Merge Sort (归并排序) — Divide and Conquer

8个 $n=8$

输入：[3,4,1,6,7,2,5,9]　　输出：[1,2,3,4,5,6,7,9]

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

Divide

[3, 4, 1, 6]　　[7, 2, 5, 9]

归并排序　　归并排序

[1, 3, 4, 6]　　[2, 5, 7, 9]

合并 (conquer)

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

array

```
Merge_Sort(a):
    if len(a) == 1: return a
    l1 = a[0] ... a[n/2]
    l2 = a[n/2+1 ... n]
    l1 = Merge_Sort(l1)
    l2 = Merge_Sort(l2)
    return Merge(l1, l2)
```

$$T(n) = c_1 + T\left(\frac{n}{2}\right) + T\left(\frac{n}{2}\right) + n$$

原　Divide　l_1　l_2　合并

Merge Sort (归并排序)

$$T(n) = \underline{a} + 2T(\tfrac{n}{2}) + \underline{n}$$

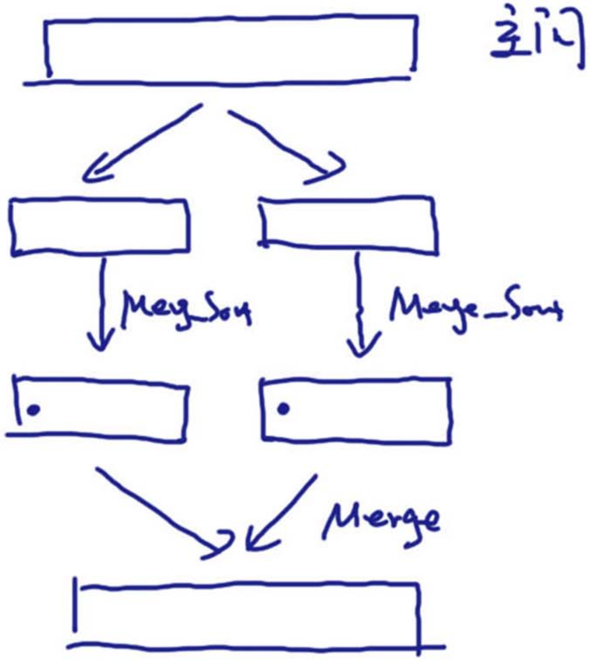
$$= 2T(\tfrac{n}{2}) + n$$

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

$$T(n) = ?$$

主定理 (Master Theorem)

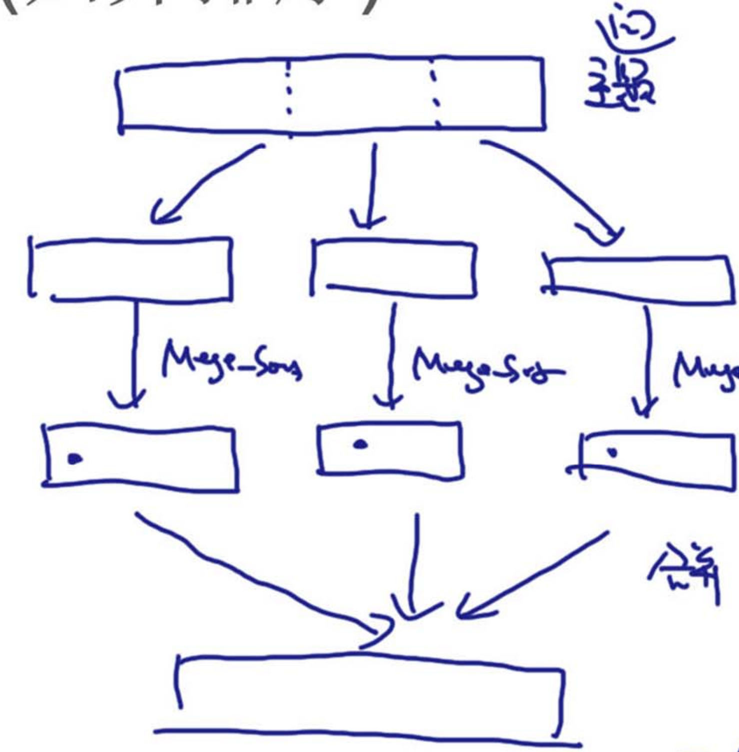
Merge Sort (归并排序)



$f(n) = n$

2块(两部分)

? $\geq$ $\leq$ $=$



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

$n^{\log_b a} = n^{\log_3 3} = n \Rightarrow ②$

$T(n) = O(n \log n)$

Master Theorem

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 $\boxed{f(n)} = O(\boxed{n^{\log_b a}}^{-\epsilon})$ 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 $O(n^{\log_b a}) > f(n) \Rightarrow T(n) = O(n^{\log_b a})$

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

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

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

$n^{\log_b a} = n^{\log_2 3} < n^2$

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

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

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

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

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

$f(n) = n^2 = n^2 \cdot \log^0 n \Rightarrow k=0$

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

$= O(n^2 \cdot \log n)$

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.

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

$$n^{\log_b a} = n^{\log_4 16} = n^2 > n \Rightarrow 1$$

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

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

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

$$3 \Leftarrow = f(n)$$

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

Master Theorem

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

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.

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

Master Theorem

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

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.

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

Merge Sort (归并排序)

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.

$$f(n) = n \cdot \log^{0=k} n$$

$$\Uparrow$$

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

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

$$\Downarrow$$

②

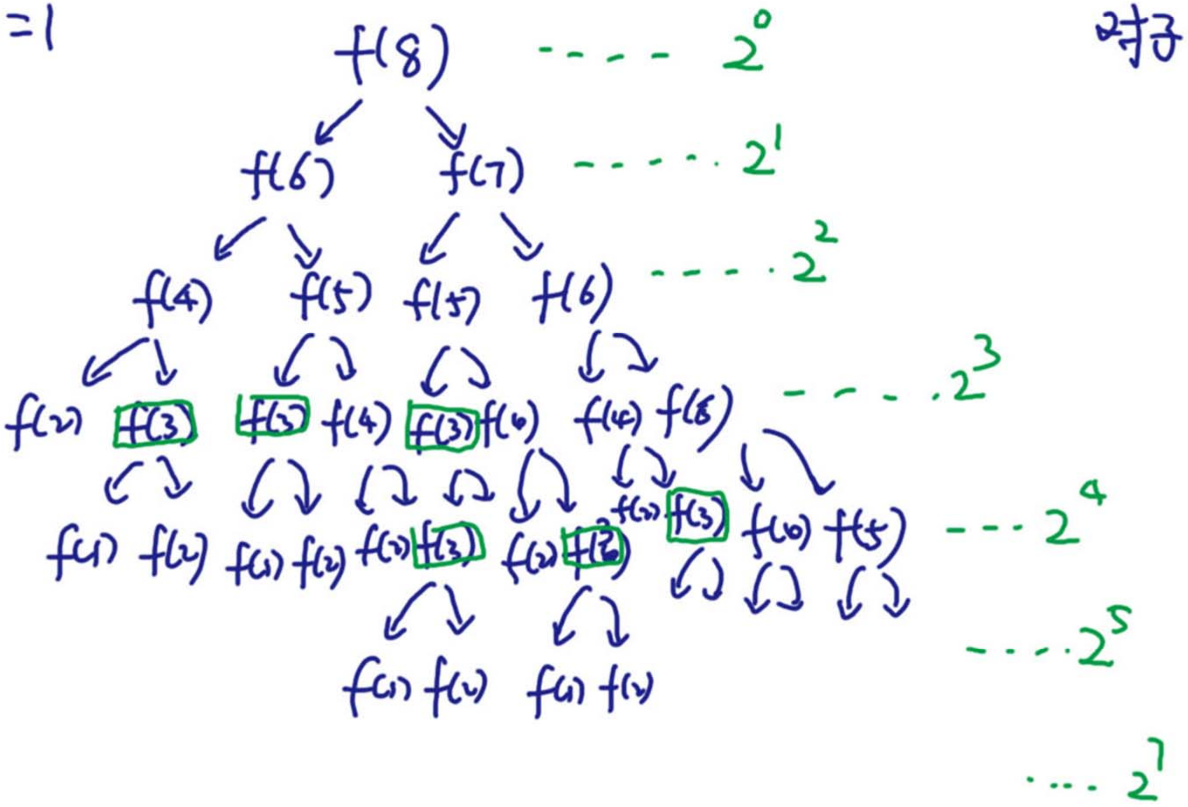
$$T(n) = O(n^{\log_b a} \cdot \log^{k+1})$$
$$= O(n \cdot \log n)$$

Compute Time Complexity for Recurrence

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

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

i.e. $f(8)$



$2^0 + 2^1 + 2^2 + \cdots + 2^7 = 2^8 - 1$

对于 n, f(n)

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

$\downarrow$

指数级复杂度 $O(p^n)$

多项式复杂度 $O(n^p)$

(polynomial)

非 NP hard

Compute Space Complexity for Recurrence

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

上下文切换.

```
def main():
  → f(8)
```

$\longrightarrow$ 21

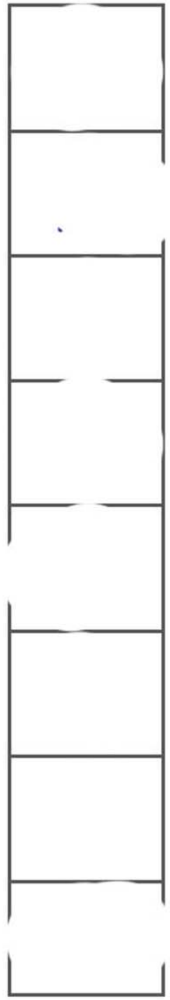
$f(1)=f(2)=1$

F(8)

O(n)

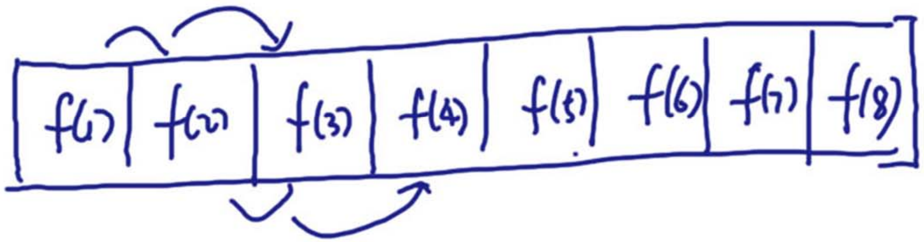
↑

线性的复杂度



Stack.

Iterative Process for Fibonacci Number



array

$f(8) \leftarrow f(7) \leftarrow f(6)$

$\quad \nwarrow f(6) \quad \nwarrow f(5)$

时间： $O(n)$

空间： $O(n)$

Closed Form Solution for Fibonacci Number

$f(n) = \boxed{}$

$\boxed{f(n) = f(n-1) + f(n-2)}$

$$f(n) = \frac{1}{\sqrt{5}}\left[\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n\right]$$

$\Leftarrow$ closed form solution

思考题 ?

Remarks on Complexity

O notation :

$O(1) < O(\log n) < O(n) < O(n\lg n) < O(n^2) < O(n^2 \lg n)$

$< O(n^3) < O(n!)$

constant time

线性

$\Rightarrow$ 指数级复杂度 $O(p^n)$

多项式复杂度 $O(n^p) \leftarrow$

P NP. NP complete, NP hard

$\boxed{\text{Computational Complexity Theory}}$

案例：搭建一个智能客服系统

常见的问题（FAQ）：

语料库

1. 本课程是线上课程还是线下课程？ 问题

   回答：线上课程为主 答案

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

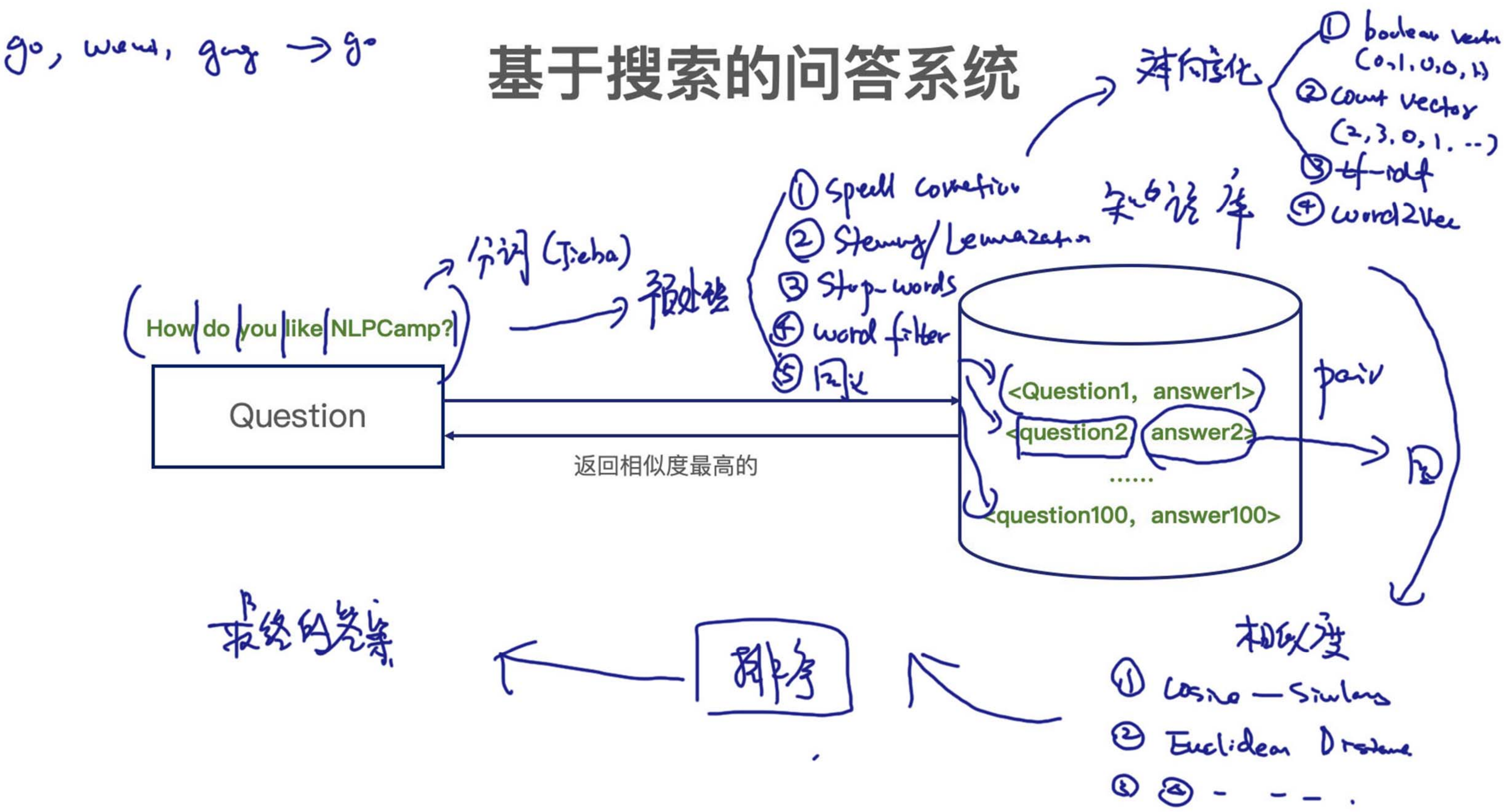
相似度：0.05

相似度：0.9

机器回复

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

基于搜索的问答系统



go, went, going → go
词向量化
① boolean vector (0,1,0,0,1)
② count vector (2,3,0,1,...)
③ tf-idf
④ word2vec
分词 (Jieba)
预处理
① Spell correction
② Stemming/Lemmazation
③ Stop-words
④ word filter
How do you like NLPCamp?
Question
返回相似度最高的
<Question1，answer1>
<question2，answer2>
......
<question100，answer100>
pair
相似度
① Cosine-Similarity
② Euclidean Distance
排序
最终的答案