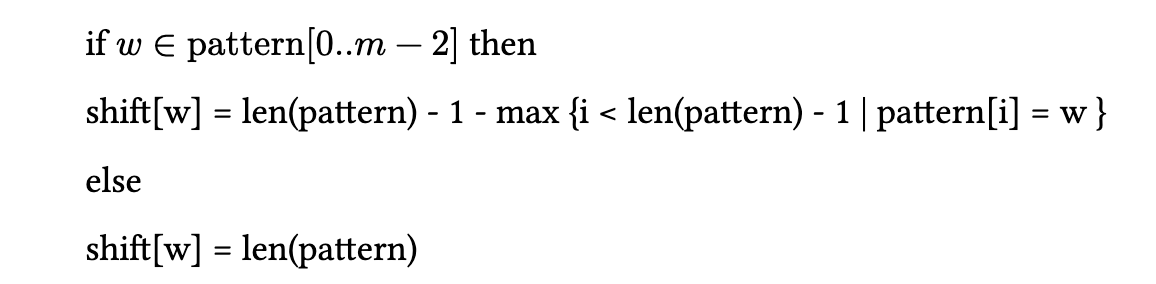
**算法设计与分析第九章作业**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **姓名** | **王靳­** | **班级** | **计科十班** | **学号** | 220111012 |
| **第1题** |  | | | | |
| **第2题** |  | | | | |
| **第3题** |  | | | | |
| **总分** |  | | | | |
| **备注** | 作业提交截止时间： 2023年11 月22 日24:00，超过提交截至时间的作业视为无效。确因网络等特殊原因无法及时提交作业的学生，应至少提前1小时与助教联系沟通。作业提交邮箱：hitsz\_algorithm@126.com。作业文件名命名方式： 第x章-x班-姓名-学号（例，第1章-1班-张三-220110101.docx）； 邮件主题为：第x章作业, x班，姓名，学号（例，第1章作业，1班，张三，220110101）。缺少这些信息的作业将被酌情扣分。 | | | | |

1. **（25分）**根据KMP算法中pi数组的计算方法，求串S= ‘babab ' 的pi 数值序列。

Pi = [0, 0, 0, 1, 2, 3, 0]

1. **（35分）**
2. 写出BMH算法的偏移表计算公式。



失配的字符是match[w]，那么shift[w] = len(pattern) – 1 – (pattern[0:m-1]最右边的下标)，否则shift[w]就是偏移整个字符串的长度

1. 如下表格是BMH算法的匹配过程，请写出第2到6行的内容。

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| i | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
| T | a | b | d | a | c | b | a | c | d | b | c | a | c | a | b | c | a | c |
| 1 | a | b | c | a | c |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 2 |  |  | a | b | c | a | c |  |  |  |  |  |  |  |  |  |  |  |
| 3 |  |  |  | a | b | c | a | c |  |  |  |  |  |  |  |  |  |  |
| 4 |  |  |  |  |  | a | b | c | a | c |  |  |  |  |  |  |  |  |
| 5 |  |  |  |  |  |  |  |  | a | b | c | a | c |  |  |  |  |  |
| 6 |  |  |  |  |  |  |  |  |  |  | a | b | c | a | c |  |  |  |
| 7 |  |  |  |  |  |  |  |  |  |  |  |  |  | a | b | c | a | c |

1. **（40分）**给你两个字符串 haystack 和 needle ，请你在 haystack 字符串中找出 needle 字符串的第一个匹配项的下标（下标从 0 开始）。如果 needle 不是 haystack 的一部分，则返回  -1 。( haystack 和 needle 仅由小写英文字符组成)

示例 1：

输入：haystack = "sadbutsad", needle = "sad"

输出：0

解释："sad" 在下标 0 和 6 处匹配。

第一个匹配项的下标是 0 ，所以返回 0 。

示例 2：

输入：haystack = "asdasd", needle = "sad"

输出：-1

解释："sad" 没有在 "asdasd" 中出现，

所以返回 -1 。

1. def compute\_prefix(pattern: str):
2. m: int = len(pattern)
3. pi: list = [0] \* (m + 1)  *# pi[0..m]*
4. *# 第m位置是虚拟位置*
5. for q in range(2, m + 1):  *# 第q个位置失配*
6. k = pi[q - 1]
7. while k > 0 and pattern[k] != pattern[q - 1]:  *# 找到最长的前后缀的, 前缀位置*
8. k = pi[k]
9. if pattern[k] == pattern[q - 1]:
10. k = k + 1
11. pi[q] = k
12. *# print(pi, len(pi))*
13. return pi
14. def search(pattern: str, match: str):
15. m: int = len(pattern)
16. n: int = len(match)
17. arr: list = []
18. pi = compute\_prefix(pattern)
19. q = 0
20. for i in range(0, n):
21. while q > 0 and pattern[q] != match[i]:
22. q = pi[q]
23. if pattern[q] == match[i]:
24. q = q + 1
25. if q == m:  *# 匹配成功*
26. arr.append(i - m + 1)
27. q = pi[q]
28. return arr
29. if \_\_name\_\_ == "\_\_main\_\_":
30. match = "sadbutsad"
31. pattern = "sad"
32. *# arr1 = "abaabaababab"*
33. *# arr2 = "abmnab56abmnabkfghabmnab56abmnab"*
34. pi = compute\_prefix(pattern)
35. found = search(pattern=pattern, match=match)
36. if len(found) == 0:
37. print(-1)
38. else:
39. print(found)