badchar.py

```python
def preprocess_bad_character(pattern):
    """Preprocesses the bad character heuristic table using a dictionary."""
    bad_char = {}
    for i, ch in enumerate(pattern):
        bad_char[ch] = i
    return bad_char

def preprocess_good_suffix(pattern):
    """Preprocess strong good suffix rule (standard approach)."""
    m = len(pattern)
    shift = [0] * (m + 1)
    bpos = [0] * (m + 1)

    # Step 1: Preprocess strong suffix
    i = m
    j = m + 1
    bpos[i] = j
    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

    # Step 2: Preprocess case 2 (when there is no occurrence of suffix in pattern)
    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):
    """Full Boyer-Moore string search algorithm with correct strong good suffix rule."""
    n = len(text)
    m = len(pattern)
    if m == 0:
        return []

    bad_char = preprocess_bad_character(pattern)
    good_suffix = preprocess_good_suffix(pattern)

    result = []
    s = 0  # shift of the pattern
    while s <= n - m:
```

```
        j = m - 1
        while j >= 0 and pattern[j] == text[s + j]:
            j -= 1

        if j < 0:
            result.append(s)
            s += good_suffix[0]
        else:
            bc_shift = j - bad_char.get(text[s + j], -1)
            gs_shift = good_suffix[j + 1]
            s += max(bc_shift, gs_shift)
    return result

# Example usage
text = "ABAAABCDABCDA"
pattern = "ABCDA"
matches = boyer_moore(text, pattern)
print("Pattern found at indices:", matches)
```