**set8.py**

```
def brute_force_match(text,pattern):
    count = 0
    for i in range(len(text)-len(pattern)+1):
        print()
        print("Alignment ",i)
        j = 0
        while j < len(pattern):
            if(text[i+j] == pattern[j]):
                print(f"Comparing text[{i+j}] vs pattern[{j}]          match")
                j = j+1
                count+=1
            else:
                print(f"Comparing text[{i+j}] vs pattern[{j}]          mismatch")
                print("Shift by 1")
                count+=1
                break
        if(j==len(pattern)):
            print("Pattern found at index ", i)
            break
    return count

res = brute_force_match("aaabaadaabaaa","aabaaa")
print("Total comparisions = ",res)


def brute_force_match_r2l(text,pattern):
    count = 0
    for i in range(len(pattern)-1,len(text)):
        print()
        print("Alignment ",i-len(pattern)+1)
        j = len(pattern)-1
        while j > -1:
            if(text[i-len(pattern)+1+j] == pattern[j]):
                print(f"Comparing text[{i-len(pattern)+1+j}] vs pattern[{j}]          match")
                j = j-1
                count+=1
            else:
                print(f"Comparing text[{i-len(pattern)+1+j}] vs pattern[{j}]
mismatch")
                print("Shift by 1")
                count+=1
                break
        if(j==-1):
            print("Pattern found at index ", i-len(pattern)+1)
            break
    return count


```

```
res = brute_force_match_r2l("aaabaadaabaaa","aabaaa")
print("Total comparisions = ",res)



def last_occurrence_table(pattern):
    table = {}
    for i, ch in enumerate(pattern):
        table[ch] = i
    return table

def good_suffix_table(pattern):
    m = len(pattern)
    shift = [0] * (m + 1)
    bpos = [0] * (m + 1)

    i = m
    j = m + 1
    bpos[i] = j

    # Preprocessing strong good suffix
    while i > 0:
        while j <= m and pattern[i - 1] != pattern[j - 1]:
            if shift[j] == 0:
                shift[j] = j - i
            j = bpos[j]
        i -= 1
        j -= 1
        bpos[i] = j

    # Preprocessing case 2
    j = bpos[0]
    for i in range(m + 1):
        if shift[i] == 0:
            shift[i] = j
        if i == j:
            j = bpos[j]
    return shift

def boyer_moore(text, pattern):
    n = len(text)
    m = len(pattern)
    last = last_occurrence_table(pattern)
    gs = good_suffix_table(pattern)

    print("Last occurrence table:", last)
    print("Good suffix table:", gs)

    comparisons = 0
    s = 0  # shift of pattern w.r.t text
```

```
    while s <= n - m:
        j = m - 1

        print(f"\nAlignment at text index {s}:")
        while j >= 0 and pattern[j] == text[s + j]:
            print(f"  Match: text[{s + j}] = '{text[s + j]}' and pattern[{j}] = '{pattern[j]}'")
            j -= 1
            comparisons += 1

        if j < 0:
            print(f"Pattern found at index {s}")
            s += gs[0]
        else:
            print(f"  Mismatch: text[{s + j}] = '{text[s + j]}' vs pattern[{j}] = '{pattern[j]}'")
            bad_char_shift = j - last.get(text[s + j], -1)
            good_suffix_shift = gs[j + 1]
            shift = max(bad_char_shift, good_suffix_shift, 1)
            print(f"   Bad char shift = {bad_char_shift}, Good suffix shift = {good_suffix_shift}")
            print(f"   Pattern shifted by {shift}")
            s += shift
            comparisons += 1

    print("\nTotal comparisons =", comparisons)
    return comparisons


# Example run
boyer_moore("aaabaadaabaaa", "aabaaa")
```