KMP

January 4, 2024

1 Knuth–Morris–Pratt

1.0.1 CSE21035_S.MEENAKSHI

1.0.2 importing libraries

[9]:

```
import time
import matplotlib.pyplot as plt
import numpy as np
from tabulate import tabulate
```

1.0.3 defining Function -Knuth–Morris–Pratt

[10]:

```
def kmp_matcher(t, p):
    n = len(t)
    m = len(p)
    pi = pi_func(p)
    q = 0
    comparisons = 0
    match_found = False
    for i in range(n):
        comparisons += 1
        while q > 0 and p[q] != t[i]:
            comparisons += 1
            q = pi[q - 1]
        if p[q] == t[i]:
            q = q + 1
        if q == m:
            q = pi[q - 1]
            match_found = True
            break

    return comparisons

def pi_func(p):
    m = len(p)
    pi = [0] * m
    k = 0
    for q in range(1, m):
        while k > 0 and p[k] != p[q]:
            k = pi[k - 1]
        if p[k] == p[q]:
            k = k + 1
        pi[q] = k
    return pi
```

[11]:

```
def generate_pattern(length, regular=True):
    if regular:
        return "1" * length
    else:
        pattern = ""
        for i in range(length):
            pattern += "1" if i % 2 == 0 else "0"
        return pattern
```

[12]:

```
def analyze_pattern2(pattern, text):
    results = []

    test_case_sizes = [100, 200, 500, 1000, 5000, 10000]
    for size in test_case_sizes:
        sub_text = text[:size]
        start_time = time.time()
        comp_count = kmp_matcher(sub_text, pattern)
        end_time = time.time()

        results.append({
            "Test Case Size": size,
            "Comparisons": comp_count,
            "Running Time": "{:.10f}".format(end_time - start_time)
        })

    return results

def print_table(results, title):
    headers = results[0].keys()
    data = [list(result.values()) for result in results]
    print(f"\n{title}\n")
    print(tabulate(data, headers=headers, tablefmt="grid"))
```

[13]:

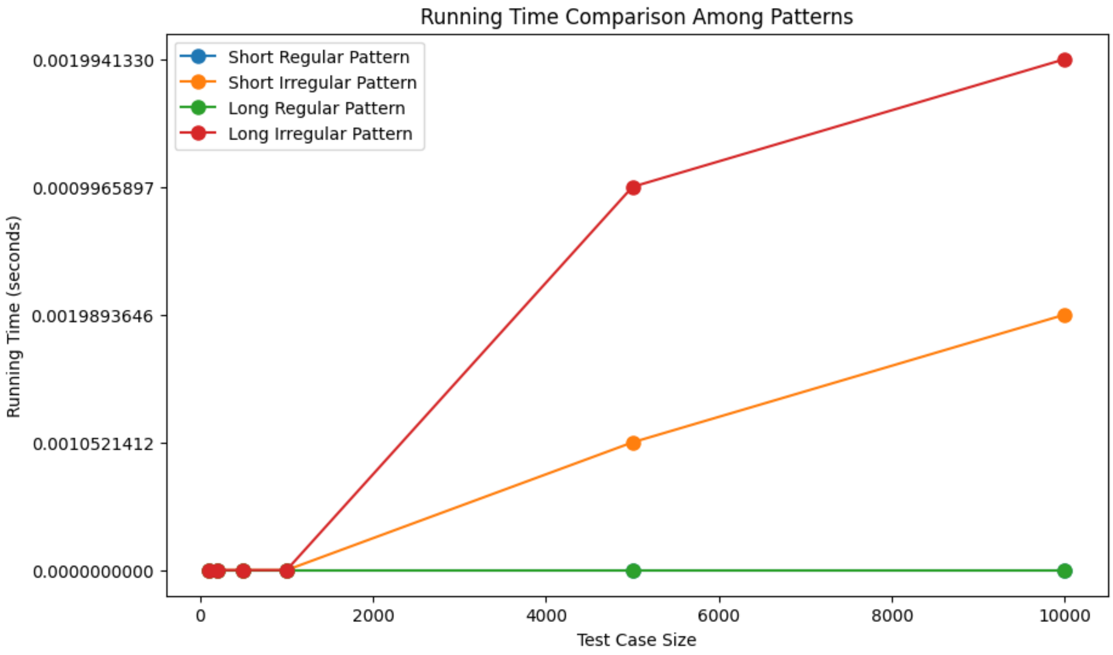
```
# Short and Regular Pattern
short_regular_pattern = generate_pattern(4, regular=True)
text = "1" * 10000  # Use a larger text for better analysis
short_regular_results = analyze_pattern2(short_regular_pattern, text)
print_table(short_regular_results, "Short Regular Pattern Analysis")
```

```python
# Short and Irregular Pattern
short_irregular_pattern = generate_pattern(4, regular=False)
short_irregular_results = analyze_pattern2(short_irregular_pattern, text)
print_table(short_irregular_results, "Short Irregular Pattern Analysis")

# Long and Regular Pattern
long_regular_pattern = generate_pattern(20, regular=True)
long_regular_results = analyze_pattern2(long_regular_pattern, text)
print_table(long_regular_results, "Long Regular Pattern Analysis")

# Long and Irregular Pattern
long_irregular_pattern = generate_pattern(20, regular=False)
long_irregular_results = analyze_pattern2(long_irregular_pattern, text)
print_table(long_irregular_results, "Long Irregular Pattern Analysis")
```

```
Short Regular Pattern Analysis

+------------------+---------------+----------------+
|   Test Case Size |   Comparisons |   Running Time |
+==================+===============+================+
|              100 |             4 |              0 |
+------------------+---------------+----------------+
|              200 |             4 |              0 |
+------------------+---------------+----------------+
|              500 |             4 |              0 |
+------------------+---------------+----------------+
|             1000 |             4 |              0 |
+------------------+---------------+----------------+
|             5000 |             4 |              0 |
+------------------+---------------+----------------+
|            10000 |             4 |              0 |
+------------------+---------------+----------------+

Short Irregular Pattern Analysis

+------------------+---------------+----------------+
|   Test Case Size |   Comparisons |   Running Time |
+==================+===============+================+
|              100 |           199 |    0           |
+------------------+---------------+----------------+
|              200 |           399 |    0           |
+------------------+---------------+----------------+
|              500 |           999 |    0           |
+------------------+---------------+----------------+
|             1000 |          1999 |    0.00101614  |
+------------------+---------------+----------------+
|             5000 |          9999 |    0.000978231 |
+------------------+---------------+----------------+
|            10000 |         19999 |    0.00156236  |
+------------------+---------------+----------------+

Long Regular Pattern Analysis

+------------------+---------------+----------------+
|   Test Case Size |   Comparisons |   Running Time |
+==================+===============+================+
|              100 |            20 |              0 |
+------------------+---------------+----------------+
|              200 |            20 |              0 |
+------------------+---------------+----------------+
|              500 |            20 |              0 |
+------------------+---------------+----------------+
|             1000 |            20 |              0 |
+------------------+---------------+----------------+
|             5000 |            20 |              0 |
+------------------+---------------+----------------+
|            10000 |            20 |              0 |
+------------------+---------------+----------------+

Long Irregular Pattern Analysis

+------------------+---------------+----------------+
|   Test Case Size |   Comparisons |   Running Time |
+==================+===============+================+
|              100 |           199 |     0          |
+------------------+---------------+----------------+
|              200 |           399 |     0          |
+------------------+---------------+----------------+
|              500 |           999 |     0          |
+------------------+---------------+----------------+
|             1000 |          1999 |     0          |
+------------------+---------------+----------------+
|             5000 |          9999 |     0.00253701 |
+------------------+---------------+----------------+
|            10000 |         19999 |     0.00199938 |
+------------------+---------------+----------------+
```

[14]:

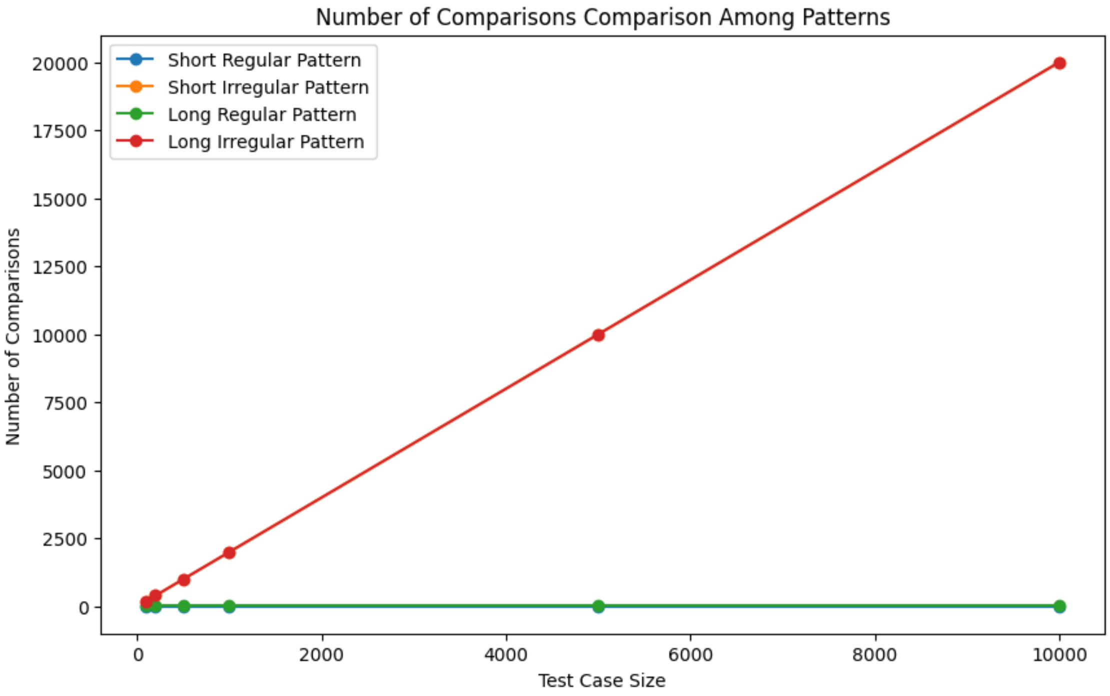
```python
def plot_running_time_comparison(patterns_results):
    plt.figure(figsize=(10, 6))

    for pattern_name, results in patterns_results.items():
        test_case_sizes = [result["Test Case Size"] for result in results]
        running_times = [result["Running Time"] for result in results]

        plt.plot(test_case_sizes, running_times, label=pattern_name,␣
 ↪marker='o',markersize=8)

    plt.xlabel("Test Case Size")
    plt.ylabel("Running Time (seconds)")
    plt.title("Running Time Comparison Among Patterns")
    plt.legend()
    plt.show()

# Analyzing multiple patterns
patterns_results = {}

# Short and Regular Pattern
short_regular_pattern = generate_pattern(4, regular=True)
short_regular_results = analyze_pattern2(short_regular_pattern, text)
patterns_results["Short Regular Pattern"] = short_regular_results

# Short and Irregular Pattern
short_irregular_pattern = generate_pattern(4, regular=False)
short_irregular_results = analyze_pattern2(short_irregular_pattern, text)
patterns_results["Short Irregular Pattern"] = short_irregular_results

# Long and Regular Pattern
long_regular_pattern = generate_pattern(20, regular=True)
long_regular_results = analyze_pattern2(long_regular_pattern, text)
patterns_results["Long Regular Pattern"] = long_regular_results

# Long and Irregular Pattern
long_irregular_pattern = generate_pattern(20, regular=False)
long_irregular_results = analyze_pattern2(long_irregular_pattern, text)
patterns_results["Long Irregular Pattern"] = long_irregular_results

# Plotting running time comparison
plot_running_time_comparison(patterns_results)
```



[15]:

```
def plot_comparisons_comparison(patterns_results):
    plt.figure(figsize=(10, 6))

    for pattern_name, results in patterns_results.items():
        test_case_sizes = [result["Test Case Size"] for result in results]
        comparisons = [result["Comparisons"] for result in results]

        plt.plot(test_case_sizes, comparisons, label=pattern_name, marker='o')

    plt.xlabel("Test Case Size")
    plt.ylabel("Number of Comparisons")
    plt.title("Number of Comparisons Comparison Among Patterns")
    plt.legend()
    plt.show()

# Analyzing multiple patterns
patterns_results = {}

# Short and Regular Pattern
short_regular_pattern = generate_pattern(4, regular=True)
short_regular_results = analyze_pattern2(short_regular_pattern, text)
patterns_results["Short Regular Pattern"] = short_regular_results

# Short and Irregular Pattern
short_irregular_pattern = generate_pattern(4, regular=False)
```

```
short_irregular_results = analyze_pattern2(short_irregular_pattern, text)
patterns_results["Short Irregular Pattern"] = short_irregular_results

# Long and Regular Pattern
long_regular_pattern = generate_pattern(20, regular=True)
long_regular_results = analyze_pattern2(long_regular_pattern, text)
patterns_results["Long Regular Pattern"] = long_regular_results

# Long and Irregular Pattern
long_irregular_pattern = generate_pattern(20, regular=False)
long_irregular_results = analyze_pattern2(long_irregular_pattern, text)
patterns_results["Long Irregular Pattern"] = long_irregular_results

# Plotting comparisons comparison
plot_comparisons_comparison(patterns_results)
```



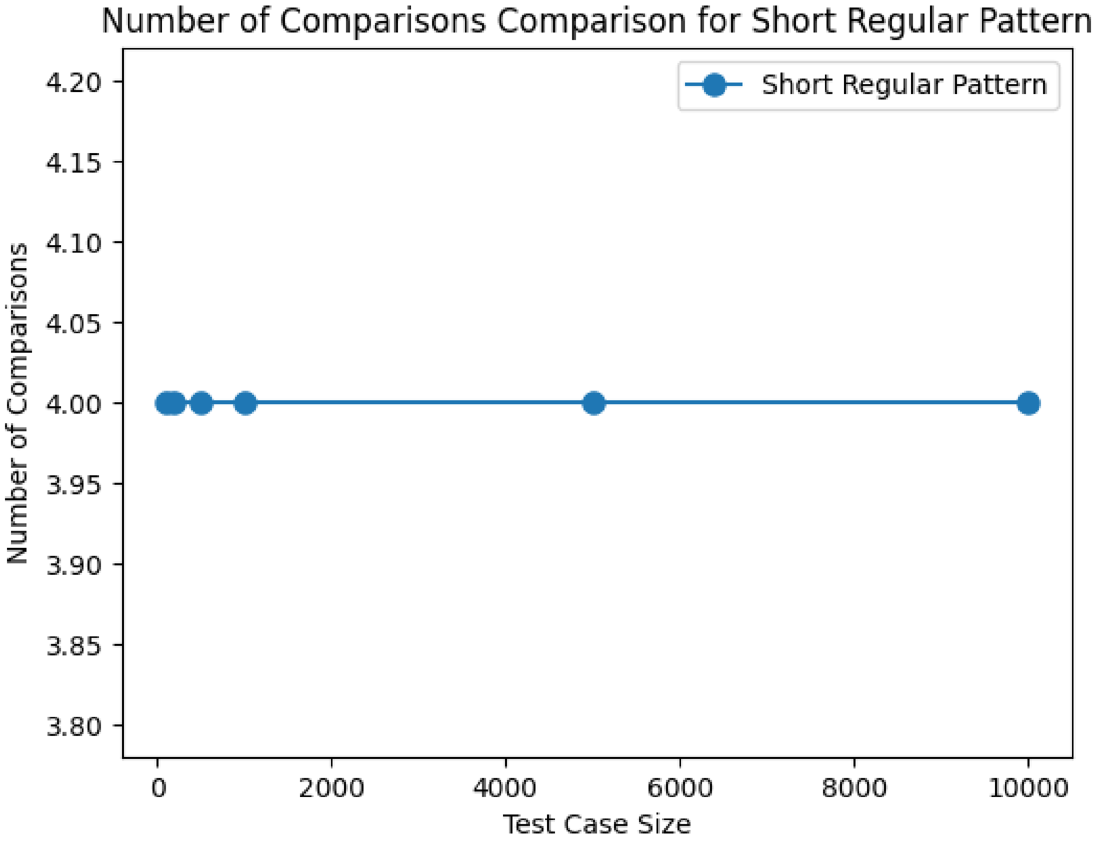
[ ]:

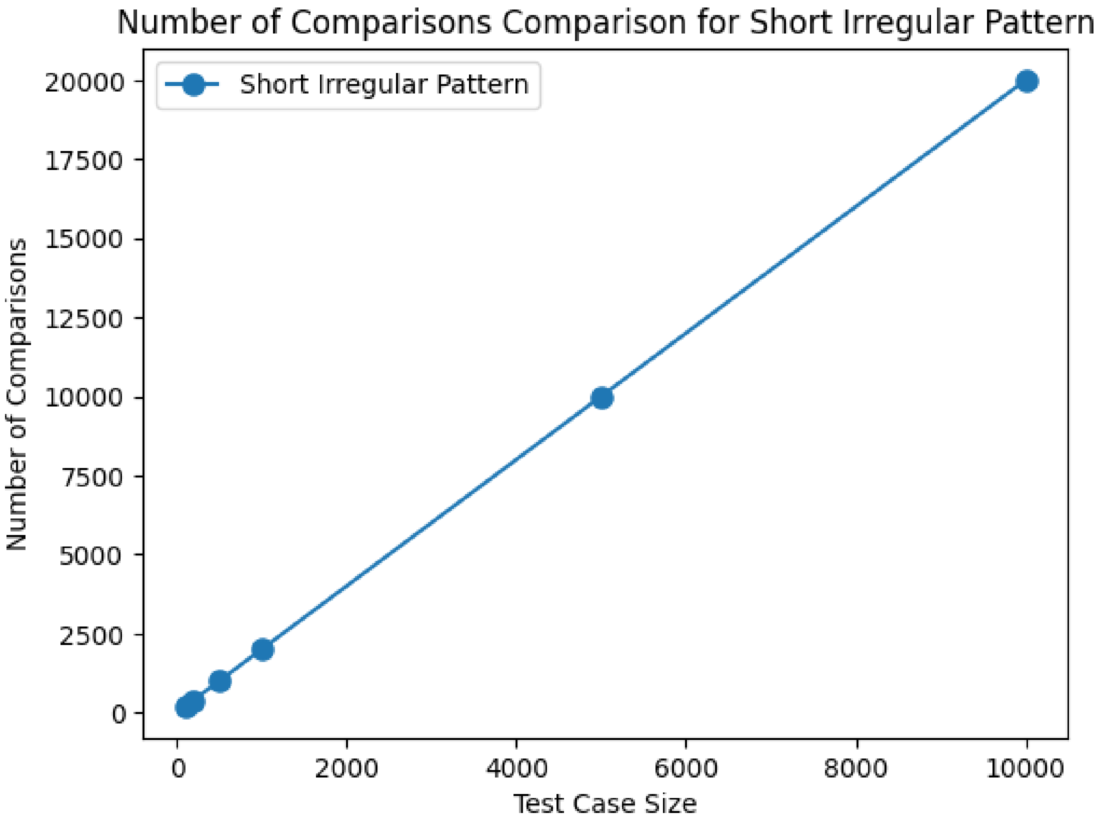
```
```

[16]:

```
def plot_comparisons_comparison(pattern_results, pattern_name):
    test_case_sizes = [result["Test Case Size"] for result in pattern_results]
    comparisons = [result["Comparisons"] for result in pattern_results]

    plt.plot(test_case_sizes, comparisons, label=pattern_name, marker='o',␣
 ↪markersize=8)
```

```
    plt.xlabel("Test Case Size")
    plt.ylabel("Number of Comparisons")
    plt.title(f"Number of Comparisons Comparison for {pattern_name}")
    plt.legend()
    plt.show()

# Analyzing short patterns
short_patterns_results = {}

# Short and Regular Pattern
short_regular_pattern = generate_pattern(4, regular=True)
short_regular_results = analyze_pattern2(short_regular_pattern, text)
short_patterns_results["Short Regular Pattern"] = short_regular_results

# Short and Irregular Pattern
short_irregular_pattern = generate_pattern(4, regular=False)
short_irregular_results = analyze_pattern2(short_irregular_pattern, text)
short_patterns_results["Short Irregular Pattern"] = short_irregular_results

# Plotting comparisons comparison for short patterns
for pattern_name, results in short_patterns_results.items():
    plot_comparisons_comparison(results, pattern_name)
```


Number of Comparisons Comparison for Short Regular Pattern
Short Regular Pattern
Number of Comparisons
4.20
4.15
4.10
4.05
4.00
3.95
3.90
3.85
3.80
0
2000
4000
6000
8000
10000
Test Case Size



1.0.4 Short and Regular Pattern (Best Case):

- **Pattern:** “1111”
- **Text:** “1111111111”
- **Performance:** In the best case, the pattern is short and regular, and it occurs at the beginning of the text. KMP will perform well with a small number of comparisons and quick execution time.

1.0.5 Short and Irregular Pattern:

- **Patterabab “1010”
- **Teaaaaaaaaaabbbbbbabababbbbbbb1011111”
- **Performance:** In this case, the pattern is short but irregular. KMP will still perform efficiently, but the number of comparisons may increase compared to the best case.

1.0.6 Long and Regular Pattern:

- **Paaaaaaaaa”1111111111” -aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa1111111
- **Performance:** KMP will perform well in this case, as it handles long regular patterns efficiently. The running time may increase slightly with the length of the pattern, but the number of comparisons remains relatively low.

1.0.7 Long and Irregular Pattern (Worst Case):

- **Pattern:** “1101101001011101101010”
- **Text:** “111111110110110100101110110101011111111111111111”
- **Performance:** In the worst case, the pattern is both long and irregular. KMP will still provide linear time complexity, but the number of comparisons will be higher compared to regular patterns.

1.0.8 Edge Case:

- **Pattern:** “1”
- **Text:** “0”
- **Performance:** In this edge case, the pattern occurs only once at the beginning of the text. KMP will perform well, and the number of comparisons will be minimal.

1.0.9 Overall:

- The KMP algorithm is efficient for various pattern types. linear time complexity.

2 Time Complexity

2.1 m->size of pattern.

2.2 n->size of text

2.3 -Running time of pi_func is Theta(m).

2.4 -Running time of kmp_matcher is Theta(n).

2.5 -Time complexity of KMP-> O(m+n)

```
[ ]:
```