

set8.py

```

1  def brute_force_match(text,pattern):
2      count = 0
3      for i in range(len(text)-len(pattern)+1):
4          print()
5          print("Alignment ",i)
6          j = 0
7          while j < len(pattern):
8              if(text[i+j] == pattern[j]):
9                  print(f"Comparing text[{i+j}] vs pattern[{j}]          match")
10                 j = j+1
11                 count+=1
12             else:
13                 print(f"Comparing text[{i+j}] vs pattern[{j}]          mismatch")
14                 print("Shift by 1")
15                 count+=1
16                 break
17             if(j==len(pattern)):
18                 print("Pattern found at index ", i)
19                 break
20         return count
21
22 res = brute_force_match("aaabaadaabaaa","aabaaa")
23 print("Total comparisions = ",res)
24
25
26 def brute_force_match_r2l(text,pattern):
27     count = 0
28     for i in range(len(pattern)-1,len(text)):
29         print()
30         print("Alignment ",i-len(pattern)+1)
31         j = len(pattern)-1
32         while j > -1:
33             if(text[i-len(pattern)+1+j] == pattern[j]):
34                 print(f"Comparing text[{i-len(pattern)+1+j}] vs pattern[{j}]          match")
35                 j = j-1
36                 count+=1
37             else:
38                 print(f"Comparing text[{i-len(pattern)+1+j}] vs pattern[{j}]          mismatch")
39                 print("Shift by 1")
40                 count+=1
41                 break
42             if(j== -1):
43                 print("Pattern found at index ", i-len(pattern)+1)
44                 break
45         return count
46
47

```

```
48 res = brute_force_match_r2l("aaabaadaabaaa", "aabaaa")
49 print("Total comparisions = ", res)
50
51
52
53 def last_occurrence_table(pattern):
54     table = {}
55     for i, ch in enumerate(pattern):
56         table[ch] = i
57     return table
58
59 def good_suffix_table(pattern):
60     m = len(pattern)
61     shift = [0] * (m + 1)
62     bpos = [0] * (m + 1)
63
64     i = m
65     j = m + 1
66     bpos[i] = j
67
68     # Preprocessing strong good suffix
69     while i > 0:
70         while j <= m and pattern[i - 1] != pattern[j - 1]:
71             if shift[j] == 0:
72                 shift[j] = j - i
73             j = bpos[j]
74         i -= 1
75         j -= 1
76         bpos[i] = j
77
78     # Preprocessing case 2
79     j = bpos[0]
80     for i in range(m + 1):
81         if shift[i] == 0:
82             shift[i] = j
83         if i == j:
84             j = bpos[j]
85     return shift
86
87 def boyer_moore(text, pattern):
88     n = len(text)
89     m = len(pattern)
90     last = last_occurrence_table(pattern)
91     gs = good_suffix_table(pattern)
92
93     print("Last occurrence table:", last)
94     print("Good suffix table:", gs)
95
96     comparisons = 0
97     s = 0 # shift of pattern w.r.t text
```

```
98
99     while s <= n - m:
100         j = m - 1
101
102         print(f"\nAlignment at text index {s}:")
103         while j >= 0 and pattern[j] == text[s + j]:
104             print(f"  Match: text[{s + j}] = '{text[s + j]}' and pattern[{j}] =
105 '{pattern[j]}'")
106             j -= 1
107             comparisons += 1
108
109         if j < 0:
110             print(f"Pattern found at index {s}")
111             s += gs[0]
112         else:
113             print(f"  Mismatch: text[{s + j}] = '{text[s + j]}' vs pattern[{j}] =
114 '{pattern[j]}'")
115             bad_char_shift = j - last.get(text[s + j], -1)
116             good_suffix_shift = gs[j + 1]
117             shift = max(bad_char_shift, good_suffix_shift, 1)
118             print(f"    Bad char shift = {bad_char_shift}, Good suffix shift =
119 {good_suffix_shift}")
120             print(f"    Pattern shifted by {shift}")
121             s += shift
122             comparisons += 1
123
124         print("\nTotal comparisons =", comparisons)
125         return comparisons
126
127 # Example run
128 boyer_moore("aaabaadaabaaa", "aabaaa")
129
```