Centro de
Investigación en
Computación

Searching

Búsqueda

Algoritmos

Complejidad

Course

Operating System (with focus on Security)

Instructor

Acosta Bermejo Raúl et al.

Lecture notes

Instituto
Politécnico
Nacional

Table of contents (outline)

Tabla de contenido

1. Introduction
2. Secuential search
3. Sorted data
    1. 4 basic algo.
    2. Binary tree search

Introduction

Introducción

Definición del problema

- La búsqueda es el proceso de localizar un registro (elemento) con un valor de llave particular (**k**).
- La búsqueda termina:
  - Exitosamente cuando se localiza(n) el registro que contenga la llave buscada, o
  - Sin éxito, cuando se determina que no aparece ningún registro con esa llave.

Introduction

Introducción

Lists of searching algorithms (most known)

1. Sequential search
2. Interval search
   i. Binary tree search
   ii. Jump search
   iii. Interpolation search
3. Hash.

Links

- https://www.geeksforgeeks.org/searching-algorithms/

Sequential Search

Búsqueda secuencial

Ordenada

Desordenada

Sequential search

Búsqueda secuencial

- También se le conoce como búsqueda lineal.
- El algoritmo básico de búsqueda secuencial consiste en empezar al inicio de la lista e ir a través de cada registro hasta encontrar la llave indicada (k), o hasta al final de la lista.
- Complejidad
  - La situación óptima es que el registro buscado sea el primero en ser examinado. O(1)
  - El peor caso es cuando las llaves de todos los n registros son comparados con k (lo que se busca). O(n)
  - El caso promedio es n/2 comparaciones. O(n)
- Este método de búsqueda es muy lento, pero si los datos no están en orden es el único método que puede emplearse para hacer las búsquedas.
- Si los valores de la llave no son únicos, para encontrar todos los registros con una llave particular, se requiere buscar en toda la lista.

Search on sorted data

Búsqueda con datos ordenados

Sorted data

Búsqueda en datos ordenados

1. Muestreo de acceso

Este método consiste en observar que tan frecuentemente se solicita cada registro y ordenarlos de acuerdo a las probabilidades de acceso detectadas.

2. Movimiento hacia el frente

- Este esquema consiste en que la lista de registros se reorganicen dinámicamente.
- Con este método, cada vez que la búsqueda de una llave sea exitosa, el registro correspondiente se mueve a la primera posición de la lista y se recorren una posición hacia abajo los que estaban antes que él.

La complejidad es:

- En el mejor de los casos O(1).
- En el peor de los casos es O(n) pero es mejor el tiempo promedio.

Sequential search

Búsqueda secuencial

3. Transposición

- Este es otro esquema de reorganización dinámica.
- Cada vez que se lleve a cabo una búsqueda exitosa, el registro correspondiente se intercambia con el anterior.
- Con este procedimiento, entre más accesos tenga el registro, más rápidamente avanzara hacia la primera posición.
- Comparado con el método de movimiento al frente, el método requiere más tiempo de actividad para reorganizar al conjunto de registros.
- Una ventaja del método de transposición es que no permite que el requerimiento aislado de un registro, cambie de posición todo el conjunto de registros. De hecho, un registro debe ganar poco a poco su derecho a alcanzar el inicio de la lista.

Sequential search

Búsqueda secuencial

4.- Ordenamiento

- Una forma de reducir el número de comparaciones esperadas cuando hay una significativa frecuencia de búsqueda sin éxito es la de ordenar los registros en base al valor de la llave.
- Esta técnica es útil cuando la lista es una lista de excepciones, tales como una lista de decisiones, en cuyo caso la mayoría de las búsquedas no tendrán éxito.
- Con este método una búsqueda sin éxito termina cuando se encuentra el primer valor de la llave mayor que el buscado, en lugar del final de la lista.

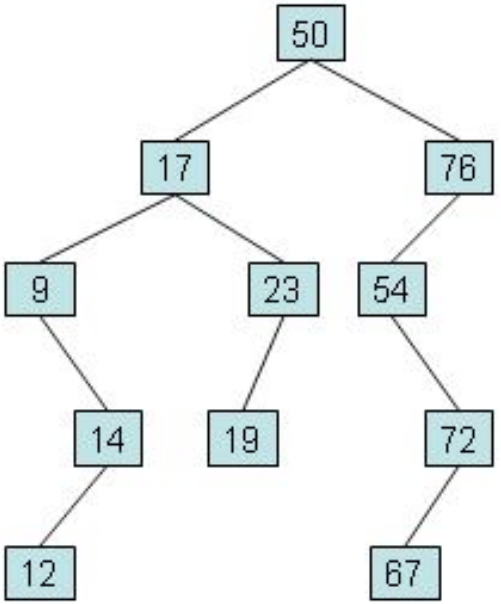
Binary Tree Search

Búsqueda en Árboles binarios

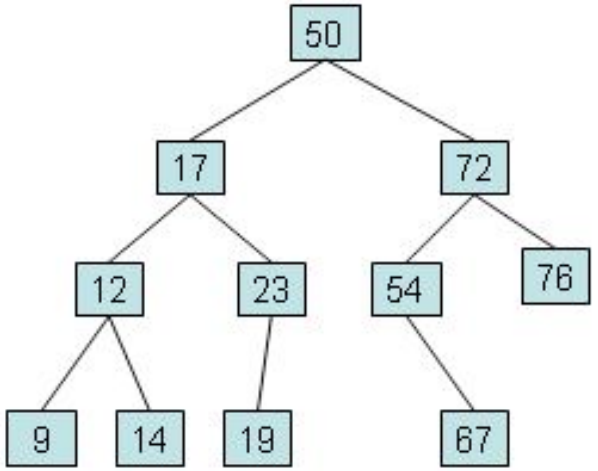
Binary Tree Search

Árboles de Búsqueda

A Binary Search Tree is a node-based data structure where each node contains a key and two subtrees, the left and right. For all nodes, the left subtree's key must be less than the node's key, and the right subtree's key must be greater than the node's key.



An unbalanced tree



The same tree after being height-balanced

Binary Tree Search

Buscar por un dato específico (llave)

Recursive

```
search-recursive(key, node)
        if node is NULL return EMPTY_TREE
        if key < node.key return search-recursive(key, node.left)
        else if key > node.key return search-recursive(key, node.right)
        else return node
```

O(log n)

Iterative

```
searchIterative(key, node)
        currentNode := node
        while currentNode is not NULL
          if currentNode.key = key return currentNode
          else if currentNode.key > key currentNode := currentNode.left
          else currentNode := currentNode.right
```

O(log n)

Binary Tree Search

Buscar el Min o Max

Minimu

```
findMinimum(node)
    if node is NULL
            return EMPTY_TREE min := node
    while min.left is not NULL
            min := min.left
    return min.key
```

O(log n)

Maximum

```
findMaximum(node)
    if node is NULL
            return EMPTY_TREE max := node
    while max.right is not NULL
            max := max.right
    return max.key
```

O(log n)

String Search Algorithms

Búsqueda en cadenas

Rabin-Karp Algorithm

Introducción

Es un Algoritmo de búsqueda de subcadenas simple enunciado por Michael Oser Rabin y Richard Manning Karp en 1987.

- Este algoritmo se basa en tratar cada uno de los grupos de m caracteres del texto (siendo m el número de símbolos del patrón) como un índice de una tabla de valores hash (la llamaremos tabla de dispersión), de manera que si la función hash de los m caracteres del texto coincide con la del patrón es posible que hayamos encontrado un acierto.
- Para verificarlo hay que comparar el texto con el patrón, ya que la función hash elegida puede presentar colisiones.
- La función hash tiene la forma donde es un número primo grande que será el tamaño de la tabla de dispersión y se calcula de la siguiente forma:
  - Para transformar cada subcadena de caracteres en un entero lo que se hace es representar los caracteres en una base que en el planteamiento original coincide con el tamaño del alfabeto. Por tanto el entero correspondiente a la subcadena de texto sería:
  - $x_i = C_i \times B^{m-1} + C_{i-1} \times B^{m-2} + \ldots + C_{i+m-1}$

Fuente. Wikipedia

Conclusions

Conclusiones

Complexities

Complejidades

Resume of complexities

Algorithm	Worst-case	Averag-case	Best-case
Binary tree search	$O(n^2)$		

Conclusions

Conclusiones

If we compare searching algorithms, we have:

1. Binary tree search

The end

Contacto

Raúl Acosta Bermejo

http:www.cic.ipn.mx
http://www.ciseg.cic.ipn.mx/

racostab@ipn.mx
racosta@cic.ipn.mx

57-29-60-00
Ext. 56652