Universidade Federal de Ouro Preto PCC104 - Projeto e An´alise de Algoritmos
For¸ca Bruta e Busca Exaustiva
Prof. Rodrigo Silva
Neivaldo I. Matos Filho

1 - Algoritmo SelectSort

```
def selectsort(array):
    for i in range(0, len(array)):
        min_i = i
        for j in range(i + 1, len(array)):
            if array[j] < array[min_i]:
                min_i = j
        array[i], array[min_i] = array[min_i], array[i]
```

Expressão: PA (a1, a2, a3, a4, a5,.. an-1, an)
Operação básica = IF(comparação)
Custo de comparação =
C = (n -1 + 1)/2 = n(n-1)/2 = (n2 - n)/2 ϵ O(n)

$$C(n) = \sum_{i=0}^{n-2} \sum_{j=i+1}^{n-1} 1 = \sum_{i=0}^{n-2} (n-1-i) = \frac{(n-1)n}{2}.$$

Tipo = algoritmo de ordenação por seleção Θn^2 nas entradas e na troca Θn (n-1)

2 - Algoritmo busca sequencial

```
def sequencialBusca(vetor, k):
    i = 0
    while i < len(vetor):
        if vetor[i] == k:
            return i
        i += 1
    return -1
```

Expressão: funçaõ fatorial ("n! = n · (n-1) · (n-2) … 3 · 2 · 1")

$$
\begin{aligned}
C_{avg}(n) &= [1 \cdot \frac{p}{n} + 2 \cdot \frac{p}{n} + \cdots + i \cdot \frac{p}{n} + \cdots + n \cdot \frac{p}{n}] + n \cdot (1 - p) \\
&= \frac{p}{n}[1 + 2 + \cdots + i + \cdots + n] + n(1 - p) \\
&= \frac{p}{n}\frac{n(n+1)}{2} + n(1 - p) = \frac{p(n+1)}{2} + n(1 - p).
\end{aligned}
$$

Operação básica = while(repetição até encontrar elemento)

Custo de comparação:
melhor caso C(n) = 1
pior caso C(n) = n
caso médio C(n) = (n+1)/2

Tipo = On
LINEAR

3 - Algoritmo busca em largura

```
def BFS(self, s):
    visited = [False] * (len(self.graph))
    queue = []
    queue.append(s)
    visited[s] = True
    while queue:
        s = queue.pop(0)
        for i in self.graph[s]:
            if visited[i] == False:
                queue.append(i)
                visited[i] = True
```

Expressão:

$$T(h, m) = \sum_{i=0}^{h-1} m^i = \frac{m^h - 1}{m - 1}$$

Função de custo:

$$T(h, m) = \frac{m^h - 1}{m - 1}$$

Eficiência: $O(m^h)$

4 - Algoritmo Busca em profundidade

```
def iterativeDFS(graph, v, discovered):
    stack = deque()
    stack.append(v)
    while stack:
        v = stack.pop()
        if discovered[v]:
            continue
        discovered[v] = True
        print(v, end=' ')
        adjList = graph.adjList[v]
        for i in reversed(range(len(adjList))):
            u = adjList[i]
            if not discovered[u]:
                stack.append(u)
```

Expressão:

$$T(h, m) = \sum_{i=0}^{h-1} m^i = \frac{m^h - 1}{m - 1}$$

Função de custo:

$$T(h, m) = \frac{m^h - 1}{m - 1}$$

Eficiência: $O(m^h)$