-Cả ba thuật toán tìm kiếm chuỗi KMP (Knuth-Morris-Pratt), Rabin-Karp và Boyer-Moore đều được sử dụng để tìm kiếm một chuỗi con trong một chuỗi lớn hơn. Tuy nhiên giữa chúng đều có những ưu, khuyết điểm riêng biệt phụ thuộc vào tính chất dữ liệu.

-Quy ước ạ:

+Ưu điểm sẽ bôi màu xanh.

+Nhược điểm sẽ bôi màu đỏ.

1. **Knuth-Morris-Pratt (KMP)**:

- KMP sử dụng một bảng tiền xử lý bằng cách duyệt qua chuỗi mẫu một lần để tránh việc lặp lại các so sánh không cần thiết.

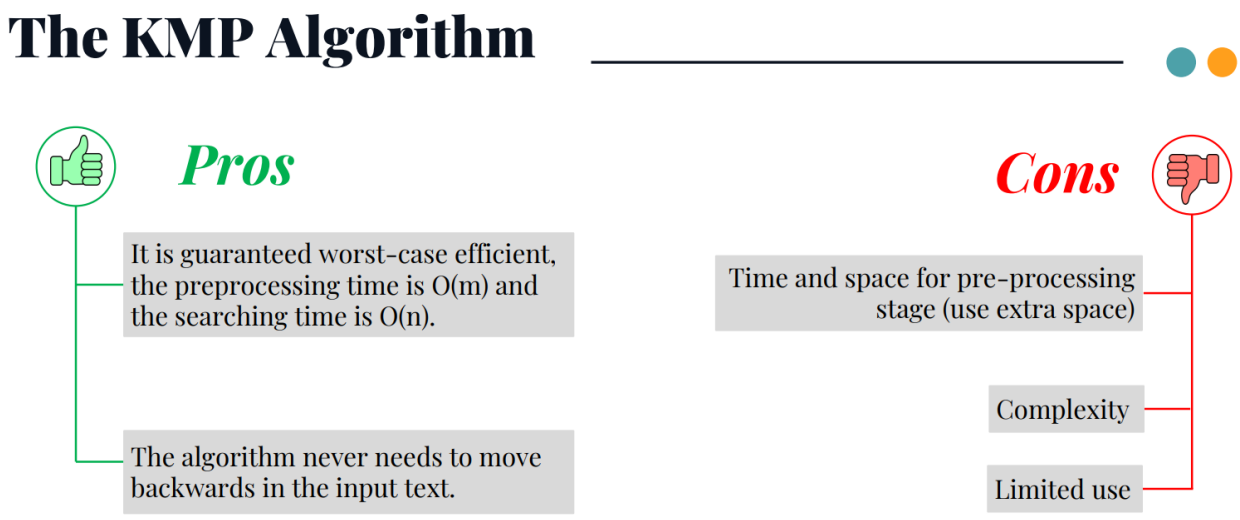
+Ví dụ bảng tiền xử lý cơ bản:

```python
def compute_lps(pattern):
    lps = [0] * len(pattern) # Longest Proper Prefix which is also suffix
    length = 0
    i = 1
    while i < len(pattern):
        if pattern[i] == pattern[length]:
            length += 1
            lps[i] = length
            i += 1
        else:
            if length != 0:
                length = lps[length - 1]
            else:
                lps[i] = 0
                i += 1
    return lps
```

- KMP không thực hiện so sánh ngược.

-Best & Average & Worst Case : khá ổn định với O(m+n), với m là độ dài của mẫu (pattern), n là độ dài của văn bản (text) . Với KMP thì độ phức tạp không phụ thuộc nhiều vào tính chất của dữ liệu.

-Không gian bộ nhớ: phải tốn thêm bộ nhớ cho bảng nên sẽ là O(m)



2. **Rabin-Karp**:

- Rabin-Karp dùng hàm băm để so sánh các phần tử của chuỗi mẫu với các phần tử của chuỗi văn bản, nó sử dụng giá trị băm để giảm bớt việc so sánh không cần thiết.Phù hợp với những Pattern dài Nếu giá trị băm của pattern và một chuỗi con của văn bản giống nhau, thuật toán sẽ kiểm tra từng ký tự.

-Thuật toán phụ thuộc vào việc chọn hàm băm, nên trong trường hợp hàm băm đã chọn gặp nhiều xung đột với dữ liệu cụ thể sẽ làm cho thuật toán chạy rất chậm.

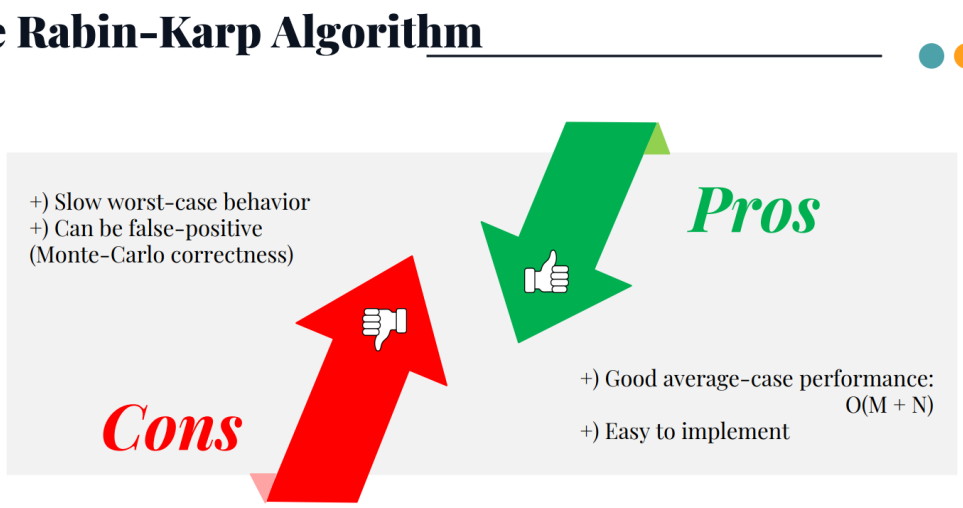
- Best Case: O(n+m), với n là độ dài của văn bản (text) và m là độ dài của pattern. Điều này xảy ra khi không có (hoặc rất ít) trùng lặp.

-Average Case: O(m+n), không chênh lệch nhiều với best case

- Worst Case: Khá chậm với O(nm), xảy ra khi tất cả (hoặc hầu hết) các ký tự của văn bản và pattern giống nhau.

Ví dụ: “aaaaaaaaaaaaaa” và “aa”

-Không gian bộ nhớ : O(1) , bởi chỉ cần tính lại giá trị băm khi dịch trên text nên tốn ít bộ nhớ nhất trong 3 thuật toán , bù lại phần nào cho thời gian chạy.



3. **Boyer-Moore**:

- Boyer-Moore sử dụng hai phương pháp chính (Bad Character Heuristic & Good Suffix Heuristic) để dịch chuyển pattern nhiều hơn so với việc chỉ dịch chuyển một ký tự như các thuật toán khác. Nó tập trung vào việc tối ưu hóa các bước nhảy dựa trên việc tìm kiếm và so sánh các ký tự không khớp. Đặc biệt phù hợp với những pattern dài, đặc biệt có thể sub-linear.

- Được cân nhắc là thuật toán phù hợp nhất trong các ứng dụng thông thường, như trong các trình soạn thảo văn bản và thay thế lệnh.

- Boyer-Moore thực hiện so sánh từ phải sang trái, điều này thường hiệu quả hơn vì nhiều ngữ cảnh, ký tự cuối cùng thường có khả năng phân biệt cao.

- Best Case: rất nhanh với O(n/m), xảy ra khi không có (hoặc rất ít) ký tự của pattern xuất hiện trong văn bản.

+Ví dụ: text “abcdefgh” và pattern “ijk”

+Ví dụ: pattern "jump" trong văn bản "The quick brown fox jumps over the lazy dog".

- Worst Case : O(mn), xảy ra khi tất cả (hoặc hầu hết) các ký tự của văn bản và pattern giống nhau, cũng như trùng lặp ở các kí tự cuối khiến cho ưu điểm của thuật toán này không phát huy được ưu thế. Thế nên thuật toán này sẽ bất lợi nếu pattern quá ngắn hoặc dùng để tìm với chuỗi nhị phân

+Ví dụ : text “aaaaaaaaaa” và pattern “aaaa”

-Không gian bộ nhớ : O(m+k) , với m là độ dài pattern, k là kích thước của bảng kí tự .

The Boyer-Moore Algorithm

Pros

+) Efficient: works particularly well for long search patterns, and it can be sub-linear

+) Considered the most efficient string-matching algorithm in usual application, such as, in text editors and command substitutions.

Cons

+) Not as good for binary strings or for very short patterns.

+) Worst-case time complexity: can be O(M*N) in certain cases.

Algorithm	Preprocessing Time	Matching Time	Best Case Time	Worst Case Time	Space Required
Brute-Force	O(1)	O(mn)	O(n)	O(nm)	O(1)
Knuth-Morris-Pratt (KMP)	O(m)	O(n)	O(m+n)	O(m+n)	O(m)
Boyer-Moore	O(m + k)	O(n)	O(n/m)	O(mn)	O(m + k)
Rabin-Karp	O(m)	O(n)	O(n+m)	O(nm)	O(1)

Kết luận:

Algorithm	Version	Operation count		Backup in input?	Correct?	Extra space
		Guarantee	Typical			
Brute Force	__	MN	1.1N	yes	yes	1
Knuth-Morris-Pratt	full DFA	2N	1.1N	no	yes	MR
	mismatch transitions only	3N	1.1N	no	yes	M
Boyer-Moore	full algorithm	3N	N/M	yes	yes	R
	mismatched char heuristic only	MN	N/M	yes	yes	R
Rabin-Karp	Monte Carlo	7N	7N	no	yes (*)	1
	Las Vegas	7N(*)	7N	yes	yes	1
*: probabilistic guarantee, with uniform and independent hash function						

+Với suy nghĩ khách quan của em , đối với tính chất dữ liệu thực tế và ứng dụng trong thực tế thì thứ tự đánh giá sẽ là:

1. **Boyer-Moore**:
   - Thuật toán này có hiệu suất tốt trong trường hợp điển hình N/M (N là độ dài chuỗi đích, M là độ dài chuỗi mẫu), đặc biệt hiệu quả khi chuỗi mẫu dài và có ít lần lặp lại trong chuỗi đích.
   - Thuật toán Boyer-Moore khai thác tính chất của dữ liệu để tối ưu quá trình tìm kiếm, giảm số lượng so sánh không cần thiết.
   - Nó hoạt động hiệu quả với nhiều loại dữ liệu khác nhau, đặc biệt là dữ liệu thực tế, ví dụ trong các trình soạn thảo văn bản và thay thế lệnh.
2. **Knuth-Morris-Pratt (KMP)**:
   - Thuật toán này có hiệu suất tốt trong trường hợp điển hình 1.1N, gần tuyến tính.

- KMP đặc biệt hiệu quả khi chuỗi mẫu có nhiều lần lặp lại trong chuỗi đích. Và hiệu quả là khá ổn định với đa số trường hợp. Có thể coi là ổn định nhất trong ba thuật toán.
- Tuy nhiên, KMP có độ phức tạp không gian cao hơn so với Boyer-Moore trong một số trường hợp.

3. **Rabin-Karp**:

- Thuật toán này có hiệu suất kém hơn so với Boyer-Moore và KMP trong trường hợp điển hình 7N.
- Rabin-Karp phụ thuộc vào hàm băm và có thể gặp vấn đề với các trường hợp xấu khi xảy ra nhiều va chạm trong hàm băm.
- Tuy nhiên, Rabin-Karp có ưu điểm là đơn giản để triển khai và hiệu quả với dữ liệu ngẫu nhiên. Độ phức tạp không gian cũng là ít nhất O(1).

Vì vậy , Lựa chọn thuật toán tối ưu cũng phụ thuộc vào bối cảnh và đặc điểm của dữ liệu cụ thể. Trong một số trường hợp, Rabin-Karp hoặc KMP có thể là lựa chọn tốt hơn Boyer-Moore tùy thuộc vào tính chất của dữ liệu đầu vào.(Như khi chúng ta cần không gian bộ nhớ và pattern dài thì có thể cân nhắc Rabin-Karp.)

Code về 3 thuật toán

+KMP

+Rabin-Karp

+Boyer-Moore Bad Character Heuristic (comment), Good Character Suffix

đã được em upload tại :

https://github.com/Limdim1604/IT003/tree/main/So%20s%C3%A1nh%20s%E1%BA%AFp%20x%E1%BA%BFp%20chu%E1%BB%97i

-Những Link chính em đã tham khảo gồm :

+File pp Stringmatching trên Course

https://cmps-people.ok.ubc.ca/ylucet/DS/Algorithms.html

https://www.geeksforgeeks.org/kmp-algorithm-for-pattern-searching/

https://www.youtube.com/watch?v=cH-5KcgUcOE

https://www.geeksforgeeks.org/rabin-karp-algorithm-for-pattern-searching/

https://www.youtube.com/watch?v=oxd_Z1osgCk

https://www.geeksforgeeks.org/boyer-moore-algorithm-for-pattern-searching/?ref=lbp

https://www.geeksforgeeks.org/boyer-moore-algorithm-good-suffix-heuristic/

https://www.youtube.com/watch?v=Wj606N0IAsw