**NAME: ABHINAV SRIVASTAVA**

**ENROLLMENT: 9918103122**

**BATCH: F5**

**APS LAB(May 17)**

**Question 1)**

**Given a text txt[0..n-1] and a pattern pat[0..m-1], write a program to search(char pat[], char txt[]) that prints all occurrences of pat[] in txt[] using Rabin-Karp algorithm for Pattern Searching. You may assume that n > m.**

**Solution:**

#include <bits/stdc++.h>

using namespace std;

#define d 256

void search(char pat[], char txt[], int q)

{

int M = strlen(pat);

int N = strlen(txt);

int i, j;

int p = 0;

int t = 0;

int h = 1;

for (i = 0; i < M - 1; i++)

{

h = (h \* d) % q;

}

for (i = 0; i < M; i++)

{

p = (d \* p + pat[i]) % q;

t = (d \* t + txt[i]) % q;

}

for (i = 0; i <= N - M; i++)

{

if ( p == t )

{

for (j = 0; j < M; j++)

{

if (txt[i+j] != pat[j])

{

break;

}

}

if (j == M)

cout<<"Pattern found at index "<< i<<endl;

}

if ( i < N-M )

{

t = (d\*(t - txt[i]\*h) + txt[i+M])%q;

if (t < 0)

{

t = (t + q);

}

}

}

}

int main()

{

char txt[] = "hello , say hello to the user ";

char pat[] = "hello";

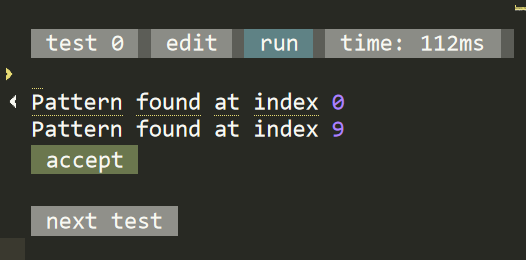
int q = 101;

search(pat, txt, q);

return 0;

}

Output:



Time Complexity:

O(n-m+1)

**Question 2)**

**Given an input string and a dictionary of words, find out if the input string can**

**be segmented into a space-separated sequence of dictionary words. See following**

**examples for more details.**

**Consider the following dictionary**

**{ i, like, sam, sung, samsung, mobile, ice,**

**cream, icecream, man, go, mango}**

**Input: ilikesamsung**

**Output: Yes**

**The string can be segmented as "i like samsung" or**

**"i like sam sung".**

**Solution:**

#include <bits/stdc++.h>

using namespace std;

int dictionaryContains(string word)

{

string dictionary[] = { "mobile", "samsung", "sam",

"sung", "man", "mango",

"icecream", "and", "go",

"i", "like", "ice", "cream" };

int size = sizeof(dictionary) / sizeof(dictionary[0]);

for (int i = 0; i < size; i++)

{

if (dictionary[i].compare(word) == 0)

{

return true;

}

}

return false;

}

bool wordBreak(string s)

{

int n = s.size();

if (n == 0)

{

return true;

}

vector<bool> dp(n + 1, 0);

vector<int> matched\_index;

matched\_index.push\_back(-1);

for (int i = 0; i < n; i++)

{

int msize = matched\_index.size();

int f = 0;

for (int j = msize - 1; j >= 0; j--)

{

string sb = s.substr(matched\_index[j] + 1, i - matched\_index[j]);

if (dictionaryContains(sb))

{

f = 1;

break;

}

}

if (f == 1)

{

dp[i] = 1;

matched\_index.push\_back(i);

}

}

return dp[n - 1];

}

int main()

{

if( wordBreak("ilikesamsung") )

{

cout << "Yes"<< endl;

}

else

{

cout << "No" << endl;

}

if( wordBreak("iiiiiiii") )

{

cout << "Yes"<< endl;

}

else

{

cout << "No" << endl;

}

if( wordBreak("") )

{

cout << "Yes"<< endl;

}

else

{

cout << "No" << endl;

}

if( wordBreak("ilikelikeimangoiii") )

{

cout << "Yes"<< endl;

}

else

{

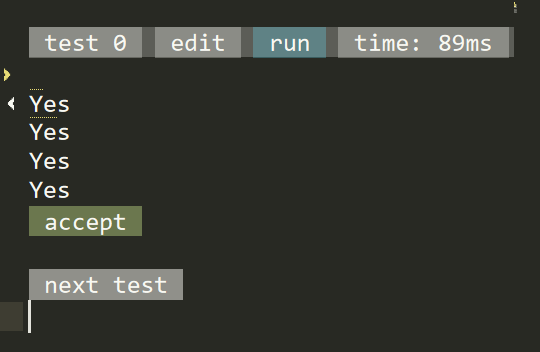
cout << "No" << endl;

}

return 0;

**}**

Output:



Time Complexity:

O(n+m) using KMP algorithm

**Question 3)**

**Given a binary matrix. Write a program which prints all unique rows of the**

**given matrix. The function takes three arguments: the first argument is a**

**matrix M and the next two arguments are row and col denoting the rows and**

**columns of the matrix.**

**Input**

**1**

**3 4**

**1 1 0 1 1 0 0 1 1 1 0 1**

**Output**

**1 1 0 1 $1 0 0 1 $**

**Solution:**

#include<bits/stdc++.h>

using namespace std;

void printArray(int arr[500][500], int row, int col)

{

unordered\_set<string> uset;

for(int i = 0; i < row; i++)

{

string s = "";

for(int j = 0; j < col; j++)

{

s += to\_string(arr[i][j]);

}

if(uset.count(s) == 0)

{

uset.insert(s);

cout << s << endl;

}

}

}

int main()

{

int m,n,x;

cin>>x;

int arr[500][500] ;

for(int i =0 ; i < x ; i++)

{

cin>>m>>n;

for(int r =0 ; r < m ; r++)

{

for(int s =0 ; s < n ; s++)

{

cin>>arr[r][s];

}

}

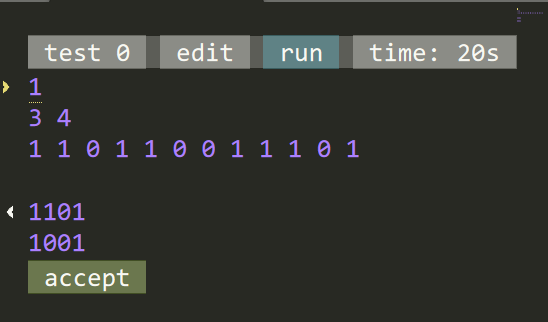
printArray(arr, m, n);

}

return 0;

}

Output:



Time Complexity:

O(n+m)