Rabin-Karp Substring Search

Implement the Rabin-Karp algorithm for substring search using a rolling hash. Discuss the impact of hash collisions on the algorithm's performance and how to handle them.

Code: -

```
package com.wipro.assignment;

import java.util.Arrays;

public class RabinKarp {

    public static int[]
search(String text, String
pattern, int d) {
        int n = text.length();
        int m =
pattern.length();
        int p = 0; // hash
value for the pattern
```

```
        int t = 0; // hash
value for the current window in
the text
        int h = 1;  // hash
function constant (d^(m-1))
        int[] result = new
int[0];

        if (m > n) {
            return result;
        }

        // Pre-compute d^(m-1)
        for (int i = 1; i < m;
i++) {
            h = (h * d) % d;
        }

        // Calculate hash
values for pattern and first
window of text
```

```
        for (int i = 0; i < m;
i++) {
            p = (d * p + (int)
pattern.charAt(i)) % d;
            t = (d * t + (int)
text.charAt(i)) % d;
        }

        int i = 0;
        while (i <= n - m) {
            // Check if hash
values match
            if (p == t) {
                // Potential
match, check characters one by
one
                boolean match =
true;
                for (int j = 0;
j < m; j++) {
```

```
                    if
(text.charAt(i + j) !=
pattern.charAt(j)) {
                        match =
false;
                        break;
                    }
                }
                if (match) {
                    // Add
starting index of the match to
the result array
                    result =
Arrays.copyOf(result,
result.length + 1);

result[result.length - 1] = i;
                }
            }

            // Shift the window
(rolling hash)
```

```
                if (i < n - m) {
                    t = (d * (t -
(int) text.charAt(i) * h) +
(int) text.charAt(i + m)) % d;
                }

                i++;
            }

            return result;
        }

        public static void
main(String[] args) {
            String text =
"GEEKSFORGEEKS";
            String pattern = "FOR";
            int d = 256;
            int[] matches =
search(text, pattern, d);
```

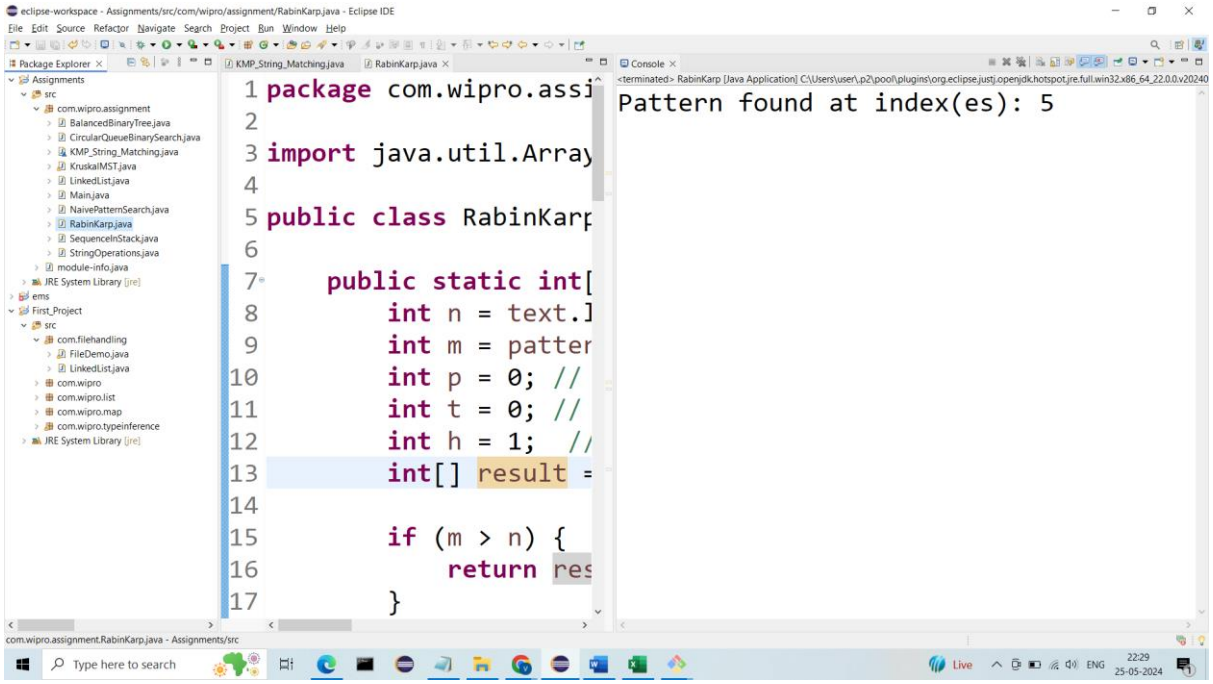
```
        if (matches.length > 0)
{

System.out.print("Pattern found
at index(es): ");
            for (int i = 0; i <
matches.length; i++) {

System.out.print(matches[i] + "
");
            }
        } else {

System.out.println("Pattern not
found");
        }
    }
}
```

Output: -


eclipse-workspace - Assignments/src/com/wipro/assignment/RabinKarp.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
Assignments
src
com.wipro.assignment
BalancedBinaryTree.java
CircularQueueBinarySearch.java
KMP_String_Matching.java
KruskalMST.java
LinkedList.java
Main.java
NaivePatternSearch.java
RabinKarp.java
SequenceInStack.java
StringOperations.java
module-info.java
JRE System Library [jre]
ems
First_Project
src
com.filehandling
FileDemo.java
LinkedList.java
com.wipro
com.wipro.list
com.wipro.map
com.wipro.typeinference
JRE System Library [jre]
KMP_String_Matching.java
RabinKarp.java
1 package com.wipro.assi
2
3 import java.util.Array
4
5 public class RabinKarp
6
7 public static int[
8 int n = text.l
9 int m = patter
10 int p = 0; //
11 int t = 0; //
12 int h = 1; //
13 int[] result =
14
15 if (m > n) {
16 return res
17 }
Console
<terminated> RabinKarp [Java Application] C:\Users\user\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_22.0.0.v20240
Pattern found at index(es): 5
com.wipro.assignment.RabinKarp.java - Assignments/src
Type here to search
Live
ENG
22:29
25-05-2024