MINISTÉRIO DA EDUCAÇÃO
Universidade Federal de Ouro Preto
Departamento de Computação e Sistemas
Campus João Monlevade

NOÇÕES DE ANÁLISE DE COMPLEXIDADE DE ALGORITMOS

Curso: Engenharia Elétrica/Sistemas de Informação — 1º semestre de 2024
Disciplina: Algoritmos e Estrutura de Dados I — Data: 10/04/24
Professor: Alexandre Magno de Sousa

1 FUNÇÕES DE CUSTO E ANÁLISE ASSINTÓTICA

1. Encontre a função de custo associada à complexidade de tempo para a instrução de interesse indicada em cada trecho de código a seguir.

   (a)
```c
void function(int n){
    if (n == 1)
        return;
    for (i = 1; i <= n; i++){
        for (j = 1; j <= n; j++){
            printf("*"); // <-- instrução de interesse
            break;
        }
    }
}
```

   (b)
```c
int count = 0;
for (i = 0; i < n; i++) {
   if (i % 2 == 0) {
      for (j = 0; j < n; j++)
         count++;  // <-- instrução de interesse
   }
   else {
      for (j = 0; j < 10; j++)
         count++;  // <-- instrução de interesse
   }
}
```

   (c)
```c
for(i = 0; i < n; i++){
   for (j = n; j > i+1; j--){
      printf("plus");  // <-- instrução de interesse
      printf("minus"); // <-- instrução de interesse
   }
}
```

   (d)
```c
int count = 0;
for (i = 0; i < n; i++) {
   for (j = i; j < 2*n; j++) {
      count++;  // <-- instrução de interesse
   }
}
```

   (e)
```c
for (i = 0; i < n; i++) {
   for (j = i+1; j < n/2; j++) {
      printf("fee"); // <-- instrução de interesse
      printf("fi");  // <-- instrução de interesse
   }
}
```

(f) Para o trecho de código a seguir, assuma que n é uma potência de 2, ou seja 2^m

```
for (i = 1; i < n/4; i *= 2) {
   printf("stop short"); // <-- instrução de interesse
}
for (i = 0; i < 2*n; i += 2) {
   printf("more stuff"); // <-- instrução de interesse
}
```

(g)
```
for (i = 0; i < n; i++)
    for (j = i+1; j < n; j++)
        for (k = j+1; k < n; k++)
                count++;  // <-- instrução de interesse
```

(h) Para o trecho de código a seguir, assuma que n é uma potência de 3, ou seja 3^m

```
for (i = 1; i < n; i++) {
    count++; // <-- instrução de interesse
}
for (j = 1; j < n; j *= 3) {
    count++; // <-- instrução de interesse
}
```

(i) Para o trecho de código a seguir, assuma que n é uma potência de 2, ou seja 2^m

```
int count = 0;
for (i = n/2; i < n; i++)
    for (j = 1; j < n; j = 2*j)
        for (k = 1; k < n; k *= 2)
            count++;  // <-- instrução de interesse
```

(j) Para o trecho de código a seguir, assuma que n é uma potência de 2, ou seja 2^m

```
int count = 0;
for (i = n/2; i <= n; i++)
    for (j = 1; j + n/2 <= n; j++)
        for (k = 1; k <= n; k = k*2)
            count++; // <-- instrução de interesse
```

(k)
```
int i = 1;
int s = 1;
while (s <= n) {
   i++;
   s += i;
   printf("*"); // <-- instrução de interesse
}
```

2. **Busca Bitônica**: Um vetor é bitônico se ele é formado por uma sequência crescente de inteiros seguida imediatamente por uma sequência decrescente de inteiros. Faça uma função que, dado um vetor bitônico de n inteiros distintos, determine se um dado inteiro está dentro do vetor. A função deverá utilizar aproximadamente $3 \log_2 n$ comparações para o pior caso.

   DICA: utilize uma versão adaptada da busca binária para encontrar o valor máximo em $\log_2 n$ comparações, então utilize a busca binária para pesquisar em cada partição do vetor em $\log_2 n$ comparações para cada partição.

3. Assuma que cada uma das expressões a seguir representa a função de custo de complexidade por um algoritmo para resolver um problema de tamanho n. Selecione o termo dominante da expressão que tem o crescimento mais acentuado em função de n e especifique o limite assintótico superior mais próximo possível da função de custo de complexidade de algoritmo e complete a Tabela 1.

Tabela 1: Expressões de funções de custo de complexidade de algoritmos.

Expressão	**Maior Termo**	**Limite Superior**
$5 + 0.001n^3 + 0.025n$		
$500n + 100n^{1.5} + 50n \log_{10} n$		
$0.3n + 5n^{1.5} + 2.5n^{1.75}$		
$n^2 \log_2 n + n(\log_2 n)^2$		
$n \log_3 n + n \log_2 n$		
$3 \log_8 n + \log_2 \log_2 \log_2 n$		
$100n + 0.01n^2$		
$0.01n + 100n^2$		
$2n + n^{0.5} + 0.5n^{1.25}$		
$0.01n \log_2 n + n(\log_2 n)^2$		
$100n \log_3 n + n^3 + 100n$		
$0.003 \log_4 n + \log_2 \log_2 n$		

4. Para cada função a seguir encontre os pares de valores para c e n_0 para cada função de acordo com o limite assintótico superior e inferior, isto é, O-grande e Ω-grande:

   (a) $f(n) = 3n^3 + 2n^2 + n$.

   (b) $f(n) = 2n^2 + 4n - 15$.

   (c) $f(n) = 3n + 7$.

   (d) $g(n) = 3n^2 + 1$.

   (e) $f(n) = 5 \log_2 n$.

   (f) $f(n) = 2^n + n^3 + n(\log_2 n)^2$.

   Agora utilizando um programa de planilhas como, por exemplo, Microsoft Excel[1] ou Google Sheets[2], apresente gráficos para cada uma das funções anteriores para os limites assintóticos superior (O-grande) e inferior (Ω-grande) para valores de $n \in [0, 10^2]$.

   ATENÇÃO: verifique se as constantes c e n_0 encontradas estão corretas, ou seja:

   - A linha do gráfico referente ao limite superior mostra que a inequação para O-grande é verdadeira para todo $n \geq n_0$?
   - Ou, a linha do gráfico que representa o limite inferior mostra que a inequação para Ω-grande é verdadeira pra todo $n \geq n_0$?

[1] https://www.microsoft.com/pt-br/microsoft-365/excel
[2] https://www.google.com/sheets/about/

5. Qual das seguintes afirmações sobre o crescimento assintótico das funções não é verdadeira:
   (a) $2n^2 + 3n + 1 = O(n^2)$.
   (b) $\log n^2 = O(\log n)$.
   (c) Se $f(n) = O(g(n))$ e $g(n) = O(h(n))$, então $f(n) = O(h(n))$.
   (d) Se $f(n) = O(g(n))$, então $g(n) = O(f(n))$
   (e) $2^{n+1} = O(2^n)$.
   (f) $2^{2n} = O(2^n)$.
   (g) $f(n) = O(u(n))$ e $g(n) = O(v(n))$ então $f(n) + g(n) = O(u(n) + v(n))$.
   (h) $f(n) = O(u(n))$ e $g(n) = O(v(n))$ então $f(n) - g(n) = O(u(n) - v(n))$.

6. As afirmações apresentas na Tabela 2 mostram algumas propriedades da notação O-grande para as funções $f \equiv f(n)$, $h \equiv h(n)$ e $g \equiv g(n)$. Determine se cada afirmação é `VERDADEIRA` ou `FALSA` e, neste caso, quando a afirmação for falsa, faça a correção da fórmula.

Tabela 2: Afirmações da notação O-grande para as funções f e g.

Afirmação	É `VERDADEIRA` ou `FALSA`?	Se a afirmação é `FALSA`, então escreva a fórmula correta
Regra das somas: $O(f + g) = O(f) + O(g)$		
Regra dos produtos: $O(f \times g) = O(f) \times O(g)$		
Transitividade: se $g = O(f)$ e $h = O(f)$, então $g = O(h)$		
$5n + 8n^2 + 100n^3 = O(n^4)$		
$5n + 8n^2 + 100n^3 = O(n^2 \log n)$		