Shell-Sort com intervalo $2^p.3^q$

Heitor Abreu
Lucas Yano
Vitor Rozeno
João Lucas Lima

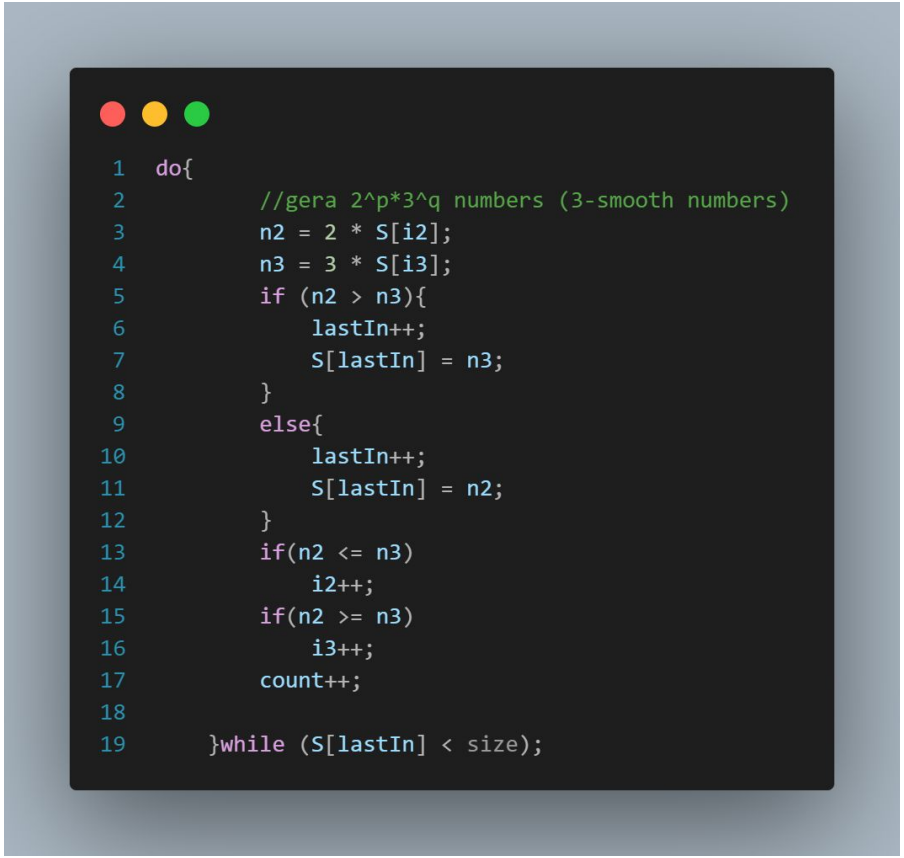
Introdução

Iremos realizar o Shellsort com intervalos da forma $2^p.3^q$, que geram a seguinte sequência de números:

1, 2, 3, 4, 6, 8, 9, 12…

Gerador da sequência

1, 2, 3, 4, 6, 8, 9, 12, 16, 18, 24, 27, 32, 36, 48, 54, 64, 72, 81, 96, 108, 128, 144, 162, 192, 216, 243, 256, 288, 324, 384, 432, 486, 512, 576, 648, 729, 768, 864, 972, 1024, 1152, 1296, 1458, 1536, 1728, 1944, 2048, 2187, 2304, 2592, 2916, 3072, 3456, 3888…



```
do{
        //gera 2^p*3^q numbers (3-smooth numbers)
        n2 = 2 * S[i2];
        n3 = 3 * S[i3];
        if (n2 > n3){
            lastIn++;
            S[lastIn] = n3;
        }
        else{
            lastIn++;
            S[lastIn] = n2;
        }
        if(n2 <= n3)
            i2++;
        if(n2 >= n3)
            i3++;
        count++;

    }while (S[lastIn] < size);
```

Análise Assintótica

Os intervalos que usaremos, causarão $\log(N)^2$ ciclos do algoritmo. Cada ciclo com N + 5 unidades de tempo, fazendo com que o tempo total seja de $N \log(N)^2 + 5 \log(N)^2$. Podemos então afirmar que esse algoritmo tem complexidade de $\Theta(N^*\log(N)^2)$.

```
1  do{
2          lastIn--;
3          for(i = S[lastIn]; i < size; i++) {
4              value = a[i];
5              j = i - S[lastIn];
6
7              while (j >= 0 && value < a[j]) {
8                  a[j + S[lastIn]] = a[j];
9                  j -= S[lastIn];
10             }
11
12             a[j + S[lastIn]] = value;
13         }
14     }while(S[lastIn] > 1);
15
```

Análise Assintótica

A prova matemática mais detalhada pode ser encontrada no arquivo abaixo, na proposição 4:

https://math.mit.edu/~poonen/papers/shell.pdf

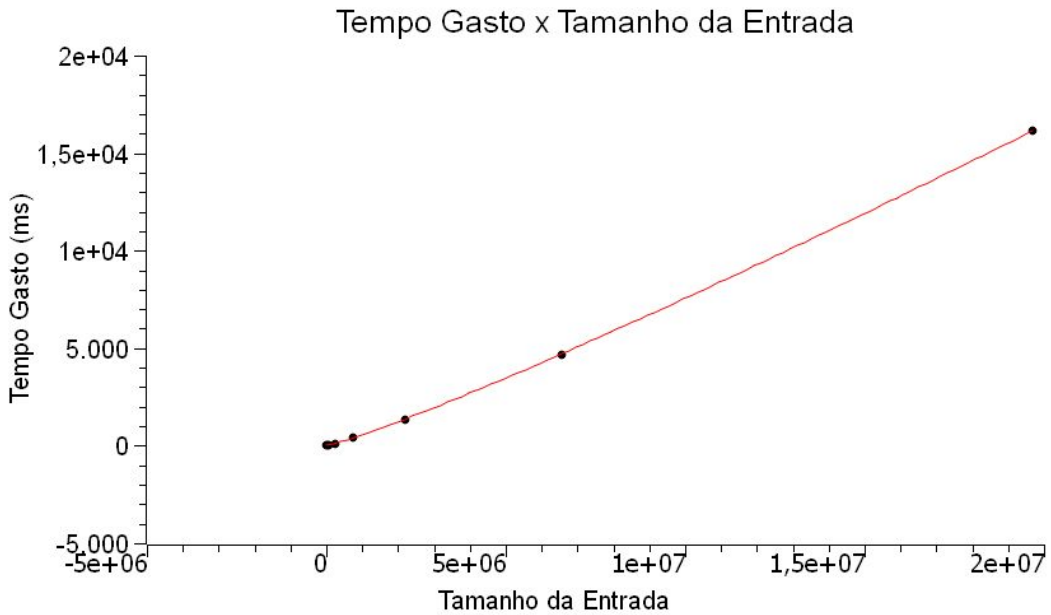
O(N(log N)²)

Ω(N(log N)²)

$$\Theta\left(N \log^2 N\right)$$

Tempo gasto x Tamanho da entrada

Non-linear fit of dataset: Table2_2, using function: ((x * log2(x) * log2(x)) * a) + b
Y standard errors: Unknown
Scaled Levenberg-Marquardt algorithm with tolerance = 0,0001
From x = 1.000 to x = 19.683.000
a = 1,39621041044348e-06 +/- 5,58517407074004e-10
b = 2,38500013793707 +/- 2,13331216386394



Pelo gráfico, vemos que o tempo gasto pelo algoritmo se aproxima do tempo previsto por **N * log(N)^2 .**

Usando o sciDavis para fazer a análise do dos pontos, chegamos que a função que mais se aproxima do tempo é:

T = (N * log(N) * log(N) * 0,00000139) + 2,385.

Referências

1. Pratt, Vaughan Ronald (1979). *Shellsort and Sorting Networks (Outstanding Dissertations in the Computer Sciences)* (PDF). Garland. ISBN 978-0-8240-4406-0. Archived (PDF) from the original on 7 September 2021.
2. Poonen,B. . **THE WORST CASE IN SHELLSORT AND RELATED ALGORITHMS** . Math.Mit.Edu, Disponível em: https://math.mit.edu/~poonen/papers/shell.pdf Acesso em: 23 nov. 2022.