Отчёт по алгоритму KMP (Knuth–Morris–Pratt)

Идея алгоритма

Наивный поиск подстроки в тексте проверяет каждое возможное смещение подстроки, и при несовпадении возвращается в начало pattern. Это может приводить к повторной проверке символов текста, что делает сложность до $O(N \cdot M)$.

Алгоритм KMP решает эту проблему с помощью префикс-функции:

- Префикс-функция хранит информацию о том, сколько символов в начале pattern совпадает с концом текущего префикса pattern.
- Когда происходит несовпадение, KMP не возвращается в начало pattern, а использует эту информацию, чтобы продолжить сравнение с уже проверенной частью pattern.

Идея в том, что каждый символ текста обрабатывается не более одного раза, даже если были совпадения и несовпадения.

работа алгоритма

1. Вычисляется префикс-функция π для pattern: $\pi[i]$ = длина наибольшего собственного префикса pattern[0..i], который совпадает с суффиксом pattern[0..i] Эта функция показывает, сколько символов уже можно «спасти» при несовпадении.
2. Проходим по тексту символ за символом, поддерживая индекс j текущего совпадения в pattern.
3. Если $text[i] = pattern[j]$, увеличиваем j.
4. Если несовпадение:
   - Вместо того чтобы возвращаться к $j = 0$, используем $\pi[j-1]$
   - Это означает: «часть символов уже совпала, начинаем проверку с позиции $\pi[j-1]$».
5. Когда j достигает длины pattern, найдено вхождение подстроки в тексте.

Сложность алгоритма

- Построение префикс-функции: $O(M)$, где M — длина pattern. Каждый символ pattern обрабатывается один раз при вычислении массива π.
- Поиск подстроки в тексте: $O(N)$, где N — длина text. Каждый символ текста просматривается не более одного раза, а индекс в pattern может лишь увеличиваться или уменьшаться по π, но суммарно для всего текста это даёт линейное время.
- Итоговая сложность: $O(N + M)$, так как обе фазы (π и проход по тексту) выполняются последовательно.
- Память: $O(M)$ для хранения массива префикс-функции π.