TDE 1 - Análise de complexidade de código não-recursivo

Marcello Fabrizio

7/9/2021

Trecho 1

```
for (i=0;i<n;i++) {
   for (j=i+1;j<n;j++) {
      if (m[i,1]>m[j,1]) {
         for (k=0;k<n;k++) {
           // Ops. Fundamentais
           aux=m[i,k];
           m[i,k]=m[j,k];
           m[j,k]=m[i,k];
         }
      }
   }
}
```

No trecho acima, considerei as três operações fundamentais como uma única só para facilitar os cálculos, pois a quantidade só influenciará na ordem de grandeza, e não na complexidade final. Na operação **for k**, teremos que ela executuá **n** vezes, então sua complexidade é de

$$O(n)$$

Como podemos re-organizar o próximo `for`, **for j** , em `for(j=i; j<n-1; j++)`, temos que ele irá executar $N - i$ vezes. Assim, a complexidade que temos até agora, multiplicando **for k** por **for j** é

$$O(N * (N - i))$$

Para o **for i**, como temos sua a complexidade interna, poderemos determinar a complexidade total do trecho. Devido a dependência do resultado com a variável de controle **i**, e tendo a complexidade de **for i** como $O(N)$, podemos obter o resultado com o seguinte somatório

$$\sum_{i=0}^{N} N * (N - i)$$

$$= N * (\sum_{i=0}^{N} N - \sum_{i=0}^{N} i)$$

Para

$$\sum_{i=0}^{N} N = N^2$$

Para

$$\sum_{i=0}^{N} i = (N^2 + N)/2$$

Então teremos

$$N^2 - (N^2 + N)/2$$

Portanto, temos que

$$N * \sum_{i=0}^{N} N - i = N * (N^2 - N)/2$$

e assim obteremos que a complexidade do trecho é igual a

$$O(N * (N^2 - N)/2)$$

Trecho 2

```
for (i=0;i<n;i++) {

      for (k=n-1;k>=i;k--) {
           // op fundamental
      }

      for (k=i+1;k<n;k++) {
            for (j=n-1;j>=i;j--){
                  // op fundamental
            }
      }

}
```

Para o `for` mais interno no trecho, **for j**, teremos que sua complexidade será de

$$O(N - i)$$

pois podemos reorganizar sua operção para `(j=i;j<n-1;j++)`. Para o **for k crescente**, temos a complexidade de

$$O(N - i)$$

Então para a complexidade de **for k crescente** com **for j**, teremos a multiplicação da complexidade das duas operações

$$O(N - i)^2$$

O segundo `for` mais externo **for k decrescente** também pode ser reorganizado como **for j**, obtendo assim a complexidade de

$$O(N - i)$$

Como **for k decrescente** e **for k crescente** não estão aninhados, a complexidade interna resultante de **for i** será a soma da complexidade de **for k decrescente** e **for k crescente**. Sendo a complexidade de **for i** igual a $O(N)$, como ambas operações `for` internas dependem da variável de controle **i**, o resultado será obtido atráves do somatório

$$\sum_{i=0}^{N} (N - i)^2 + (N - i)$$

$$= \sum_{i=0}^{N} (N - i)^2 + \sum_{i=0}^{N} (N - i)$$

Agora, com um pouco de álgebra, obteremos os seguintes resultados: Para $\sum_{i=0}^{N} (N - i)^2$ iremos obter

$$\sum_{i=0}^{N} (N^2 - 2Ni + i^2)$$

$$= \sum_{i=0}^{N} N^2 - 2N * \sum_{i=0}^{N} i + \sum_{i=0}^{N} i^2$$

Para

$$\sum_{i=0}^{N} N^2 = N^3$$

Para

$$2N * \sum_{i=0}^{N} i = 2N * (N^2 + N)/2$$
$$= N^3 + N^2$$

Para

$$\sum_{i=0}^{N} i^2 = N(N+1) * (2N+1)/6$$
$$= N^3/3 + N^2/2 + N/6$$

Assim temos

$$= N^3 - (N^3 + N^2) + N^3/3 + N^2/2 + N/6$$
$$= N^3 - N^3 - N^2 + N^3/3 + N^2/2 + N/6$$
$$= -N^2 + N^3/3 + N^2/2 + N/6$$

Assim(finalmente), temos que

$$\sum_{i=0}^{N} (N-i)^2 = -N^2 + N^3/3 + N^2/2 + N/6$$

Agora para $\sum_{i=0}^{N}(N-i)$ temos

$$\sum_{i=0}^{N} (N-i) = (N^2 - N)/2$$

Agora só precisamos somar os resultados obtidos

$$(N^2 - N)/2 - N^2 + N^3/3 + N^2/2 + N/6$$
$$= (2N^2 - N)/2 - N^2 + N^3/3 + N/6$$
$$= ((2N^2 - N) * 3)/6 + N^3 * 2/6 + N/6$$
$$= ((2N^2 - N) * 3 + N^3 * 2 + N)/6$$
$$= (2N^3 + 6N^2 - 2N)/6$$
$$= (2N(N^2 + 3 * N + -1)/6$$

Então chegamos a complexidade do trecho 2 sendo de

$$O((2N(N^2 + 3 * N + -1)/6)$$

Trecho 3

(consideremos a operação *log* como log_2)

```
void Moo(int N)
{
      for (j = 1; j <= N; j = j * 2)
            for (i = j; i > 1; i = i / 2)
                  printf("%d\n", i);
}
```

É possível re-organizar os laços do trecho da seguinte maneira:

```
void Moo(int N)
{
      for (i = 1; i <= N; i = i * 2)
            for (j = 2; j <= i; j = j * 2)
                 // op fundamental
}
```

Para resolver a complexidade do trecho de código acima, como suas variáveis de controle não são incrementadas for um a cada iterção, **i** e **j** serão substituidas por 2^y e 2^k respectivamente.

Assim vamos resolver a complexidade do **for j** interno

```
void Moo(int N)
{
      for (2^y = 1; 2^y <= N; y++)
            for (2^k = 2; 2^k <= 2^y; k++)
                  // op fundamental
}
```

E podemos simplificar ainda mais,

```
void Moo(int N)
{
      for (y = 0; y <= logN; y++)
            for (k = 1; k <= log2^y; k++)
                  // op fundamental
}
```

Dessa forma teremos o laço interno como

$$log_2 2^y = y$$

E temos o laço externo como

$$log_2 N$$

Assim teremos a complexidade interna de **for y** como

$$O(y)$$

como essa complexidade é dependente de uma variável de controle **y**, precisaremos resolver seu somatório

$$\sum_{y=0}^{log_2 N} y$$

$\sum_{y=0}^{log_2 N} y$ pode ser representado como uma progressão aritimética, assim teremos

$$(log_2 N^2 + log_2 N)/2$$

Sendo assim, a complexidade do trecho 3 é igual a

$$O((log_2 N^2 + log_2 N)/2)$$