Universidade de São Paulo - Instituto de Ciências Matemáticas e de Computação
SCC0220 - Laboratório de Introdução à Ciência da Computação II
Aluno: Gabriel Toschi de Oliveira (9763039)

Atividade 3

Análise formal de buscas

Introdução

Nesta atividade, foram realizadas as análises formais de contagem de operações para três algoritmos de busca conhecidos: busca sequencial, busca binária iterativa e busca binária recursiva.

Para as análises formais, serão usados padrões para representar certos tipos de instrução, de forma a aproximá-las: **A** para operações aritméticas e atribuições, **C** para comparações, **M** para acessos à memória e **F** para chamadas e retornos de funções. Durante as simplificações, o símbolo **X** será usado para uma instrução qualquer. O símbolo **n** será usado para representar o tamanho do vetor.

Algoritmo 1: busca sequencial

```
int sequentialSearch(int* data, int size, int searchQuery) { // F
 for (int index = 0; index < size; index++) {                // A+C [1a it], 2A+C [2a it+]
   if (data[index] == searchQuery) {                         // M+C
     return index;                                           // F
   }
 }

 return NOT_FOUND;
}
```

Pior caso do algoritmo

Elemento buscado não está presente ou está na última posição do array.

Equação de contagem de operações

F + A + C + n(2A + C + M + C) + F

= **(2)F + (2n + 1)A + (2n + 1)C + (n)M**

Simplificação da equação (F = M = A = C = X)

(2)F + (2n + 1)A + (2n + 1)C + (n)M

= (2)X + (2n + 1)X + (2n + 1)X + (n)X

= **(5n + 4)X**

Complexidade de tempo

Linear, O(n).

Algoritmo 2: busca binária iterativa

```
int iterativeBinarySearch(int* data, int size, int searchQuery) { // F
  int left = 0;                                                   // A
  int right = size - 1;                                           // 2A

  while (left < right) {                                          // C
    int checkingIndex = (left + right) / 2;                       // 3A

    if (searchQuery == data[checkingIndex]) {                     // M+C
      return checkingIndex;                                       // F
    }

    if (searchQuery < data[checkingIndex]) {                      // M+C
      right = checkingIndex - 1;                                  // 2A
    } else { // data[checkingIndex] > searchQuery
      left = checkingIndex + 1;                                   // 2A
    }
  }

  return NOT_FOUND;                                               // F
}
```

Pior caso do algoritmo

Elemento buscado não está presente ou está na primeira posição do array.

Equação de contagem de operações

$F + A + 2A + \log_2 n * (C + 3A + M + C + M + C + 2A) + F$

$= \mathbf{(2)F + (3 + 5\log_2 n)A + (3\log_2 n)C + (2\log_2 n)M}$

Simplificação da equação (F = M = A = C = X)

$(2)F + (3 + 5\log_2 n)A + (3\log_2 n)C + (2\log_2 n)M$

$= (2)X + (3 + 5\log_2 n)X + (3\log_2 n)X + (2\log_2 n)X$

$= \mathbf{(10\log_2 n + 5)X}$

Complexidade de tempo

Logarítmica, O(log n)

Algoritmo 3: busca binária recursiva

```
int binSearch(int *data, int left, int right, int query) {      // F
 int checkingIndex = (left + right) / 2;                        // 3A

 if (right >= left) {                                           // C
   if (data[checkingIndex] == query) {                          // M+C
     return checkingIndex;                                      // F
   }

   if (query < data[checkingIndex]) {                           // M+C
     return binSearch(data, left, checkingIndex - 1, query);    // F+5A
   } else { // data[checkingIndex] > query
     return binSearch(data, checkingIndex + 1, right, query);   // F+5A
   }
 }

 return NOT_FOUND;                                              // F
}
```

Pior caso do algoritmo

Elemento buscado não está presente.

Equação de contagem de operações

T(N) = F + 3A + C + M + C + M + C + F + 5A + T(N/2)
T(1) = F + 3A + C + F

T(N) = 2F + 8A + 3C + 2M + T(N/2)
T(1) = 2F + 3A + C

Simplificação da equação (F = M = A = C = X)

T(N) = 2X + 8X + 3X + 2X + T(N/2)
T(1) = 2X + 3X + X

T(N) = 15X + T(N/2)
T(1) = 6X

Forma fechada da equação de recorrência

$T(N) = 15X * k + T(b/2^k)$

Caso base

$b/2^k = 1$, $b = 2^k$, **$k = \log_2 b$**

Com $k = \log_2 n$ e $T(1) = 6X$

$T(N) = 15X * \log_2 n + T(1)$

$T(N) = 15X * \log_2 n + 6X$

Complexidade de tempo

Logarítmica, O(log n)