NLP Bootcamp (2)

2019年01月22日

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

Master Theorem

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

$a = 2,\ b = 2$

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

$f(n) = n = n \lg^0 n$

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

$= O(n \log n)$

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

$a = 2,\ b = 2,\ f(n) = n = \Theta(n)$

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[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.

① $O(n^{\log_b a}) > f(n) \Rightarrow O(n^{\log_b a})$

② $O(n^{\log_b a}) = f(n) \Rightarrow O(n^{\log_b a} \log^k n)$

③ $O(n^{\log_b a}) < f(n) \Rightarrow f(n)$

$a = 3,\ b = 2$

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

$n^{\log_b a} = n^{\log_2 3} = n^{1.6}$

$f(n) = n^2$

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

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

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

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

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

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

∴ case 2 $f(n) = n^2 = n^2 \lg^0 n$

$T(n) = O(n^2 \cdot \log^{k+1} n)$

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

Master Theorem

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.

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

$a = 16,\ b = 4$

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

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

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

$a = 2,\ b = 4$

$n^{\log_b a} = n^{0.5} < n^{0.51}$

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

Master Theorem

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.

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

$a=16,\ b=4$

$n^{\log_b a} = n^2 < n!$

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

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

Exercise?

Master Theorem

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

The Master Theorem applies to recurrences of the following form:

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

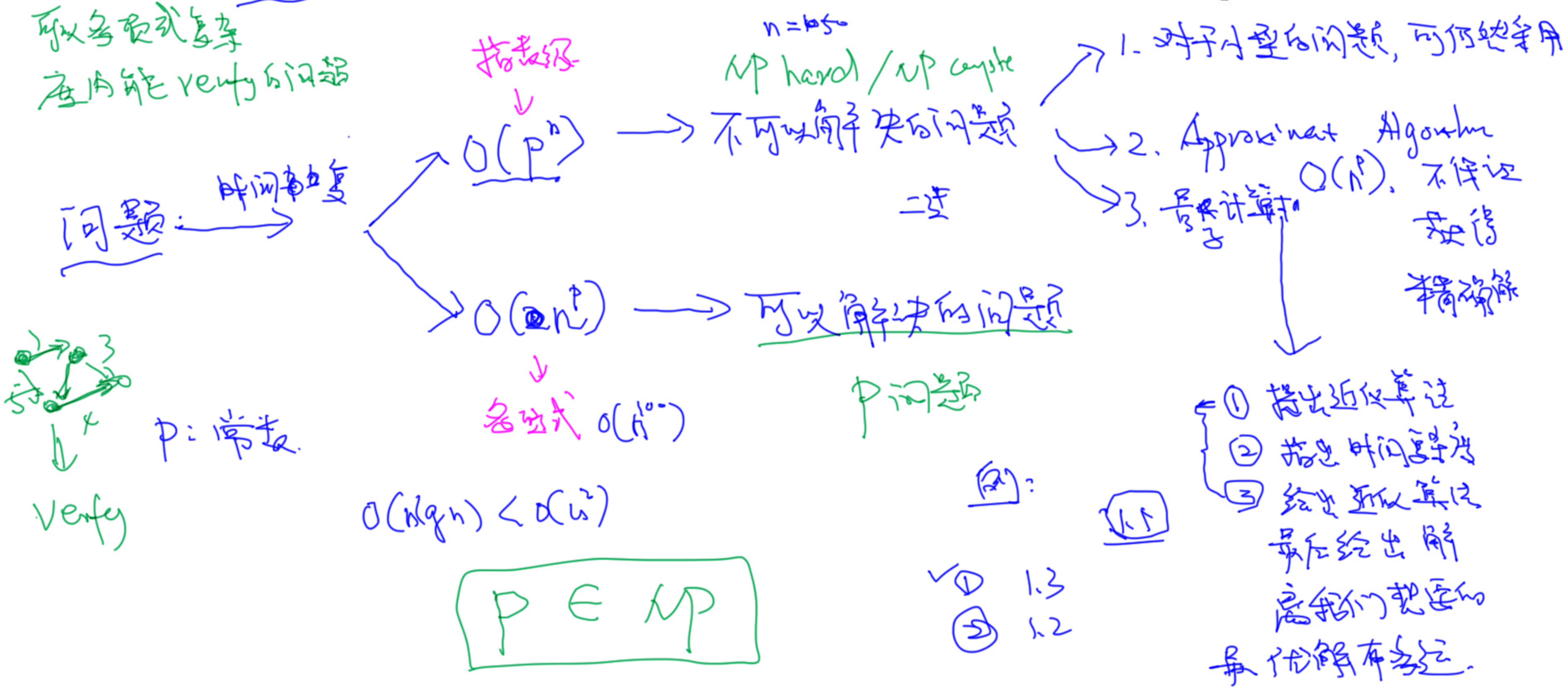
where $a \geq 1$ and $b > 1$ are constants and $f(n)$ is an asymptotically positive function.

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

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

P vs NP vs NP Hard vs NP Complete



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

可不可以用这些语料库来搭建一个智能客服系统？

案例：搭建一个智能客服系统

常见的问题（FAQ）：

用户输入：“我想了解老师的背景”

相似度：0.1

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？

相似度：0.05

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

相似度：0.9

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？

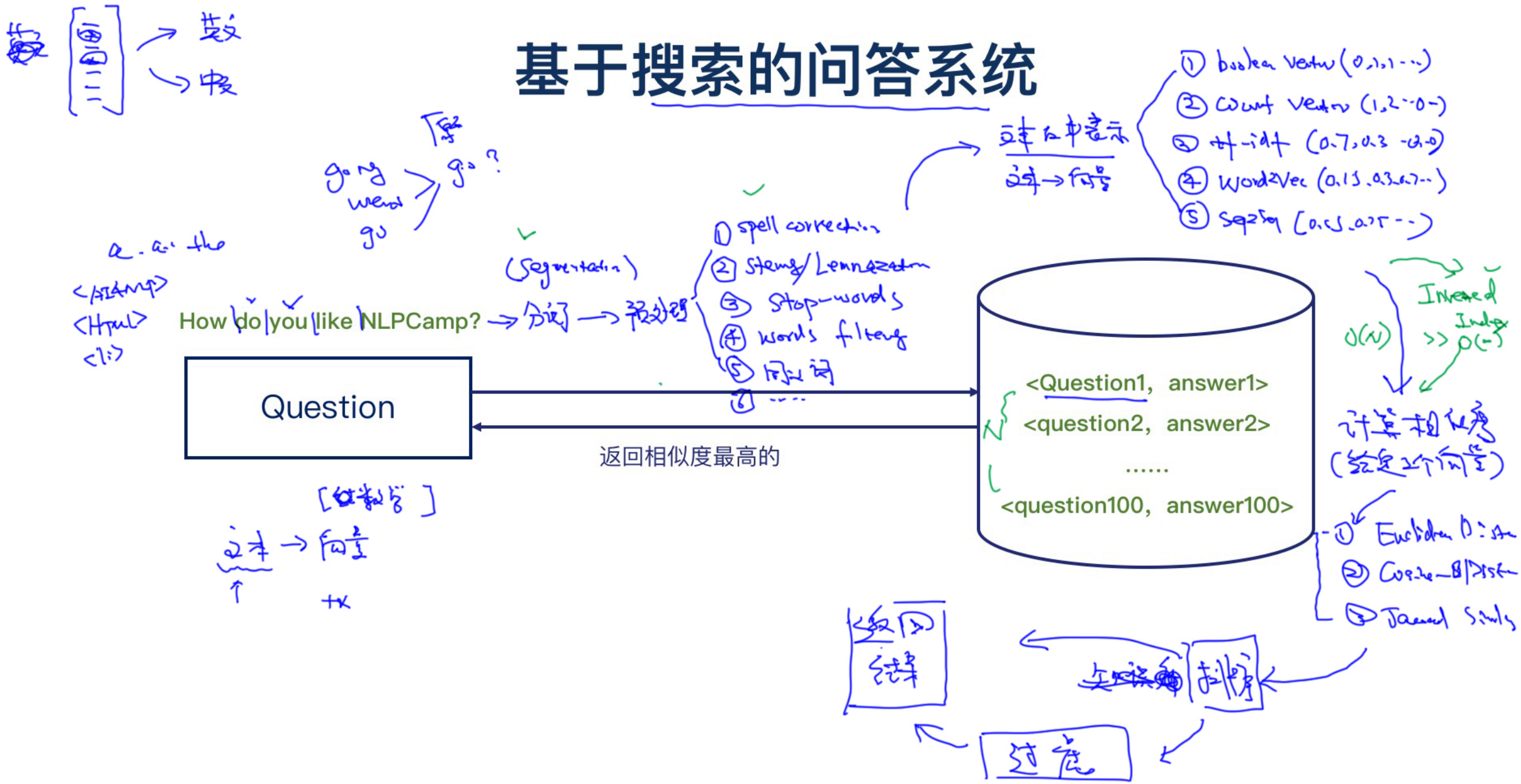
回答：可以找相关岗位的工作问题不大

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

机器回复

基于搜索的问答系统

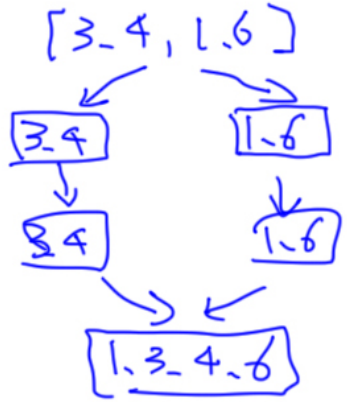

How do you like NLPCamp?
Question
返回相似度最高的
<Question1, answer1>
<question2, answer2>
.....
<question100, answer100>
spell correction
Stemming/Lemmazation
Stop-words
words filtering
Segmentation
boolean vector
count vector
tf-idf
Word2Vec
Sqzsq
Inverted Index
O(N)
>> O(1)
Euclidean Distance
Cosine Similarity
Jaccard Similarity

Sorting: Merge Sort (归并)

- Divide and Conquer (Category)

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

目标: Sort(A)



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

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

$= O(n \lg n)$

[3, 4, 1, 6, 7, 2, 5, 9] ⇐ 个数: n

3个问题1 | 3个问题2

[3, 4, 1, 6] | [7, 2, 5, 9]

Merge Sort | Merge Sort

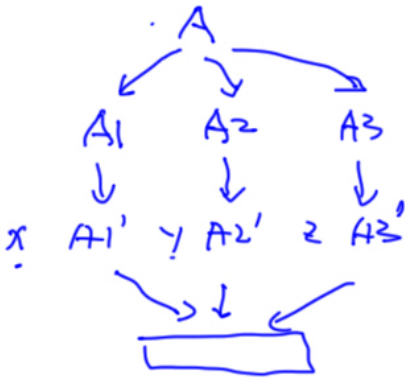
[1, 3, 4, 6] | [2, 5, 7, 9]

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

1个问题 | 3个问题1 | 3个问题2

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

if divide into 3个子问题

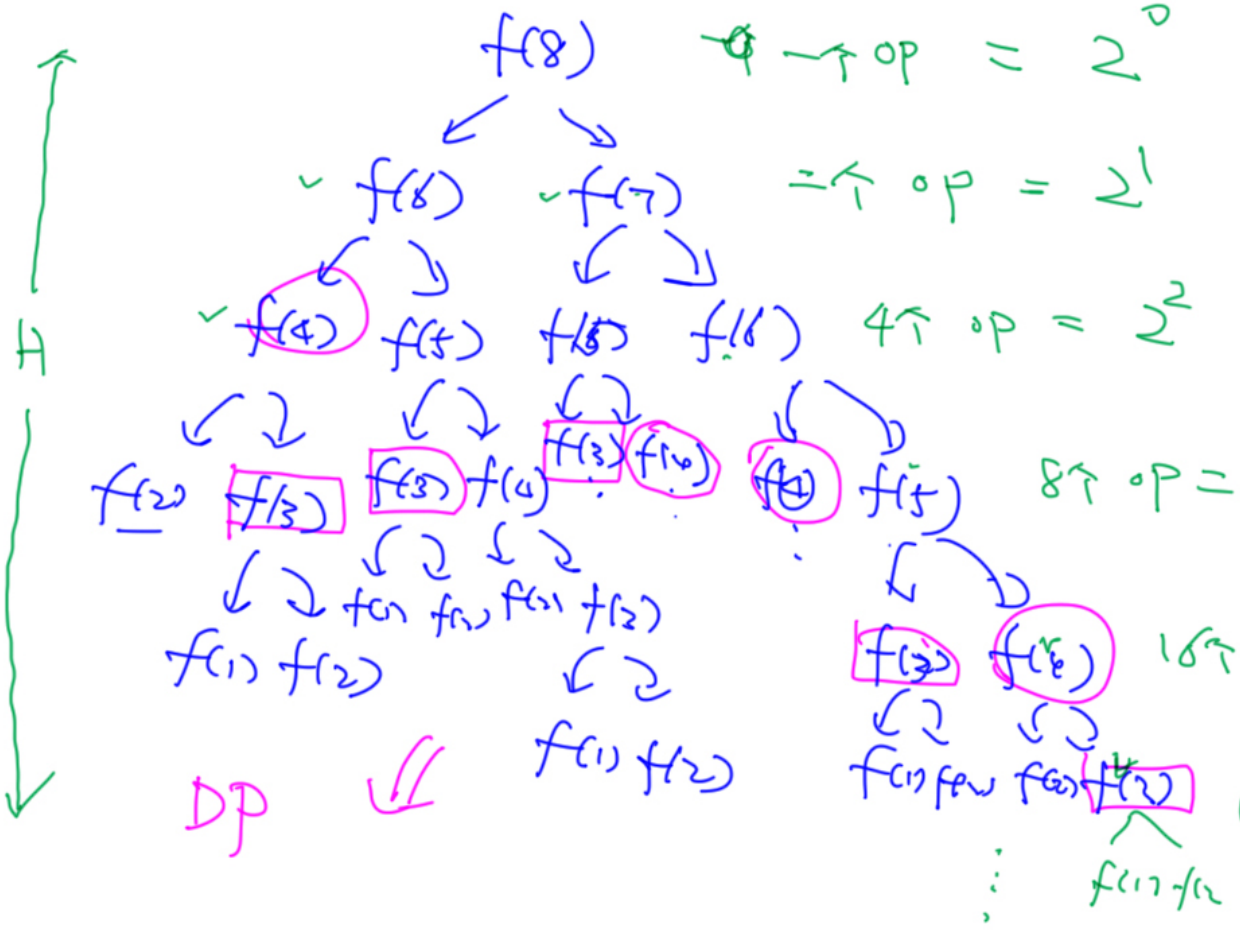


$T(n) = T(\frac{n}{3}) + T(\frac{n}{3}) + T(\frac{n}{3}) + 2n$

$T(n) = 3 \cdot T(\frac{n}{3}) + 2n$

$f(n) = f(n-2) + f(n-1) \Rightarrow f(8)$

$f(1) = f(2) = 1 \Leftarrow$ base case



1个 op = 2^0

2个 op = 2^1

4个 op = 2^2

8个 op = 2^3

16个 op = 2^4

$n = 8 \Rightarrow h = 6$

$n = h = 8$

$n = h^T$

$h = O(n)$

时间：$O(n^2)$ $O(n \lg n)$

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

$2^0 + 2^1 + 2^2 \cdots + 2^h = 2^{T+1}$

$T(n) = O(n)$

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

$O(2^h)$ 2^{n-1}

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

$= O(2^n)$

$O(2^n)$

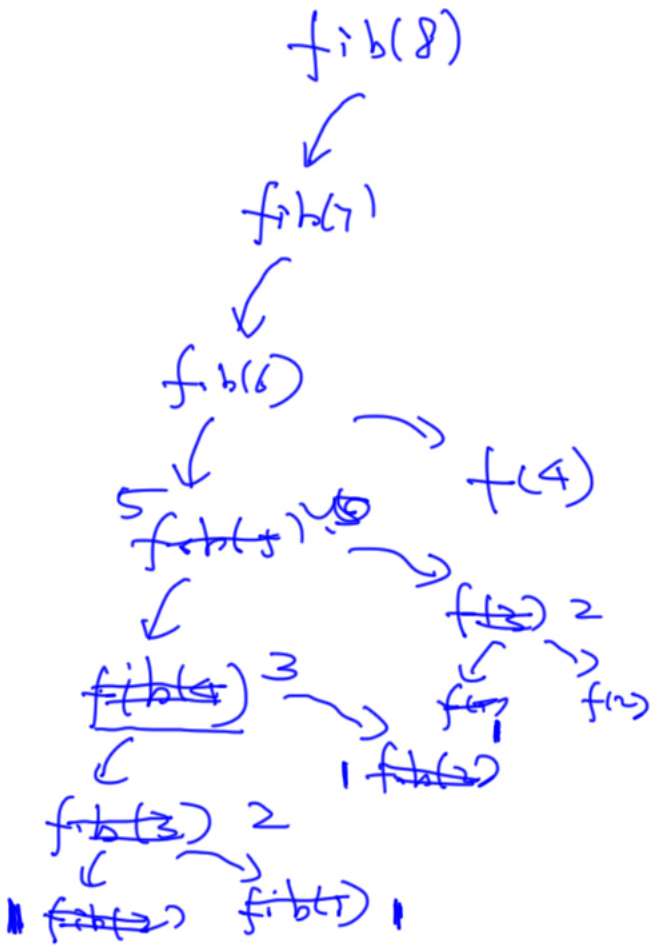
$O(2^n)$

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

$f(1) = f(2) = 1$

Space complexity?

```
def main():
    fib(8)
```



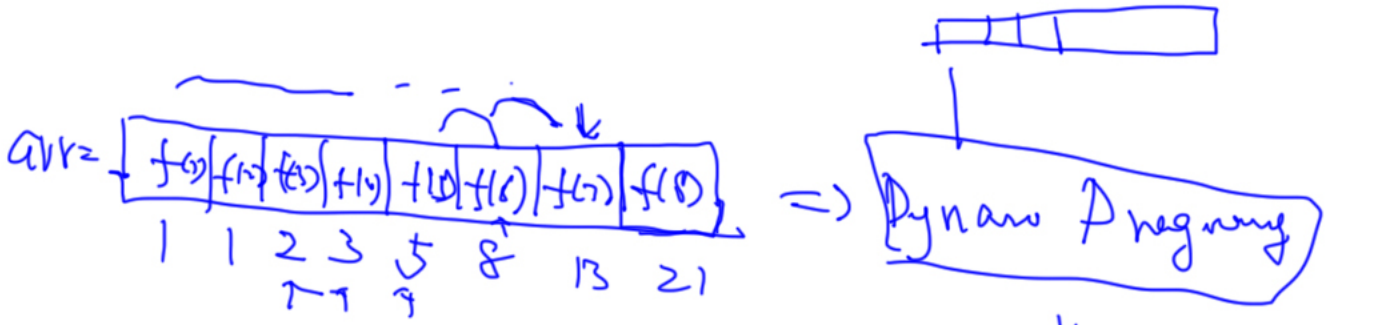
O(n)

f(8)

8个内存空间

Stack

f(8)



=> Dynam Progrmg

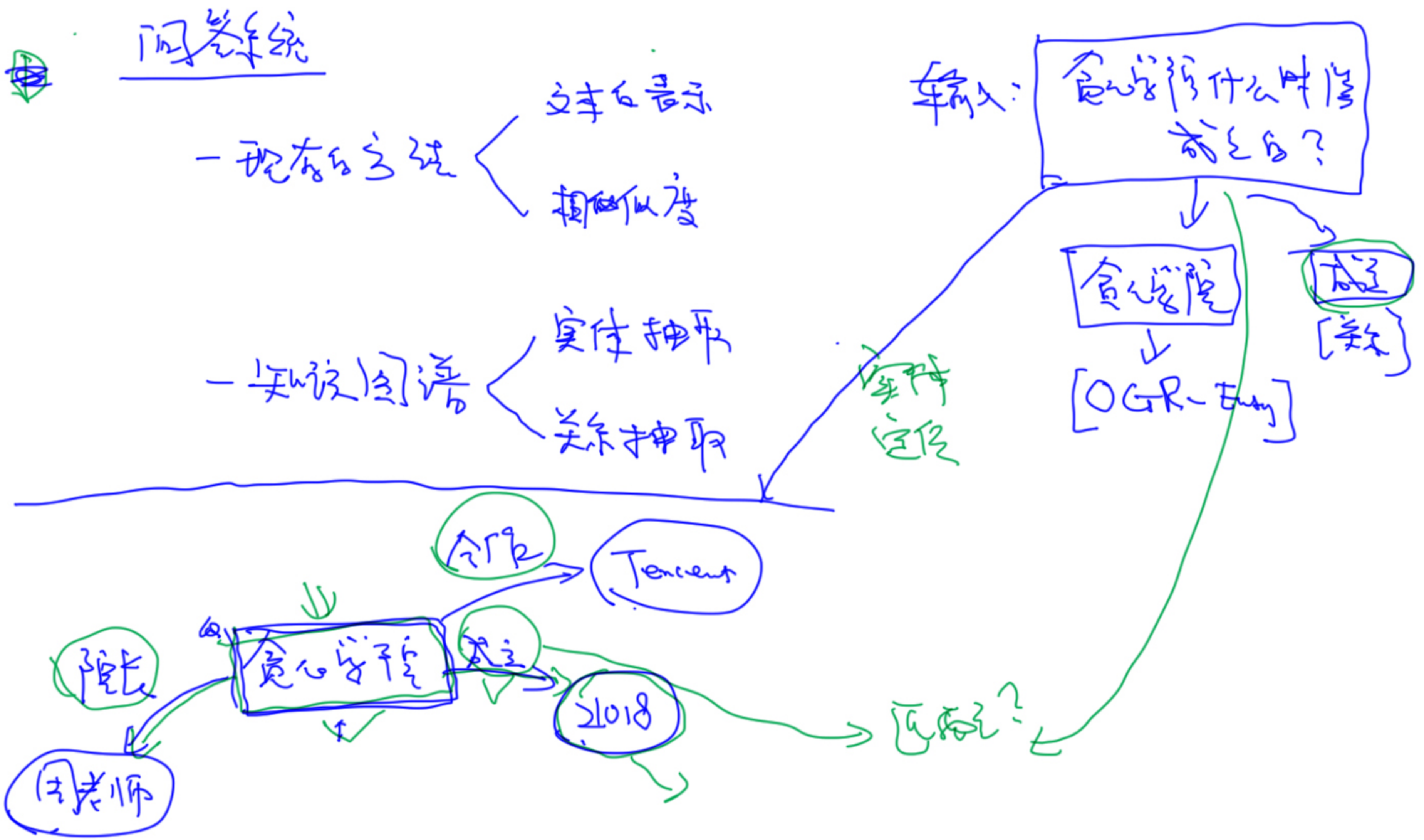
dl

Spell correction

$n! \approx n^n$

@ def f(n):

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


问答系统
一 检索式方法
文本的表示
相似度
一 知识图谱
实体抽取
关系抽取
输入:
贪心学院什么时候成立的?
贪心学院
成立
实体定位
合作
Tencent
贪心学院
院长
成立
2018