A. If the hash value of $A_i A_{i+1} \ldots A_{i+k-1}$ is known, then the hash value of $A_{i+1} A_{i+2} \ldots A_{i+k}$ can be computed in constant time.

For example, say that f(X) is the hashing function of each given char in the string. For the string : abcdefg, if we wanted to find $A_{i+1} A_{i+2} \ldots A_{i+k}$ (Blue) and we have $A_i A_{i+1} \ldots A_{i+k-1}$ (Green), we can find blue by taking the green hash value, discarding (subtracting) the first term, and adding the last value of blue to give the new hash value. The formula is: Bhash value = Ghash value − f(String i) + f(String i+k)

Since this is a direct mathematical formula that doesn't change with regards to amount of characters in the string, we can say that the time complexity is O(1) or constant time.

B.

```java
public static int hashSearch(String pattern, String target) {
    int patternSize = pattern.length();
    boolean found = false;
    int patternIndex = -1;

    for (int i = 0; i < target.length() - patternSize + 1; i++) {
        String trimmedTarget = target.substring(i, i + patternSize);

        if (hashString(trimmedTarget) == hashString(pattern)) {
            patternIndex = i;

            //Verify that the match is correct
            if (target.substring(i, i + patternSize).equals(pattern)) {
                System.out.println("Found Pattern in Target at Index: " + i);
                found = true;
                break;
            } else {
                //Match is false
                patternIndex = -1;
                continue;
            }
        }
    }

    if (found == false) {
        System.out.println("Pattern Not Found.");
    }

    return patternIndex;
}
```

N

K

"Time Refuting False Matches"

Given a Pattern string of length K and a target string of size N, We can see that the length of both can affect runtime. Examples on the right.

We can see that as N increases, runs needed increases, and as K increases, runs needed decreases. Due to the Big-Oh rules, it would be appropriate to say that this algorithm therefore has an $O(K+N)$ complexity.

The time complexity, therefore, is linear.

Green: # of runs