Karp-Rabin String Algorithm

Q) Fing whether string a is a subset of string b
string a = "Kunal" → size (n) of the array
string b = "Prakash Kunal Apoorv" → size (m) of the array

If we go through comparing every 5 letters with Kunal. It would be quadratic time complexity.
$O(n * m)$

Can we do it in a better time complexity like linear time complexity $O(N+M)$.

Yes, by using Karp Rabin Algorithm

$h(a) = h\, b[i : i + len(a)]$

a = $\overset{0}{K}\ \overset{1}{U}\ \overset{2}{N}\ \overset{3}{A}\ \overset{4}{L}$
b = $\overset{0}{A}\ \overset{1}{P}\ \overset{2}{O}\ \overset{3}{O}\ \overset{4}{R}\ \overset{5}{V}\ \ \overset{6}{K}\ \overset{7}{U}\ \overset{8}{N}\ \overset{9}{A}\ \overset{10}{L}\ \ \overset{11}{R}\ \overset{12}{A}\ \overset{13}{H}\ \overset{14}{U}\ \overset{15}{L}$

It will compare the entire hashvalue of 'a' with hash value of b like slots.
So first slot of size 5 [APUOR]
then next slot of size 5 [POORV]

① If hash (a) = $h\, b[i : i + len(a)]$

★ It might be possible that match has found Same string will have the same hash value. Also we know that it's possible that hash value of Kunal is equal to hash value of Rahul. At the end it is just an algorithm. Probability of this happening is low.

→ ★ check every character of O(a)
if all characters equal found ans

★★ suppose not all chars are equal.

{ hash(a) = h (substring)
but a != substring

→ Probability of this happening

$P = \frac{1}{\text{len}(a)}$ ⟶ Universal hashing lecture

→ Expected or amortized cost per i (index) is constant.
i = O(1)

Time = O[length(a) + length(b) * cost of hash functions]

→ worst case O(length (a) * length (b))
in case a lot of collisions.

Advanced or Extra'
* We can avoid worst case by Monte Carlo Variant

* Las Vegas Variant.

It can reduce all the false positives.

Code :-

```
public class KarpRabin {
  private Final int PRIME = 101;

 private double calculateHash (string str){
     double hash = 0;

     for(int i=0 ; i< str.length(); i++){
         hash = hash + str.charAt(i)*
                 Math.pow(PRIME,i);
     }
     return hash;
 }
```

```
private double updateHash(double prevHash, char
 Old char , char newChar , int patternLength){

    double newHash =(prevHash - oldchar)/PRIME ;
  newHash = newHash + newChar * Math.pow (PRIME,
                          patternLength - 1 );
    return  newHash ;
 }


 public void Search (String text, String pattern){
     int patternLength  =  pattern.length();
     double  patternHash =  calculateHash (pattern );
     double  textHash  =  calculateHash ( text.substring
                              (0, patternLength));

   for( int i =0 ; i<= text.length() - patternLength ;
          i++){     to not get out of bound

        if (textHash == patternHash){
            if (text.Substring (i, i+patternLength)
                   equals (pattern)){
            System.out.println ("Pattern found at
                index   " +  i) ,
            }
        }
```

```
            if (i < text.length() - patternLength){
                textHash = updateHash(textHash,
                text.charAt(i), text.charAt(i+patternLength),
                patternLength);
            }
        }
    }
}

Class Main {

    public static void main (String[] args){

        KarpRabin algo= new KarpRabin();

        algo.search("ApoorvKunalRahul",
                    "Kunal");
    }
}
```