# 前(正)向最大匹配算法

## 分词目标

在词典中进行扫描，尽可能地选择与词典中最长单词匹配的词作为目标分词，然后进行下一次匹配。

## **算法流程**

假设词典中最长的单词为 5 个（MAX\_LENGTH），那么最大匹配的起始子串字数也为 5 个：

1. 扫描字典，测试读入的子串是否在字典中；
2. 如果存在，则从输入中删除掉该子串，重新按照规则取子串，重复（1）；
3. 如果不存在于字典中，则从右向左减少子串长度，重复（1）。

## **分词实例**

比如说输入 “北京大学生前来应聘”：

1. 取子串 “北京大学生”，正向取词，如果匹配失败，每次去掉匹配字段最后面的一个字
2. “北京大学生”，扫描 5 字词典，没有匹配，子串长度减 1 变为“北京大学”；
3. “北京大学”，扫描 4 字词典，有匹配，输出“北京大学”，输入变为“生前来应聘”。
4. 取子串“生前来应聘”
5. “生前来应聘”，扫描 5 字词典，没有匹配，子串长度减 1 变为“生前来应”；
6. “生前来应”，扫描 4 字词典，没有匹配，子串长度减 1 变为“生前来”；
7. “生前来”，扫描 3 字词典，没有匹配，子串长度减 1 变为“生前”；
8. “生前”，扫描 2 字词典，有匹配，输出“生前”，输入变为“来应聘”。
9. 取子串“来应聘”
10. “来应聘”，扫描 3 字词典，没有匹配，子串长度减 1 变为“来应”；
11. “来应”，扫描 2 字词典，没有匹配，子串长度减 1 变为“来”；
12. 颗粒度最小为 1，直接输出“来”，输入变为“应聘”。
13. 取子串“应聘”
14. “应聘”，扫描 2 字词典，有匹配，输出“应聘”，输入变为“”。
15. 输入长度为0，扫描终止。

正向匹配法最终的切分结果为：”北京大学 / 生前 / 来 / 应聘”。

# 逆(后)向最大匹配法

## **分词目标**

在词典中进行扫描，尽可能地选择与词典中最长单词匹配的词作为目标分词，然后进行下一次匹配。**在实践中，逆向最大匹配算法性能优于正向最大匹配算法**。

## **算法流程**

假设词典中最长的单词为 5 个（MAX\_LENGTH），那么最大匹配的起始子串字数也为 5 个

1. 扫描字典，测试读入的子串是否在字典中；
2. 如果存在，则从输入中删除掉该子串，重新按照规则取子串，重复（1）；
3. 如果不存在于字典中，则从左向右减少子串长度，重复（1）。

## **分词实例**

比如说输入 “北京大学生前来应聘”：

1. 取子串 “生前来应聘”，逆向取词，如果匹配失败，每次去掉匹配字段最前面的一个字
2. “生前来应聘”，扫描 5 字词典，没有匹配，字串长度减 1 变为“前来应聘”；
3. “前来应聘”，扫描 4 字词典，没有匹配，字串长度减 1 变为“来应聘”；
4. “来应聘”，扫描 3 字词典，没有匹配，字串长度减 1 变为“应聘”；
5. “应聘”，扫描 2 字词典，有匹配，输出“应聘”，输入变为“大学生前来”。
6. 取子串“大学生前来”
7. “大学生前来”，扫描 5 字词典，没有匹配，字串长度减 1 变为“学生前来”
8. “学生前来”，扫描 4 字词典，没有匹配，字串长度减 1 变为“生前来”
9. “生前来”，扫描 3 字词典，没有匹配，字串长度减 1 变为“前来”
10. “前来”，扫描 2 字词典，有匹配，输出“前来”，输入变为“北京大学生”
11. 取子串“北京大学生”
12. “北京大学生”，扫描 5 字词典，没有匹配，字串长度减 1 变为“京大学生”
13. “京大学生”，扫描 4 字词典，没有匹配，字串长度减 1 变为“大学生”
14. “大学生”，扫描 3 字词典，有匹配，输出“大学生”，输入变为“北京”
15. 取子串“北京”
16. “北京”，扫描 2 字词典，有匹配，输出“北京”，输入变为“”
17. 输入长度为0，扫描终止

逆向匹配法最终的切分结果为：”北京/ 大学生/ 前来 / 应聘”

# 双向最大匹配法

## **分词目标**

将正向最大匹配算法和逆向最大匹配算法进行比较，从而确定正确的分词方法。

## **算法流程**

1. 比较正向最大匹配和逆向最大匹配结果
2. 如果分词数量结果不同，那么取分词数量较少的那个
3. 如果分词数量结果相同
4. 分词结果相同，可以返回任何一个
5. 分词结果不同，返回单字数比较少的那个

## **分词实例**

正向匹配最终切分结果为：北京大学 / 生前 / 来 / 应聘，分词数量为 4，单字数为 1

逆向匹配最终切分结果为：”北京/ 大学生/ 前来 / 应聘，分词数量为 4，单字数为 0

逆向匹配单字数少，因此返回逆向匹配的结果。