Counting Pattern Occurrences

in the Periodic String B_k = S·S…S (k copies)

An O(n + m) algorithm — independent of k

Pattern Matching • String Periodicity • KMP

Problem Statement

Given: a pattern P, a string S, and a large integer k.

Let ***Bk*** = S repeated k times.

Goal: count occurrences of P in Bk — without ever constructing Bk explicitly.

$|S| = n$

pattern source length

$|P| = m$

pattern length

$k \approx 10^{18}$

may be astronomically large

Constraint: |Bk| = k·n can be enormous — building it is infeasible.

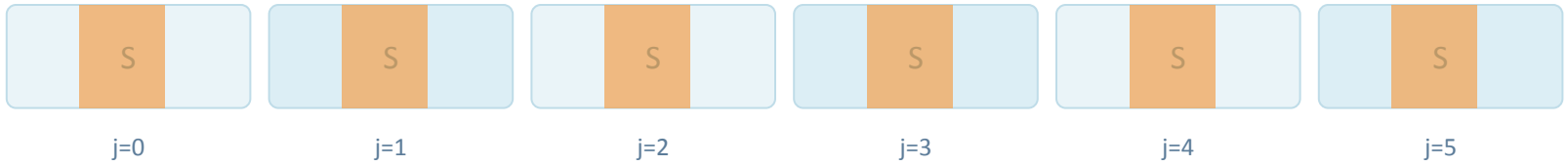
Key Idea: Periodicity

Bk is a prefix of the infinite periodic string ***S∞ = S·S·S·…***

$$S_\infty[p] = S_\infty[p + n] \quad \text{for every position } p \geq 0$$

Consequence:

Whether P matches starting at offset i ∈ [0, n) depends **only on i** — not on which copy j of S the match starts inside.



The highlighted offset i behaves identically in every copy of S.

Step 1 & 2: Build T, Run KMP

1 Build a small witness string

$$t = \lceil m/n \rceil + 1 \qquad T = S \text{ repeated } t \text{ times} \qquad |T| = nt$$

$$m + n \leq nt \leq m + 2n$$

$|T| = O(n + m)$ — independent of k. T is long enough that any match starting at offset $i \in [0, n)$ fits entirely inside it.

2 Match P against T with KMP

$$\text{Match} = \{\, i \in [0, n) : P = T[i \,..\, i+m-1] \,\}$$

Finds every valid starting offset — including clustered or overlapping matches — in $O(n + m)$ time.

Why this works

Match correctness at offset i is a property of i alone (by periodicity). Every position p in Bk decomposes uniquely as $p = j \cdot n + i$. So checking all $i \in [0,n)$ once covers every possible match phase, no matter how closely matches cluster within a period.

Step 3: Handling the Boundary of Bk

Every position p in Bk decomposes uniquely: **$p = j \cdot n + i$** (i = offset, j = copy index)

For each matched offset $i \in$ Match, the occurrence may need to reach into several following copies of S. We must not count an occurrence that runs past the end of Bk.

Copies needed to contain the match:

$$r_i = \lceil (i + m) / n \rceil$$

Valid starting copies j: condition $j \leq k - r_i$

Number of valid repetitions for offset i:

$$count_i = \max(0, k - r_i + 1)$$

Excludes occurrences that would spill past the end of Bk.

$$Answer = \sum_{i \in Match} \max(0, k - r_i + 1)$$

Algorithm Summary

```
function CountOccurrences(S, P, k):
    n = |S|;  m = |P|
    t = ceil(m / n) + 1
    T = S repeated t times                    # length O(n+m)
    Match = KMP_search(P, T)                   # offsets i in [0, n)

    answer = 0
    for i in Match:
        r_i = ceil((i + m) / n)
        if r_i <= k:
            answer += (k - r_i + 1)
    return answer
```

Total complexity: O(n + m)

Why No Match Is Missed or Double-Counted

1 Unique decomposition

Every position p in Bk has exactly one pair (i, j) with $p = j \cdot n + i$, $i \in [0, n)$. No overlap, no gap.

2 Offsets need not be n apart

Matches found in Match can be adjacent, overlapping, or clustered — KMP on T finds every valid i, regardless of spacing.

3 Long matches are captured

T has $t = \lceil m/n \rceil + 1$ copies of S, so even matches spanning several periods are verified correctly via T[i..i+m-1].

4 Boundary is checked separately

The count $\max(0, k - r_i + 1)$ only trims occurrences that would run past the end of Bk — it never affects whether a match is valid.

Complexity

$O(n + m)$	$O(n)$	$O(1)$
Build T + run KMP	Final summation over offsets	Dependence on k

Bottom line

We never build Bk. Periodicity reduces the problem to a single small window T of size O(n+m); KMP finds all match offsets once; a closed-form sum then accounts for repetitions and the boundary — giving an exact count even for astronomically large k.

Thank you