Thuật toán Naive/Brute Force

Là thuật toán cơ bản và đơn giản nhất trong các thuật toán tìm kiếm chuỗi con pattern trong chuỗi cha text, sử dụng nguyên lý vét cạn. Có thể giải thích đơn giản, thuật toán Naive/Brute Force so sánh lần lượt mỗi chuỗi con subtext của text có cùng chiều dài với pattern với pattern, nếu tìm được, trả về kết quả là vị trí tìm thấy; khi không tìm được kết quả mong muốn, trả về giá trị quy ước là không tìm thấy.

Mã giả cho thuật toán Naive/Brute Force:

```
function: Brute-Force String Matching
text: chuỗi cha
pattern: chuỗi cần tìm
subtext: 1 đoạn thân của chuỗi cha có cùng chiều dài với pattern.
Trả về: vị trí tìm thấy đầu tiên hoặc -1 nếu không tìm thấy.

  vị trí tìm thấy = -1
  subtext = chuỗi con đầu text có độ dài bằng pattern
  while (chưa tìm thấy hoặc chưa tới cuối text)
    if (từng ký tự của subtext = pattern):
      trả về vị trí
    else:
      dịch chuyển chuỗi con subtext trong text sang phải 1 chữ cái
  Trả về: vị trí tìm thấy
```

Cài đặt thuật toán Naive/Brute Force:

- Khai báo hàm và các biến:

```python
def search(pat, txt):
    M = len(pat)
    N = len(txt)
```

  Với pat là pattern, txt là text.

- Thực hiện vòng lặp để xét pattern:

```python
    for i in range(N - M + 1):
        j = 0
        while (j < M):
            if (txt[i + j] != pat[j]):
                break
            j += 1

        if (j == M):
            print("Pattern found at index ", i)
```

Xét M kí tự liên tiếp, nếu subtext match với pattern thì trả về vị trí của pattern trong text, nếu không thì break.

Thuật toán Naive/Brute Force không cần công việc chuẩn bị cũng như các mảng phụ cho quá trình tìm kiếm. Với chuỗi pattern có độ dài M và chuỗi text có độ dài N thì trong trường hợp xấu nhất:

- Mỗi lần so sánh với subtext, pattern phải so sánh nhiều nhất là M lần (trong trường hợp cả M – 1 kí tự đầu đều đúng).
- Có tất cả N – M + 1 chuỗi, vậy số chuỗi cần so sánh nhiều nhất là N – M + 1 subtext như vậy (trong trường hợp N – M + 2 chuỗi subtext đầu không trùng với pattern).
  - ⇨ Cần M(N – M + 1) lần, vì duyệt tới cuối mảng nên đây là trường hợp tìm thấy ở cuối mảng hoặc không tìm thấy.
  - ⇨ Cận trên O(MN) (vì N > N – M + 1).
  - ⇨ Cấp phát bộ nhớ: 0.

Phân tích độ phức tạp thuật toán trong trường hợp tốt nhất:

- Trong trường hợp tốt nhất, có thể thấy pattern chính là subtext đầu tiên của text.
- Như vậy, chỉ cần tốn M lần so sánh các kí tự.
  - ⇨ Cần M lần.
  - ⇨ Cận trên O(M).
  - ⇨ Cấp phát bộ nhớ: 0.

Độ phức tạp thuật toán trong trường hợp trung bình: O(MN).

Đánh giá:

- Dễ hiểu, thuật toán này chỉ duyệt từ đầu đến cuối, so sánh tuần tự từng chuỗi con với chuỗi cần tìm kiếm.
- Không cần bước tiền xử lý.
- Độ phức tạp O(MN).
- Không cần xin thêm bộ nhớ.