Universidade Federal de Ouro Preto
PCC104 - Projeto e An´alise de Algoritmos
Diminuir e Conquistar
Prof. Rodrigo Silva
April 17, 2023

Neivaldo I. Matos Filho

Atividades Práticas
1. Implemente o algoritmo de busca binária.

```cpp
int binarySearch(vector<int> v, int x) {
    int left = 0;
    int right = v.size()-1;

    while (left <= right) {
        int mid = (left + right)/2;

        if (v[mid] == x) {
            return mid;
        }
        else if (v[mid] < x) {
            left = mid + 1;
        }
        else {
            right = mid - 1;
        }
    }
    return -1;
}
```

Expressão matemática que define o custo do algoritmo:

$$T(n) = T(n/2) + 1$$

• Cálculo da função de custo: **h = log2 n**
para n = 32 > temos $\log_2 32 = 5$.

• Indicação da classe de eficiência (O ou Θ): O (log n)

n	$\log_2 n$
1	0
2	1
4	2
8	3
16	4
32	5

2. Implemente o método interpolation search.

```
1  int interpolationSearch(int A[], int n, int target)
2  {
3      if (n == 0) {
4          return -1;
5      }
6      int low = 0, high = n - 1, mid;
7
8      while (A[high] != A[low] && target >= A[low] && target <= A[high])
9      {
10         mid = low + ((target - A[low]) * (high - low) / (A[high] - A[low]));
11
12         if (target == A[mid]) {
13             return mid;
14         }
15         else if (target < A[mid]) {
16             high = mid - 1;
17         }
18         else {
19             low = mid + 1;
20         }
21     }
22     if (target == A[low]) {
23         return low;
24     }
25
26     else {
27         return -1;
28     }
29 }
```

• Expressão matemática que define o custo do algoritmo: $$x = l + \left\lfloor \frac{(v - A[l])(r - l)}{A[r] - A[l]} \right\rfloor .$$

Função do tipo linear f(x) = **a.x + b.**

• Cálculo da função de custo:
v = [2, 3, 4, 5, 9, 20]

v = 5 > x = 0 + (5 – 2) * (5-0/20-2) > = 3 * (5/18) > = 0,8 → 1.

• Indicação da classe de eficiência (O ou Θ):
melhor caso: O (log(log n)) (Elementos distribuídos uniformemente)
Pior caso: O(n) (Valores numéricos dos alvos aumentam exponeciamente)

3. Implemente a estrutura de dados binary search tree e os métodos buscar e inserir.

```cpp
class Node {
public:
    int key;
    Node* left;
    Node* right;

    Node(int key) {
        this->key = key;
        left = right = NULL;
    }
};

class BST {
private:
    Node* root;

public:
    BST() {
        root = NULL;
    }

    Node* search(Node* node, int key) {
        if (node == NULL || node->key == key) {
            return node;
        }
        if (node->key < key) {
            return search(node->right, key);
        }
        return search(node->left, key);
    }
    Node* insert(Node* node, int key) {
        if (node == NULL) {
            return new Node(key);
        }
        if (key < node->key) {
            node->left = insert(node->left, key);
        } else if (key > node->key) {
            node->right = insert(node->right, key);
        }
        return node;
    }

    void insert(int key) {
        root = insert(root, key);
    }

    Node* search(int key) {
        return search(root, key);
    }
};

```

A estrutura de nó deve conter um ponteiro para o **filho esquerdo**, um **ponteiro para o filho direito** e o elemento em si.

• Expressão matemática que define o custo do algoritmo:
$T(n) = T(n/2) + 1$

• Cálculo da função de custo: O(log n)

• Indicação da classe de eficiência (O ou Θ):
Em uma árvore balanceada, a altura é O(log n), onde n é o número de nós na árvore. No entanto, em uma árvore não balanceada, a altura pode ser O(n), onde n é o número de nós na árvore.