Análise de Algoritmos: https://www.ime.usp.br/~pf/analise_de_algoritmos/aulas/string-match.html

Izabela Ramos Ferreira - 2012130024

Exercícios 1

1. O algoritmo É-Sufixo produz resultado correto se n < m?

Não, porque na instrução:

2 enquanto $l \geq 1$ e $P[l] = T[k-m+l]$ faça

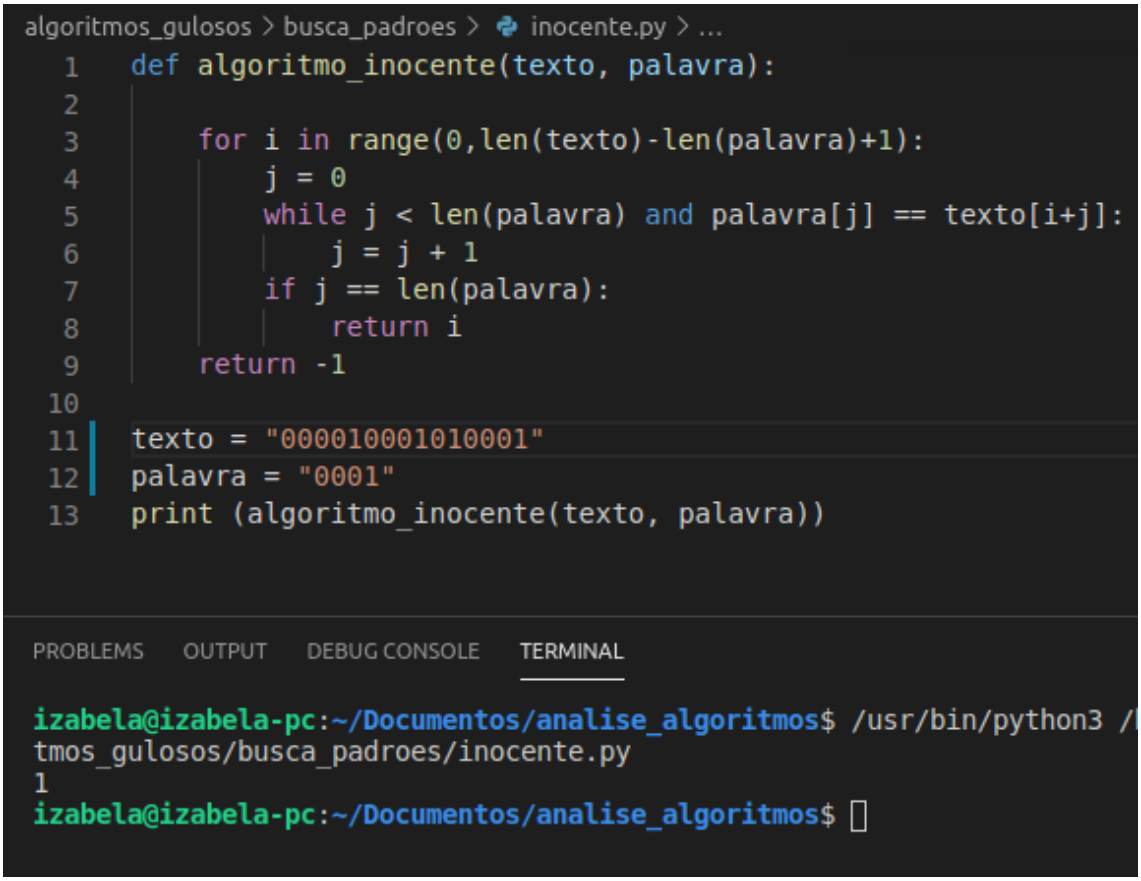
3 $l \leftarrow l - 1$

"l" não pode assumir valor negativo.

2. Que acontece se a linha 2 de É-Sufixo for trocada por "enquanto P[l] = T[k−m+l] e l ≥ 1 faça"?

Nada. Pois a operação lógica indicada é a "e" e a permutação de suas posições não vão alterar a tabela-verdade.

3. Aplique o algoritmo Inocente à palavra P = 0001 e ao texto T = 000010001010001.



Exercícios 2

1. Posso trocar a linha 2 por Busca-De-Palavra-Em-Texto por "para k crescendo de 1 até n−m+1 faça"?

Não pois n deve ser menor que m. Dessa forma, se permitido resultaria em valores negativos que não são válidos para índices visto que partem do 0.

2. Posso trocar a linha 2 por Busca-De-Palavra-Em-Texto "para k crescendo de m até n faça"?

Não, pois m representa o m-ésimo item da palavra e n o n-ésimo do texto. Dessa forma, m é menor que n.

Exercícios 3

1. A linha q = 0 da tabela D é realmente necessária?

Sim, para garantir que q seja o maior índice da tabela com a propriedade de ser sufixo da palavra e do texto. Ou seja, dado um q, q' e q", q" <= D[q,a] e q" <= q'.

Análise Matemática String-Search

1. Problema da busca de palavra em texto: Dados vetores P[1..m] e T[1..n], encontrar todos os valores de k para os quais P[1..m] é sufixo de T[1..k].

O que sabemos é:

P[1...m] = T[(**k** - m + 1) ... **k**] aonde m <= **k** e significa que para cada índice de P pode haver uma correspondência em T que se inicia de forma crescente de m até o **k**-ésimo item.
Assim:
P[m-1] = T[**k** - m + m - 1] = [**k**-1]
P[m] = T[**k** - m + m] = T[**k**]
P[1] = T[**k** - m + 1]

No caso do autômato:

P[1...q] é o maior prefixo de P que é sufixo de T[1...**k**−1], portanto P[1...q-1] consiste apenas em um prefixo menor que o maior prefixo de P e mantém intacta a propriedade T[1...**k**−1]

Portanto, o custo computacional considerando o pior e o melhor cenário, a grandeza **k** e se **k** = n, O(n).

O pré-processamento:

Para se definir o maior prefixo P[1..i] de P[1..q] o algoritmo recebe um elemento do alfabeto A e devolve o índice i. Tem-se que no pior caso:

$$i = q, q-1, \ldots, 1$$

O custo computacional se dá na **soma** dessa **sequência**:

$$q+(q-1)+\ldots+1$$

A jogada é lembrar que isso é uma **progressão aritmética finita** e a soma de dois termos equidistantes dos extremos é igual a soma dos extremos, ou seja, a soma da PA é dada pela soma dos extremos vezes a **metade** do número de termos já que em cada soma estão envolvidos dois termos:

$$[(a1 + an)*n]/2$$

Logo:

$$[(q+1)*q]/2 = (q^2 +2)/2$$

Portanto, o custo computacional considerando o pior cenário, a grandeza q, $O(q^2)$.

Porém, considerando o **consumo total** do pré-processamento, é importante destacar que **q** cresce de **0 até m** e que ambos representam uma sequência do menor ao maior prefixo da palavra com relação ao texto, como se fosse a sequência vezes ela mesma: $\mathbf{0^2 + 1^2 + 2^2 + 3^2 + ... + m^2}$ e essa **soma** representa o custo computacional do todo.

Por um método de indução finita, dá pra demonstrar que a sequência $\mathbf{0^2 + 1^2 + 2^2 + 3^2 + ... + m^2}$ pode ser reescrita considerando as seguintes propriedades:

Dado o produto notável $\mathbf{a^3 - b^3 = (a - b)*(a^2 + ab + b^2)}$:
Se a = m e b = m - 1, ou seja, b é antecessor de a:
Então $a^3 - b^3 = m^3 - (m - 1)^3 = (m - (m - 1))*(m^2 + m*(m - 1) + (m - 1)^2) = 1 * (m^2 + m^2 - m + m^2 - 2m + 1)$

Por isso $m^3 - (m - 1)^3 = 3m^2 - 3m + 1$ e cada execução representaria cada item dessa **soma**:

(Repara que a linha de baixo cancela o 1º item da linha de cima: $1^3 - 1^3$, $2^3 - 2^3$ ...)

para m = 1: $1^3 - 0^3 = 3*1^2 - 3*1 + 1$
para m = 2: $2^3 - 1^3 = 3*2^2 - 3*2 + 1$
para m = 3: $3^3 - 2^3 = 3*3^2 - 3*3 + 1$ +
para m = 4: $4^3 - 3^3 = 3*4^2 - 3*4 + 1$
para m = 5: $5^3 - 4^3 = 3*5^2 - 3*5 + 1$
[...]
para m: m^3 - $\mathbf{(m - 1)^3} = 3*m^2 - 3m + 1$ *[pela lógica o - **$(m - 1)^3$** também será cancelado restando apenas o m^3]*

$m^3 = 3*(1^2 + 2^2 + 3^2 + ... + m^2) - 3*(1 + 2 + 3 + ... + m) + 1*m$

$m^3 = 3*(1^2 + 2^2 + 3^2 + ... + m^2) - 3*\mathbf{[(m^2 + m)/2]} + m$

Como apresentei anteriormente a sequência **1 + 2 + 3 + ... + m** também é uma progressão aritmética finita e sua soma é dada por:

$$[(a1 + an)*n]/2$$

Logo **1 + 2 + 3 + ... + m** pode ser reescrito como $[(1+m)*m]/2 = \mathbf{(m^2 + m)/2}$

Isolando $\mathbf{1^2 + 2^2 + 3^2 + ... + m^2}$:

$$m^3 + 3*[(m^2 + m)/2] - m = 3*\mathbf{(1^2 + 2^2 + 3^2 + ... + m^2)}$$

Desse modo:
$\mathbf{1^2 + 2^2 + 3^2 + ... + m^2} = \{m^3 + 3*[(m^2 + m)/2] - m\} /3$ *[multiplica o numerador e o denominador por 2 para reduzir a expressão]*
$(2m^3 + 3m^2 + 3m - 2m)/6$
:. $(2\mathbf{m^3} + 3m^2 + m)/6$

Portanto, o custo computacional considerando o pior cenário, a grandeza m e a relação com um alfabeto A, $O(m^3|A|)$.

Implementações

Algoritmo Inocente:

https://github.com/MissHead/analise_algoritmos/blob/master/algoritmos_gulosos/busca_padroes/inocente.py

Algoritmo autômato:

https://github.com/MissHead/analise_algoritmos/blob/master/algoritmos_gulosos/busca_padroes/automata.py

Algoritmo Boyer Moore:

https://github.com/MissHead/analise_algoritmos/blob/master/algoritmos_gulosos/busca_padroes/boyer_moore.py

Algoritmo KMP:

https://github.com/MissHead/analise_algoritmos/blob/master/algoritmos_gulosos/busca_padroes/kmp.py