# Projeto e Análise de Algoritmos (2022-1)

Seleção de exercícios para a segunda avaliação

Prof. Flávio L. C. de Moura

1 de setembro de 2022

1. Mostre como podemos ordenar $n$ inteiros contidos no intervalo de 0 a $n^2 - 1$ em tempo linear, ou seja, em $O(n)$.

2. Mostre como podemos ordenar $n$ inteiros contidos no intervalo de 0 a $n^3 - 1$ em tempo linear, ou seja, em tempo $O(n)$.

3. Prove que o algoritmo counting-sort é estável.

```
1  let C[0..h − l] be a new array;
2  for i = 0 to h − l do
3  |  C[i] ← 0;
4  end
5  for i = 0 to n − 1 do
6  |  C[A[i] − l] ← C[A[i] − l] + 1;
7  end
8  for j = 1 to h − l do
9  |  C[j] ← C[j] + C[j − 1];
10 end
11 for i = n − 1 downto 0 do
12 |  j ← A[i] − l;
13 |  B[C[j] − 1] ← A[i];
14 |  C[j] ← C[j] − 1;
15 end
16 return B;
```

**Algorithm 1:** counting-sort($A[0..n-1], l, h$)

4. Prove que o algoritmo *insertion sort* é estável.

```
1 for  j = 1 to n − 1 do
2 |  key ← A[j];
3 |  i ← j − 1;
4 |  while  i ≥ 0 and A[i] > key do
5 |  |  A[i + 1] ← A[i];
6 |  |  i ← i − 1;
7 |  end
8 |  A[i + 1] ← key;
9 end
```

**Algorithm 2:** InsertionSort($A[0..n-1]$)

5. Prove que o algoritmo *merge sort* é estável.

```
1  if p < r then
2  |  q = ⌊(p+r)/2⌋;
3  |  mergesort(A, p, q);
4  |  mergesort(A, q + 1, r);
5  |  merge(A, p, q, r);
6  end
```

**Algorithm 3:** mergesort$(A, p, r)$

onde

```
1  n1 = q − p + 1 ;                          // Qtd. de elementos em A[p..q]
2  n2 = r − q ;                              // Qtd. de elementos em A[q + 1..r]
3  let L[1..n1 + 1] and R[1..n2 + 1] be new arrays;
4  for i = 1 to n1 do
5  |  L[i] = A[p + i − 1];
6  end
7  for j = 1 to n2 do
8  |  R[j] = A[q + j];
9  end
10 L[n1 + 1] = ∞;
11 R[n2 + 1] = ∞;
12 i = 1;
13 j = 1;
14 for k = p to r do
15 |  if L[i] ≤ R[j] then
16 |  |  A[k] = L[i];
17 |  |  i = i + 1;
18 |  end
19 |  else
20 |  |  A[k] = R[j];
21 |  |  j = j + 1;
22 |  end
23 end
```

**Algorithm 4:** merge$(A, p, q, r)$

6. Considere uma enumeração qualquer $1, 2, \ldots, |V|$ dos vértices de $G$. A matriz de adjacências $G.A$ de dimensão $|V| \times |V|$ é dada por:

$$G.A[i][j] = \begin{cases} 1, & \text{se } (i,j) \in G.E \\ 0, & \text{caso contrário.} \end{cases} \tag{1}$$

O pseudocódigo a seguir apresenta o algoritmo BFS onde o grafo $G$ é representado por sua matriz de adjacências:

```
1  for i = 1 to |V| do
2  |  i.color ← WHITE;
3  end
4  s.color ← GRAY;
5  Q ← ∅;
6  enqueue(Q, s);
7  while Q ≠ ∅ do
8  |  u ← dequeue(Q);
9  |  for i = 1 to |V| do
10 |  |  if G.A[u][i] = 1 and i.color = WHITE then
11 |  |  |  i.color ← GRAY;
12 |  |  |  enqueue(Q, i);
13 |  |  end
14 |  end
15 end
```

**Algorithm 5:** BFS($G, s$)

Qual é a complexidade de tempo de BFS neste caso?

7. Escreva o pseudocódigo para o algoritmo DFS onde o grafo $G$ dado como argumento é representado por sua matriz de adjacências, e em seguida faça a análise assintótica do seu pseudocódigo.

8. Mostre que 2-SAT $\in$ P.

9. Assumindo que SAT é um problema NP-completo, mostre que 3-SAT é NP-completo. Isto é, mostre que 3-SAT $\in$ NPC.

10. Mostre que CLIQUE $\in$ NPC.

11. Mostre que VERTEX-COVER $\in$ NPC.

12. Mostre que HAMPATH $\in$ NPC.