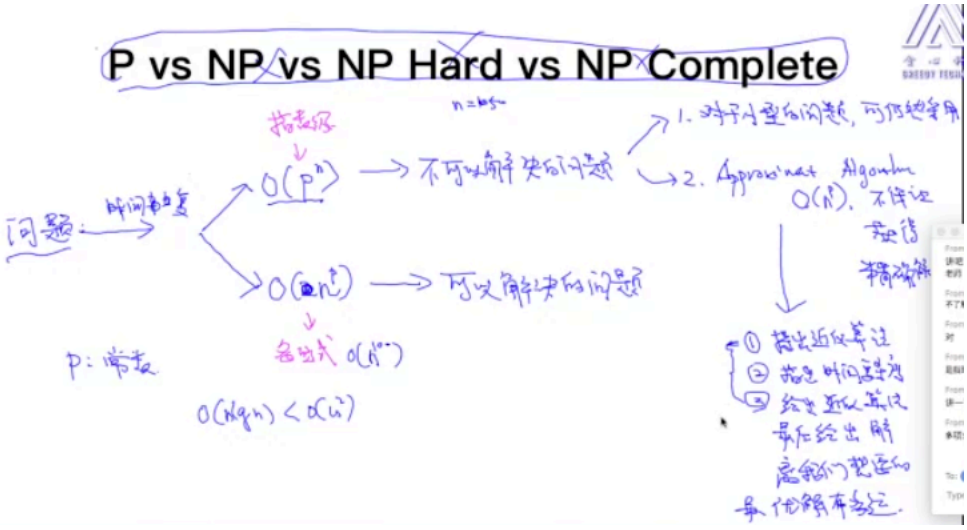
能复用的变量、资源等等尽量复用



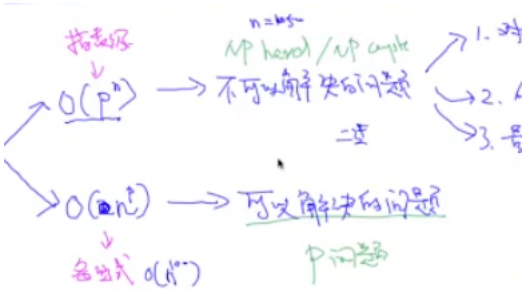
用动态规划的思想改进算法，见 jupter 2
因为求解某个 fib 数值时，它只依赖前两个数值，所以我们只需要维护前两个空间。
https://algorithms.tutorialhorizon.com/dynamic-programming-edit-distance-problem/

## 5. P, NP, NP hard, NP complete






量子计算


指数级
O(P^n)
n=log
NP hard / NP cplt
不可以解决的问题
二选
1.
2.
3.
O(n^2)
可以解决的问题
多项式
P问题