MAC0323 ALGORITMOS E ESTRUTURAS DE DADOS II

FOLHA DE SOLUÇÃO

Nome: Lucas Quaresma Medina Lam **Número USP: 11796399**

Assinatura

Lucas Q. M. Lam

Sua assinatura atesta a autenticidade e originalidade de seu trabalho e que você se compromete a seguir o código de ética da USP em suas atividades acadêmicas, incluindo esta atividade.

Exercício: 7 **Data: 25/07/2021**

SOLUÇÃO

Sendo então 'a' a cadeia de caracteres onde queremos encontrar o sufixo mais longo que é prefixo de 'b', um possível algoritmo para achar a solução para o problema teria a seguinte abordagem:
A ideia é passar pelos caracteres das duas strings de trás para frente, colocando ponteiros no final de cada uma delas, i e j respectivamente, e verificando se os dois caracteres são iguais. Se os caracteres forem iguais, é guardado esse caractere e decrementado i e j. Se eles diferirem, os caracteres guardados são resetados e o ponteiro i volta para o final do caractere a, e é decrementado j para ir buscando um outro prefixo.
Ao final, se existir, teremos guardado a maior cadeia de caractere que é sufixo da palavra a e prefixo da palavra b, porém em ordem invertida, então passamos um for para inverter os caracteres e nos retornar a ordem certa.

Eis o código implementado em java:

```java
import edu.princeton.cs.algs4.StdOut;

public class SuffixPrefix
{
    private static String solveSP(String a, String b){
        int i = a.length()-1;
        int j = b.length()-1;

        String match = "";
        while(j >= 0 && i >= 0){
            if(a.charAt(i) == b.charAt(j)){
                match += b.charAt(j);
                i--;
            }
            else{
                match = "";
                i = a.length()-1;
            }
            j--;
        }
        if(match.length()>0){
            String result = "";
            for(int m = match.length()-1; m >= 0; m--){
                result += match.charAt(m);
            }
            return result;
        }
        else
            return match;
    }
```

Sabemos que o programa acima tem a complexidade pedida O(N+M), onde N e M são os comprimentos dos strings a e b, pois fazemos apenas 1 passagem pela string 'b' até encontrar o prefixo, e também fazemos uma passagem em tamanho proporcional ao sufixo da palavra 'a' para encontrar o resultado. E ao final disso, passamos uma vez pelo resultado encontrado invertendo seus caracteres, onde tudo acaba sendo majorado por O(N+M).