



## Experiment 3.3

**Student Name:** Lipakshi

**UID:** 20BCS5082

**Branch:** CSE

**Section/Group:** 607-B

**Semester:** 5<sup>th</sup>

**Date of Performance:** 03/11/2022

**Subject Name:** DAA LAB

**Subject Code:** 21CSP-312

### **1. Aim/Overview of the practical:**

Knuth morris pratt to search pattern matching

### **2. Task to be done/which logistics used:**

Knuth morris pratt to search pattern matching

### **3.Steps for experiment/practical/Code:**

```
#include <bits/stdc++.h>

void computeLPSArray(char* pat, int M, int* lps);

// Prints occurrences of txt[] in pat[]
void KMPSearch(char* pat, char* txt)
{
    int M = strlen(pat);
    int N = strlen(txt);

    // create lps[] that will hold the longest prefix suffix
    // values for pattern
    int lps[M];

    // Preprocess the pattern (calculate lps[] array)
    computeLPSArray(pat, M, lps);

    int i = 0; // index for txt[]
    int j = 0; // index for pat[]
    while ((N - i) >= (M - j)) {
        if (pat[j] == txt[i]) {
            j++;
            i++;
        }

        if (j == M) {
            printf("Found pattern at index %d ", i - j);
```

```

        j = lps[j - 1];
    }

    // mismatch after j matches
    else if (i < N && pat[j] != txt[i]) {
        // Do not match lps[0..lps[j]-1] characters,
        // they will match anyway
        if (j != 0)
            j = lps[j - 1];
        else
            i = i + 1;
    }
}

}

// Fills lps[] for given pattern pat[0..M-1]
void computeLPSArray(char* pat, int M, int* lps)
{
    // length of the previous longest prefix suffix
    int len = 0;

    lps[0] = 0; // lps[0] is always 0

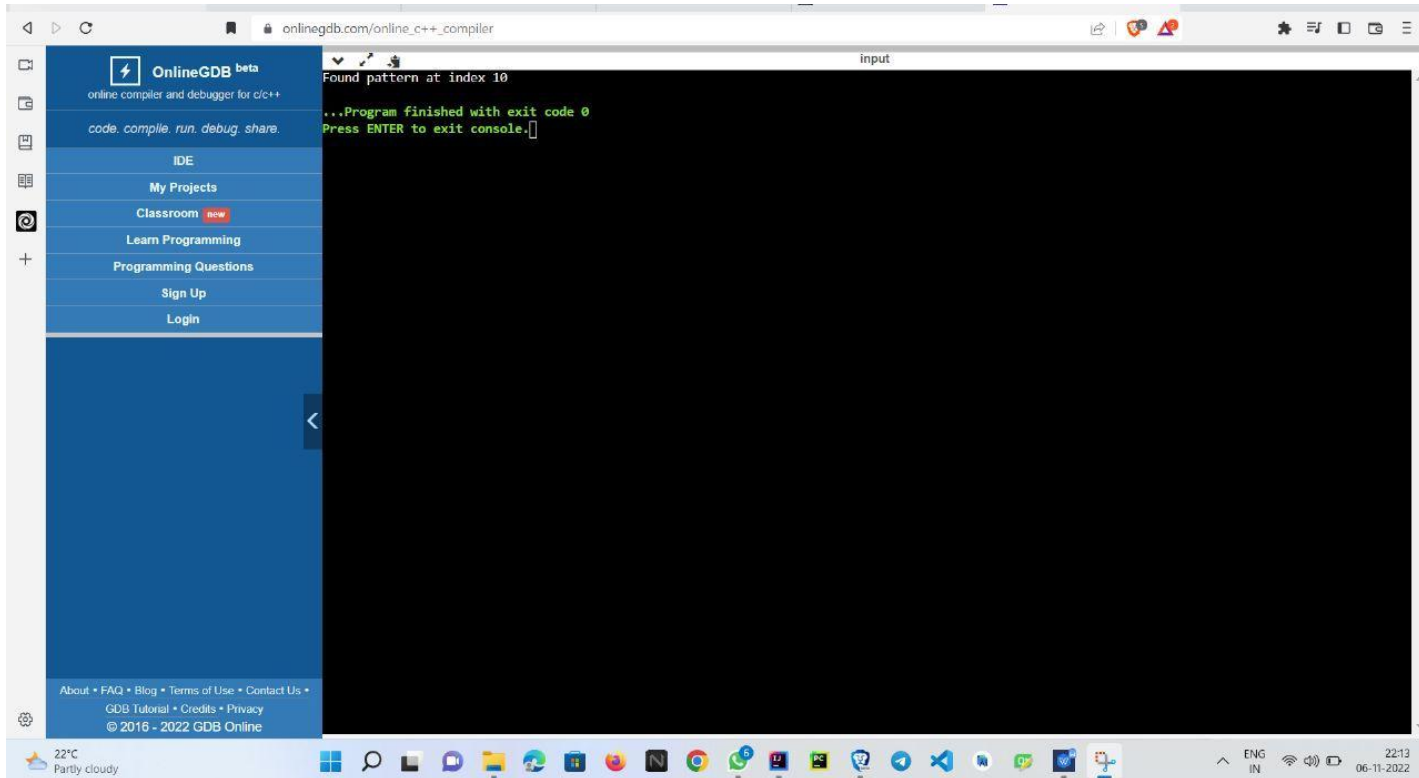
    // the loop calculates lps[i] for i = 1 to M-1
    int i = 1;
    while (i < M) {
        if (pat[i] == pat[len]) {
            len++;
            lps[i] = len;
            i++;
        }
        else // (pat[i] != pat[len])
        {
            // This is tricky. Consider the example.
            // AAACAAAA and i = 7. The idea is similar
            // to search step.
            if (len != 0) {
                len = lps[len - 1];

                // Also, note that we do not increment
                // i here
            }
            else // if (len == 0)
            {
                lps[i] = 0;
                i++;
            }
        }
    }
}

```

```
// Driver program to test above function
int main()
{
    char txt[] = "ABABDABACDABABCABAB";
    char pat[] = "ABABCABAB";
    KMPSearch(pat, txt);
    return 0;
}
```

## 4. Output:



The screenshot shows the OnlineGDB beta web IDE interface. The left sidebar contains navigation links: IDE, My Projects, Classroom (new), Learn Programming, Programming Questions, Sign Up, and Login. The main editor area displays the output of the program, which is: Found pattern at index 10, followed by a green message: ...Program finished with exit code 0, and a prompt: Press ENTER to exit console. The browser address bar shows onlinegdb.com/online\_c++\_compiler. The Windows taskbar at the bottom shows the date and time as 06-11-2022, 22:13.

```
Found pattern at index 10
...Program finished with exit code 0
Press ENTER to exit console.
```

## 5. Learning Outcomes:

- Create a program keeping in mind the time complexity.
- Create a program keeping in mind the space complexity.



# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

- c) Steps to make optimal algorithm.
- d) Learnt about how to implement 0-1 Knapsack problem using dynamic programming.



# **DEPARTMENT OF** **COMPUTER SCIENCE & ENGINEERING**

Discover. Learn. Empower.