Instituto Tecnológico de Aeronáutica - ITA
CSC-64 - Programação Paralela
Aluno: Ulisses Lopes da Silva

Relatório do Laboratório 01 - Jogo da Vida

---

1 Complexidade do programa

Observando atentamente cada um dos módulos que compõem o programa, verificamos a existência de 3 arquivos principais, que são: `ModVida.c`, `Tempo.c` e `wall_time.c`. Não obstante, cada módulo possui funções com diferentes tempos de execução, os quais serão detalhadas abaixo. Para efeito de cálculo, sempre consideraremos o pior caso.

1.1 `ModVida.c`

- Função `UmaVida()`:
  - 3 atribuições;
  - repete `tam` vezes [repete `tam` vezes (9 atribuições + 4 comparações)];

  Assim, temos: $3 + n(n.13) = 13n^2 + 3 \Rightarrow \mathrm{O}(n^2)$.

- Função `DumpTabul()`:
  - 5 atribuições;
  - 1 print;
  - (`last - first`) prints (k prints);
  - repete `n` vezes [repete `n` (1 print) + 1 print];
  - repete `n` vezes 1 print;

  Assim, temos: $5 + 1 + k + n(n.(1) + 1) + n.(1) = n^2 + 2\mathrm{n} + \mathrm{k} + 6 \Rightarrow \mathrm{O}(n^2)$.

- Função `InitTabul()`:
  - 3 atribuições;
  - 1 operação aritmética;
  - repete `n` vezes (2 atribuições);
  - 5 atribuições;

  Assim, temos: $3 + 1 + n(2) + 5 = 2n + 9 \Rightarrow \mathrm{O}(n)$

- Função `Correto()`:
  - 2 atribuições;
  - 2 atribuições;
  - repete `n` vezes (1 atribuição);
  - 6 retornos;

  Assim, temos: $4 + n(1) + 6 = n + 10 \Rightarrow O(n)$

1.2 `wall.c`:

Neste arquivo, há apenas a definição de uma struct, que podemos considerar como tempo constante.

1.3 `Tempo.c`:

- Função `main`:
  - repete (`POWMAX - POWMIN`) (k) vezes (1 atribuição + `n` vezes ($n^2 + n^2$);
  - 1 atribuição;
  - 1 print;

  Assim, temos: $k(1 + n(n^2 + n^2)) + 1 + 1 = k + 2k(n^3) + 2 \Rightarrow O(n^3)$.

  Portanto, **a complexidade de tempo total do programa é da ordem de n^3, ou** $O(n^3)$.

2 Análise da complexidade e Gráfico comparativo

Após a execução do código, foi retornado o arquivo `OutTempo`, que continha os tempos de compilação para entradas múltiplas de 2^n, com n variando de 3 a 9. Os tempos foram:

Tabela 1: Tamanho da entrada (n) vs. tempo de execução (s)

n	**Tempo (s)**
8	0.000020
16	0.000178
32	0.001576
64	0.012339
128	0.100733
256	0.815215
512	6.553873

Note que:

$$T(n) \approx C.n^3 \Rightarrow T(2n) \approx 2^3.C.n^3$$

Logo, se n dobra, podemos obter uma razão aproximada de 2^3. Percerba que:

$$T(16)/T(8) = 0.000178/0.000020 \approx 8.9$$
$$T(32)/T(16) = 0.001576/0.000178 \approx 8.85$$
$$T(64)/T(32) = 0.012339/0.001576 \approx 7,83$$
$$T(128)/T(64) = 0.100733/0.012339 \approx 8.16$$
$$T(256)/T(128) = 0.815215/0.100733 \approx 8.09$$
$$T(512)/T(256) = 6.553873/0.815215 \approx 8.04$$

O que confirma a teoria.

Além do exemplificado, potando o gráfico com os tempos, podemos verificar, conforme previsto na estimativa de complexidade da seção anterior, que o tempo gasto varia segundo $O(n^3)$.

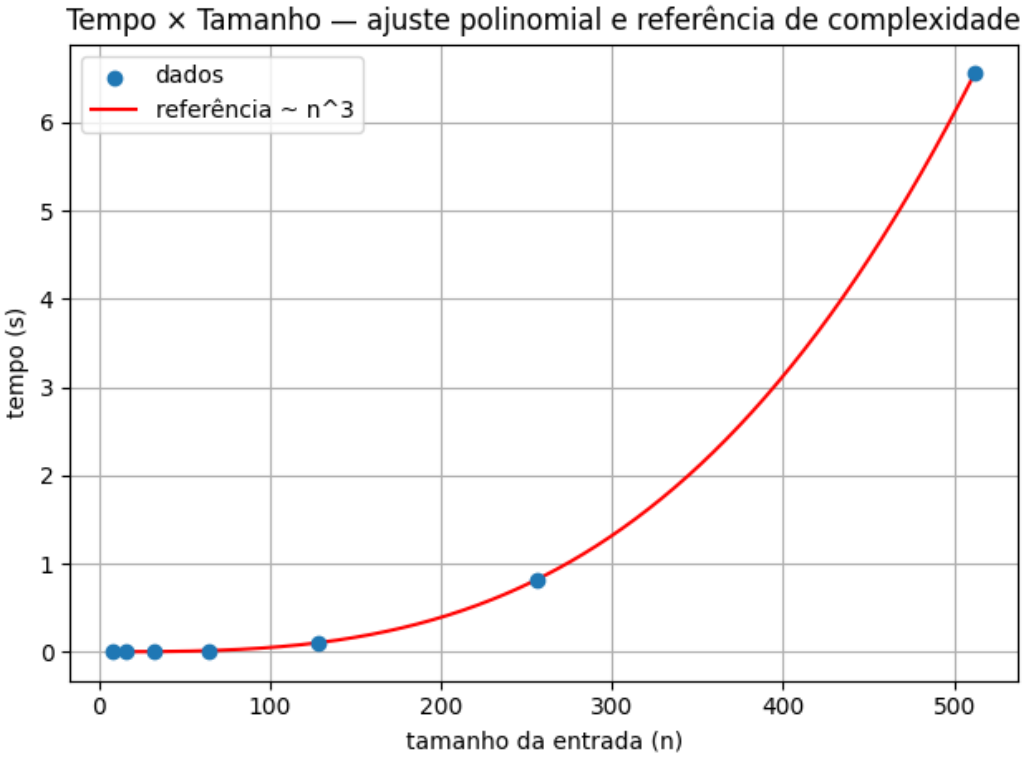

Fig. 1: Gráfico dos pontos obtidos e da curva ajustada a eles.