

NLP Bootcamp (2)

2019年01月22日

点此获取更多资源

Today's Agenda

- 复杂度回归
- 归并排序以及Master Theorem
- P, NP, NP hard, NP complete 问题
- 斐波那契数的计算
 - 递归实现
 - 循环实现
- 问答系统介绍

$$T(n) = 2 \cdot T\left(\frac{n}{2}\right) + n$$

$$a=2, b=2$$

$$n^{\log_b a} = n = f(n) \Rightarrow \text{Case 2}$$

$$f(n) = n = n \lg n$$

$$T(n) = O(n \cdot \log^{k+1} n)$$

$\Rightarrow O(n \log n)$

$$T(n) = 2 \cdot T\left(\frac{n}{2}\right) + n$$
$$a=2, b=2, f(n)=n=o(n)$$

The Master Theorem applies to recurrences of the following form:

$$\boxed{T(n) = aT(n/b) + f(n)}$$

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 $k \geq 0$, 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 .

$$\begin{aligned} \textcircled{1} \quad 0 \left(n^{\log_b a} \right) &> f(n) \Rightarrow O(n^{\log_b a}) \\ \textcircled{2} \quad O(n^{\log_b a}) &= f(n) \Rightarrow O(n^{\log_b a} (\log^k n)) \\ \textcircled{3} \quad O(n^{\log_b a}) &< f(n) \Rightarrow \text{Case 3} \end{aligned} \quad \left. \right\}$$

Master Theorem

$$a=3, b=2$$

$$T(n) = 3T(n/2) + n^2$$

$$r n^{\log_b a} = n^{\log_2 3} = n^{1.6}$$
$$f(n) = n^2$$

$$n^{\log_b a} = n^{1.6} < r^2 = f(n)$$

$$n^{\log_b a} < f(n) \quad [3]$$

$$\boxed{T(n) = O(n^2)}$$

$$T(n) = 4T(n/2) + n^2$$

$$\Rightarrow a=4, b=2, f(n) = n^2$$

$$n^{\log_b a} = n^{\log_2 4} = n^2 = f(n)$$

∴ Case 2

$$f(n) = n^2 = \underline{n^2 \lg n}$$

$$\begin{aligned} T(n) &= O(n^2 \cdot \log^{k+1} n) \\ &= O(n^2 \lg n) \end{aligned}$$

$$T(n) = 2T\left(\frac{n}{2}\right) + n$$

$$\Rightarrow T(n) = O(n \lg n)$$

$$T(n) = 2T\left(\frac{n}{2}\right) + n \log n$$

$a=2, b=2$

$n^{\log_2 2} = n = f(n) \Rightarrow \boxed{\text{Case 2}}$

$$f(n) = n \lg n$$

$\boxed{k=1}$

$$T(n) = O(n^{\log_b a + \frac{1}{k}})$$

$$= O(n \cdot \cancel{\log}^2 n)$$

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 $k \geq 0$, then $\boxed{T(n) = \Theta(n^{\log_b a} \log^{k+1} n)}$.

$$f(n) = \boxed{n \lg n}$$

$$= n \lg^2 n$$

$$= \lg n$$

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 .

Master Theorem

$$\log_b a \quad \log_b a$$

$$T(n) = 16T(n/4) + n$$

$$a=16, b=4$$

$$n^{\log_4 16} = n^2 > n = f(n)$$

$$T(n) = O(n^2)$$

$$T(n) = 2T(n/4) + n^{0.51}$$

$$a=2, b=4$$

$$n^{\log_4 2} = n^{0.5} < n^{0.51}$$

$$T(n) = O(n^{0.51})$$

Master Theorem

$$T(n) = 16T(n/4) + n!$$

a=16, b=4

$n^{16 \cdot 4} = n^2 < n!$

$T(n) = O(n!)$

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 $f(n) = \Theta(n^{\log_b a} \log^k n)$ with¹ $k \geq 0$, 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 .

$$T(n) = \boxed{2^n T(n/2) + n^n}$$

Exercise?

Master Theorem

Hw $T(n) = 64T(n/8) - n^2 \log n$

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 $f(n) = \Theta(n^{\log_b a} \log^k n)$ with¹ $k \geq 0$, 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 .

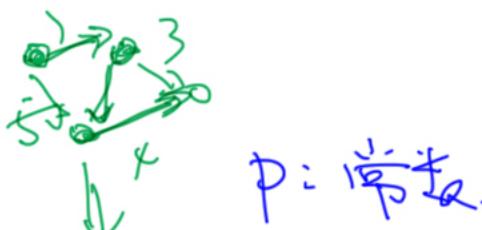
Hw $T(n) = \sqrt{2}T(n/2) + \log n$

NP: 什么

问题

可以多变或复杂
在内存能 verify 的问题

问题: 时间复杂度



Verify

指数级

$O(P^n)$

$O(2^n)$

多项式 $O(k^n)$

$O(n \lg n) < O(n^2)$

$P \in NP$

$n=10^{50}$
NP hard / NP complete

不可以解决的问题

二进制

NP 问题

问:

① ② 1.3
③ 1.2

- 1. 对于小型的问题，可以使用
- 2. Approximate Algorithm $O(n^P)$, 不保证
获得精确解
- 3. 贪心计算

① 指出近似算法
② 指出时间复杂度
③ 给出近似算法
最后给出解
离我们最近的
最优解有多远。

案例：搭建一个智能客服系统？

常见的问题 (FAQ) :

1. 本课程是线上课程还是线下课程？

回答：线上课程为主

2. 课程有助教吗

回答：每门课程都配备专业助教

3. 学习周期是多久啊？

回答：通常来讲在3-4个月不等

4. 如果不满意可以退款吗？

回答：前两周提供无条件退款

5. 老师都是什么背景啊？

回答：绝大部分都是全美前10学校的博士

6. 课程会有考试吗

回答：有的。一般包括期中和期末

7. 我只有编程基础，可以报名吗

回答：对于初级的项目班只要求编程基础

8. 课程有实操吗

回答：大部分都是实操，动手能力是最重要的

9. 课程为什么贵？

回答：跟别的知识付费不一样，我们会提供很多教学服务，辅助完成学员做完所有的项目

10. 课程学完了能做什么？

回答：可以找相关岗位的工作问题不大

11. 课程多久开一次啊？

回答：我们每个月开一期，但价格通常会不断升高

案例：搭建一个智能客服系统

常见的问题 (FAQ) :

1. 本课程是线上课程还是线下课程？

回答：本课程是线上课程还是线下课程？

2. 课程有助教吗

回答：每门课程都配备专业助教

3. 学习周期是多久啊？

回答：通常来讲在3-4个月不等

4. 如果不满意可以退款吗？

回答：前两周提供无条件退款

5. 老师都是什么背景啊？

回答：绝大部分都是全美前10学校的博士

6. 课程会有考试吗

回答：有的。一般包括期中和期末

7. 我只有编程基础，可以报名吗

回答：对于初级的项目班只要求编程基础

8. 课程有实操吗

回答：大部分都是实操，动手能力是最重要的

9. 课程为什么贵？

回答：跟别的知识付费不一样，我们会提供很多教学服务，辅助完成学员做完所有的项目

10. 课程学完了能做什么？

回答：可以找相关岗位的工作问题不大

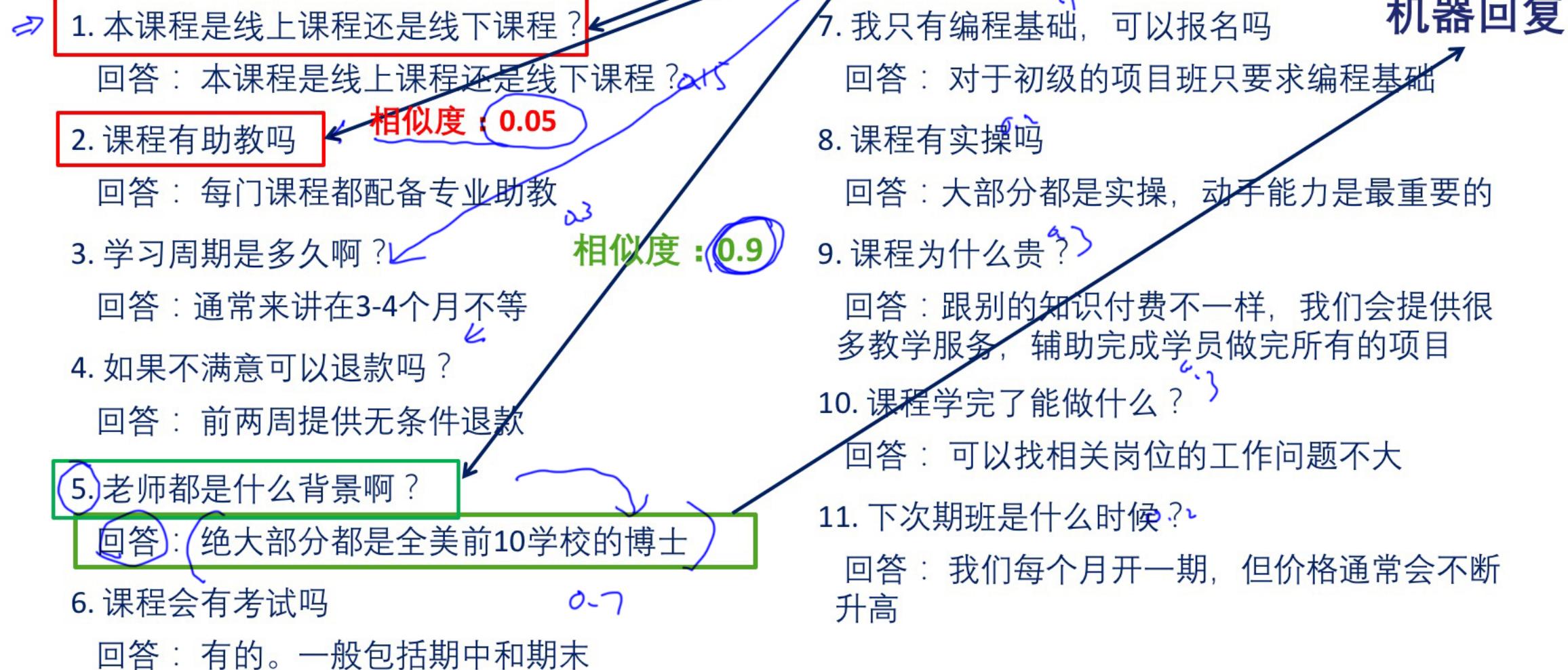
11. 下次期班是什么时候？

回答：我们每个月开一期，但价格通常会不断升高

正向/规则 → 没有
抢先手 → 训练数据
常见的问题 (FAQ) :

案例：搭建一个智能客服系统

相似度  ① 正向匹配
② 字符串相似



基础



进阶
→ 英文
→ 中文

a. a. the

<ATAMP>
<Html>
<1:>



[基础]
进阶 → 高阶
↑
+x

基于搜索的问答系统

- ✓
- (Segmentation)
- How do you like NLPCamp? → 分词 → 问题型
- ① spell correction
 - ② Stemming/Lemmatization
 - ③ Stop-words
 - ④ Words filtering
 - ⑤ 同义词

返回相似度最高的

进阶表示
进阶 → 高阶

- ① boolean vector (0,1,0,1...)
- ② Count vector (1,2...0-)
- ③ tf-idf (0.7,0.3...0.0)
- ④ Word2Vec (0.13,0.3,0.7...)
- ⑤ Squeeze (0.15,0.25...)

Inverted
Index
O(n) >> O(1)

计算相似度
(给定两个向量)

- ① Euclidean Distance
- ② Cosine Distance
- ③ Jaccard Similarity

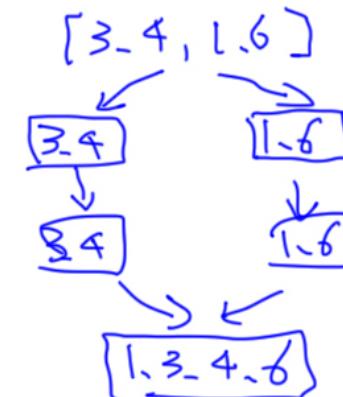
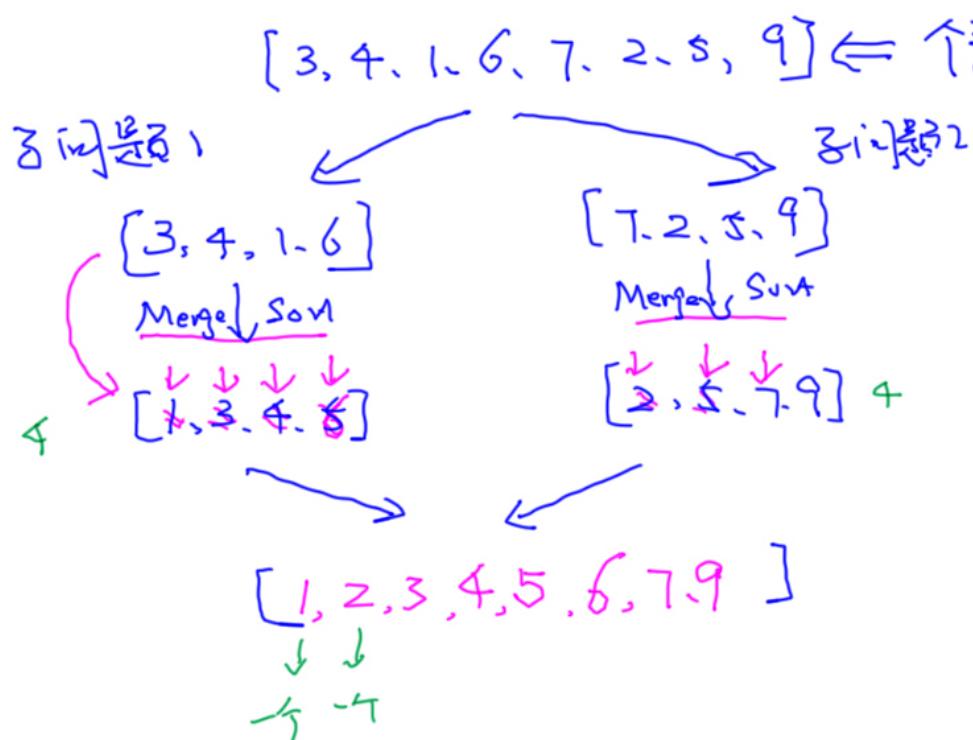


- Sorting: Merge Sort (归并)

- Divide and Conquer (Category)

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

目标: Sort(A)

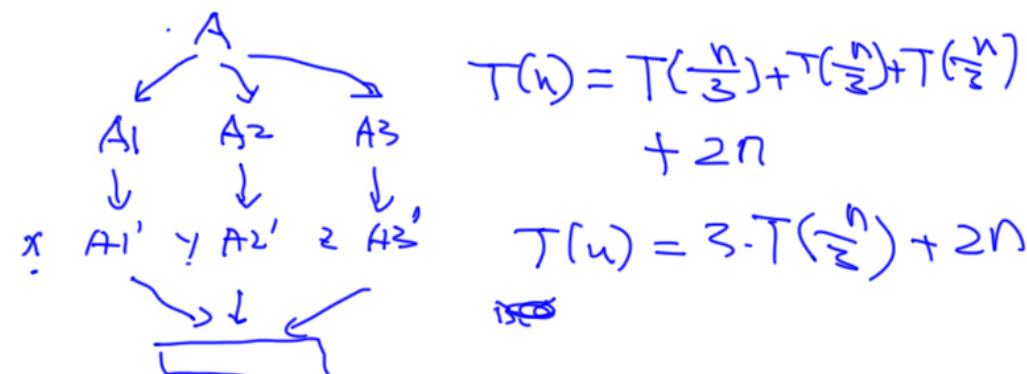


$$\boxed{T(n) = 2 \cdot T\left(\frac{n}{2}\right) + n}$$
$$T(n) = O(n^2)$$
$$= O(n \lg n)$$

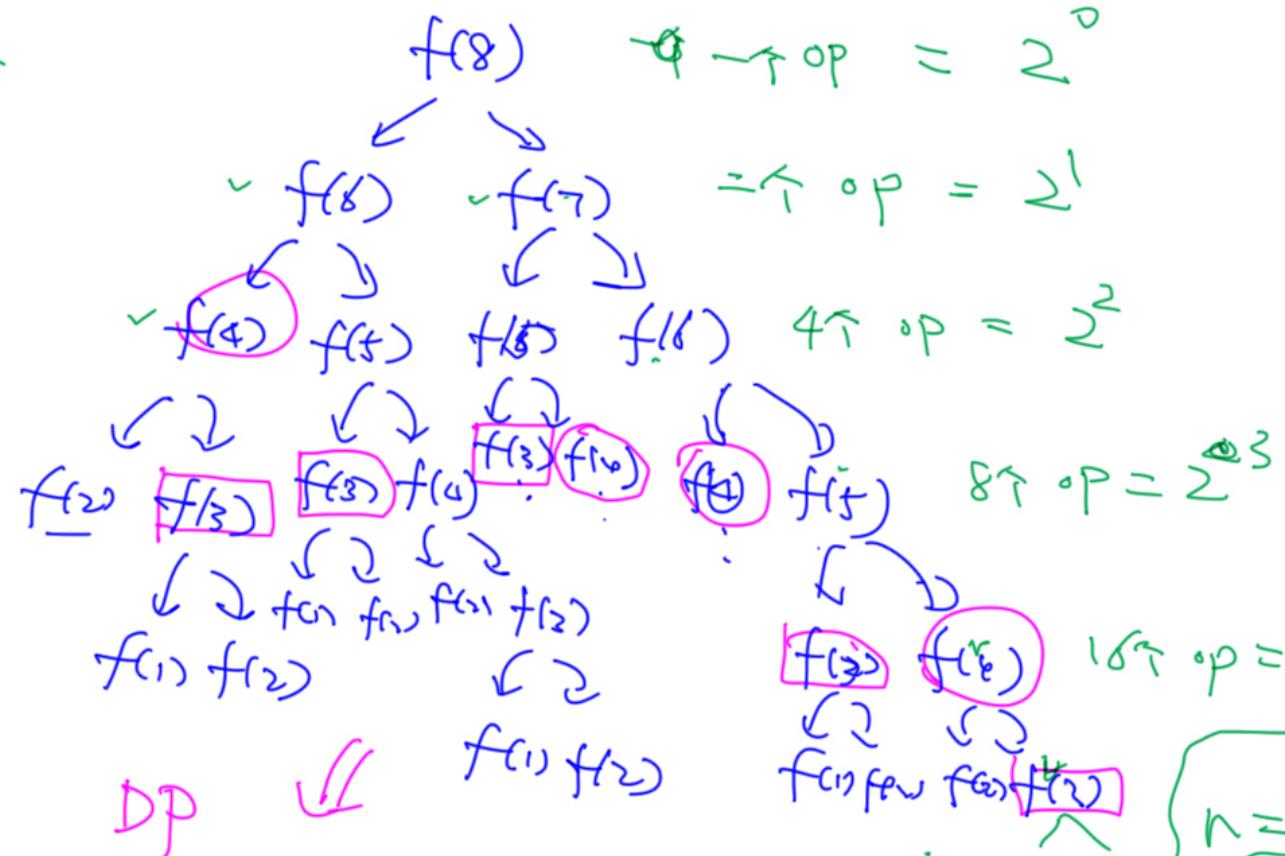
3个问题

$T(n) = T\left(\frac{n}{3}\right) + T\left(\frac{n}{3}\right) + n$

if divide into 3个子问题



$$\begin{aligned} f(n) &= f(n-2) + f(n-1) \Rightarrow f(8) \\ f(1) = f(2) &= 1 \Leftarrow \text{base case} \end{aligned}$$



$n = 8 \Rightarrow h = 6$

$\Theta(\text{Time}) = O(n^2) \quad O(n \lg n)$

$? f(n) = 2 \cdot f\left(\frac{n}{2}\right) + n$

$2^0 + 2^1 + 2^2 + \dots + 2^6 = 2^{7-1}$

$\Theta(2^h) \quad 2^{n-1}$

$\Theta(2^{n-1}) \cdot 2$

$\Theta(2^n)$

$n = h = 8$

$n = hT$

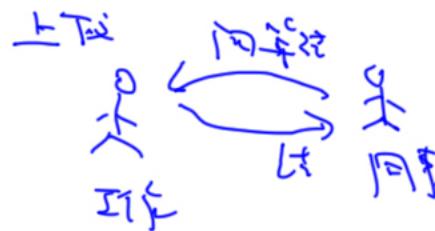
$h = O(n)$

$$\begin{aligned}f(n) &= f(n-2) + f(n-1) \\f(1) &= f(2) = 1\end{aligned}$$

Space complexity?

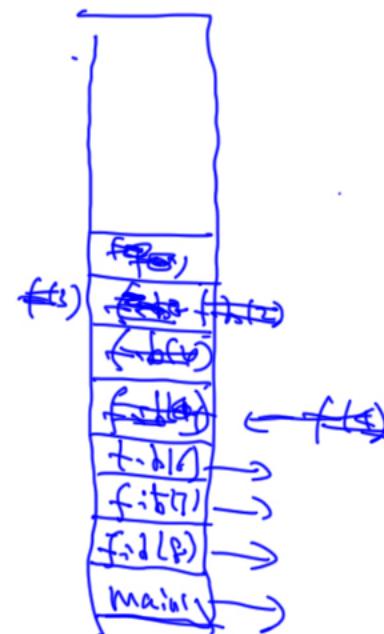
$$\left. \begin{array}{l} f_1 \\ f_2 \\ f_3 \end{array} \right\} \Rightarrow \text{上层之加法}$$

$$\begin{array}{l} f_1 \\ f_2 \\ f_3 \end{array}$$



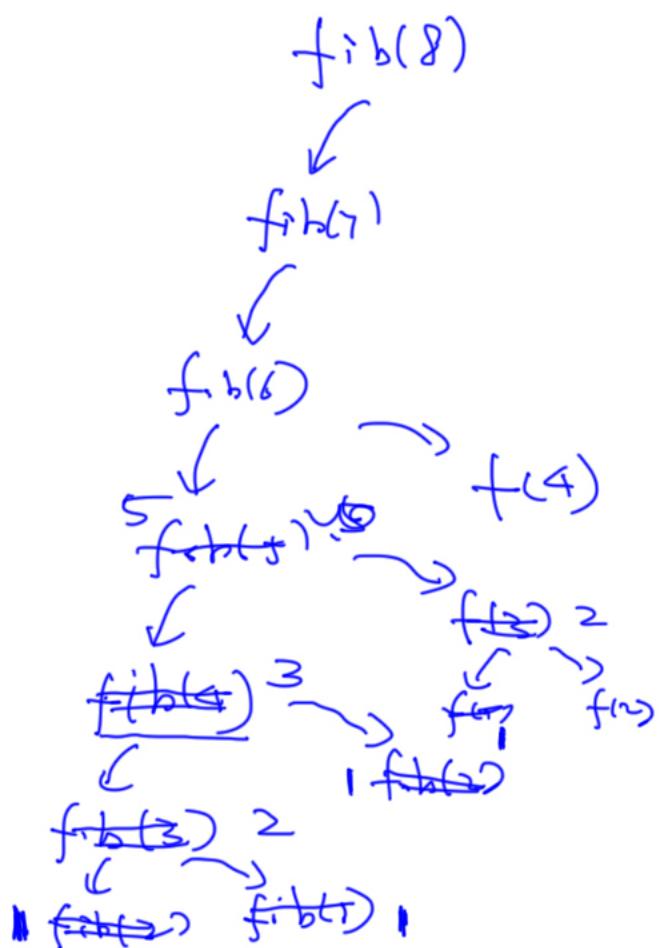
递归
↑
递归
↓

① 递归 ② stack



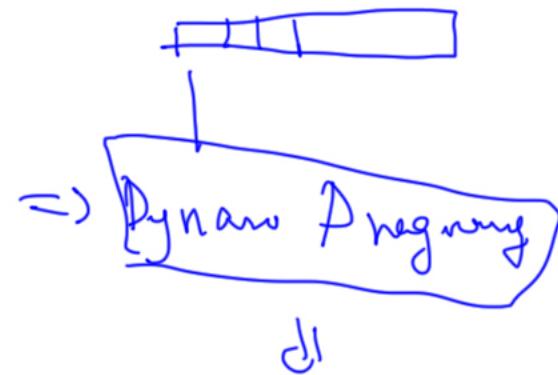
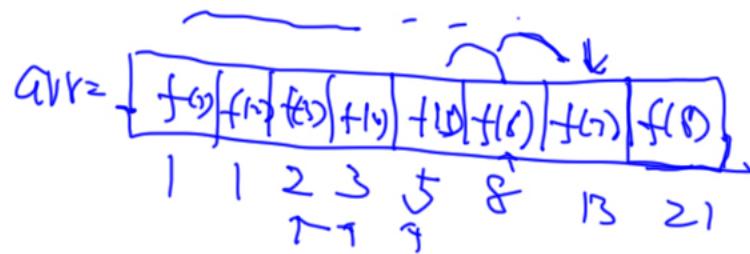
8个内存空间

③



stack

$f(8)$



$$n! \approx \mathbb{R}^n$$

②

def $f(n)$:

 spell correction

 return $f(n-2) + f(n-1)$

问答系统

- 现存方法
 实体表示
 相似度

- 基础句语
 实体抽取
 关系抽取

