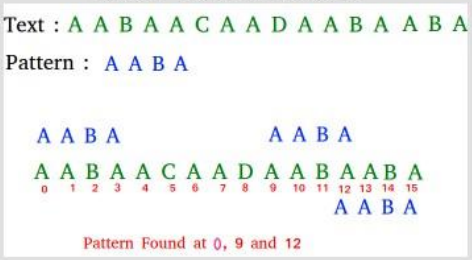
Naive Pattern Searching Algorithm

Given a text *txt[0..n-1]* and a pattern *pat[0..m-1]*, write a function *search(char pat[], char txt[])* that prints all occurrences of *pat[]* in *txt[]*. You may assume that *n > m*.

Examples:

```
Input:  txt[] = "THIS IS A TEST TEXT"
        pat[] = "TEST"
Output: Pattern found at index 10

Input:  txt[] =  "AABAACAADAABAABA"
        pat[] =  "AABA"
Output: Pattern found at index 0
        Pattern found at index 9
        Pattern found at index 12
```



Pattern searching is an important problem in computer science. When we do search for a string in notepad/word file or browser or database, pattern searching algorithms are used to show the search results.

Naive Pattern Searching: The idea is to slide the pattern over text one by one and check for a match. If a match is found, then slides by 1 again to check for subsequent matches.

That is check for the match of the first character of the pattern in the string, if it matches then check for the subsequent characters of the pattern with the respective characters of the string. If a mismatch is found then move forward in the string.

Below is the implementation of the above approach:

C++ Java

```cpp
// C++ program for Naive Pattern Searching algorithm
#include<bits/stdc++.h>
using namespace std;
void search(char* pat, char* txt)
{
    int M = strlen(pat);
    int N = strlen(txt);
    /* A loop to slide pat[] one by one */
    for (int i = 0; i <= N - M; i++) {
        int j;
        /* For current index i, check for pattern match */
        for (j = 0; j < M; j++)
            if (txt[i + j] != pat[j])
                break;
        if (j == M) // if pat[0...M-1] = txt[i, i+1, ...i+M-1]
            cout << "Pattern found at index "<< i << endl;
    }
}
int main()
{
    char txt[] = "AABAACAADAABAAABAA";
    char pat[] = "AABA";
    search(pat, txt);
    return 0;
}
```

Output:

```
Pattern found at index 0
Pattern found at index 9
Pattern found at index 13
```

What is the best case? The best case occurs when the first character of the pattern is not present in text at all.

```
txt[] = "AABCCAADDEE";
pat[] = "FAA";
```

The number of comparisons in best case is O(n).

What is the worst case ? The worst case of Naive Pattern Searching occurs in following scenarios.

1. When all characters of the text and pattern are same.

```
txt[] = "AAAAAAAAAAAAAAAAAA";
pat[] = "AAAAA";
```

2. Worst case also occurs when only the last character is different.

```
txt[] = "AAAAAAAAAAAAAAAAAB";
pat[] = "AAAAB";
```

The number of comparisons in the worst case is O(m*(n-m+1)). Although strings which have repeated characters are not likely to appear in English text, they may well occur in other applications (for example, in binary texts). The KMP matching algorithm improves the worst case to O(n). We will be covering KMP in the next post. Also, we will be writing more posts to cover all pattern searching algorithms and data structures.