Universidad Autónoma de Baja California

Facultad de Ciencias Químicas e Ingeniería



**ALGORITMOS Y ESTRUCTURA DE DATOS**

**Análisis de Algoritmos Empírico y Recursión**

**Docente:** M.I Palacios Guerreros Alma Leticia

**Alumno:** Gómez Cárdenas Emmanuel Alberto

**Matricula:** 1261509

Contents

[INTRODUCCION 3](#_Toc53784184)

[DESARROLLO 3](#_Toc53784185)

[TABLA COMPARATIVA 4](#_Toc53784186)

[CONCLUSIONES 4](#_Toc53784187)

[ENLACE A CARPETA CON CODIGO (DRIVE) 4](#_Toc53784188)

[CODIGO 5](#_Toc53784189)

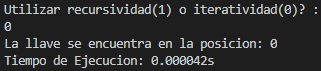
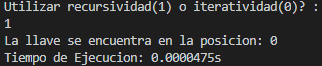
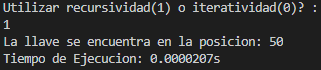
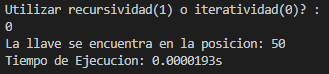
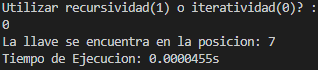
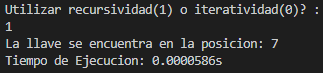
# INTRODUCCION

La búsqueda binaria es un algoritmo que encuentra la posición de un valor en un array ordenado. La búsqueda binaria funciona de la siguiente manera:

Calcular el centro de la lista, con la fórmula (Izquierdo + derecho) / 2. Izquierdo y derecho son las posiciones del elemento menor y mayor del vector. Encontrar el elemento central del arreglo, la llave se compara con el centro si es igual aquí termina la búsqueda. Si no es igual determinar si la llave se encuentra en el lado izquierdo o derecho de la lista. Redefinir el inicio o el final según donde ese haya ubicado la llave. Si la llave es mayor que el centro entonces izquierdo=centro+1. Si la llave es menor que el centro entonces derecho=derecho-1. Repetir desde el primer paso hasta encontrar el dato o hasta que ya no sea posible dividir más. Si la llave no fue encontrada regresar -1.

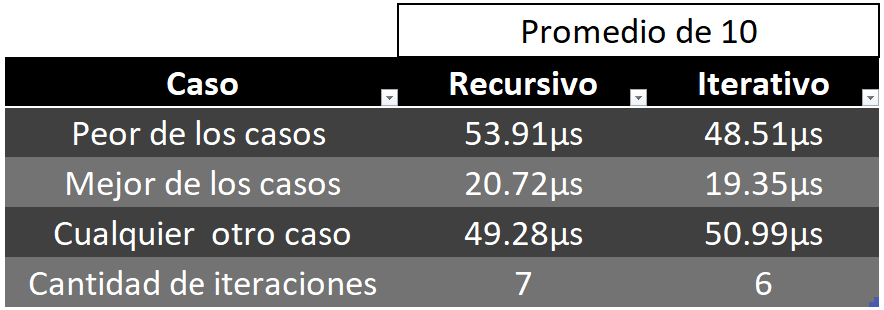
# DESARROLLO

Utilizando el mismo conjunto de datos para ambas implementaciones de la búsqueda. Elabore una tabla con los siguientes casos. (Cadena de 100 palabras)

1. El tiempo para el peor de los casos
2. El tiempo para el mejor de los casos
3. El tiempo para cualquier otro caso
4. Numero de iteraciones:
5. iteraciones para recursiva y 6 para iterativa

* Ejecute los incisos a-c con cadenas de doble tamaño ¿Afecta esto el tiempo de ejecución?
  + Si, aunque los tiempos obtenidos fueron bastante parecidos en ambas pruebas.
* ¿Cuál implementación es más rápida la iterativa o la recursiva?
  + En el mejor y peor caso la iterativa obtuvo mejores resultados, sin embargo, en cualquier otro caso fue más rápida la recursiva.
* ¿Afecta el tamaño de los datos al tiempo de ejecución?
  + Si, se hicieron las pruebas con cadenas de 3 caracteres de longitud y después con las cadenas de más de 20. Con las cadenas de 3, se obtuvo un promedio de 41µs y con las cadenas de mas de 20 aproximadamente 48 µs.
* ¿Afecta si tiene otras aplicaciones abiertas el tiempo de ejecución?
  + Si, se observaron incrementos ligeros en el tiempo de ejecución.

# TABLA COMPARATIVA



# CONCLUSIONES

La recursividad como técnica de programación es una idea bastante interesante, sin embargo, no debe ser tomada a la ligera. Tener una función recursiva nos puede ayudar a resolver problemas que no son tan sencillos si se intentan resolver de forma iterativa, con la principal desventaja de que, de estar mal implementada puede consumir bastantes recursos (memoria principalmente) ya que el hecho de estar llamándose a si misma hace al programa reservar memoria para cumplir dicha función.

# ENLACE A CARPETA CON CODIGO (DRIVE)

https://drive.google.com/file/d/10ZCegfDo2HMkjfOWyeaXRlJDhNhMBnT\_/view?usp=sharing

# CODIGO

import java.text.DecimalFormat;

import java.util.Scanner;

/\*

    Algoritmos y Estructura de Datos

    Práctica 4. Análisis de Algoritmos Empírico y Recursión

    Alumno: Gómez Cárdenas Emmanuel Alberto

    Docente: M.I Palacios Guerreros Alma Leticia

    Fecha de Entrega: 19 de octubre a las 13:00

Implementar el algoritmo de busqueda binaria recursiva e iterativamente

    La búsqueda binaria es un algoritmo que encuentra la posición de un valor

    en un array ordenado. Este compara el valor con el elemento en el medio

    del array, si no son iguales, la mitad en la cual el valor no puede estar

    es eliminada y la búsqueda continúa en la mitad restante hasta que el valor

    se encuentre.

\*/

public class practica4\_GomezEmmanuel{

    public static void main(String args[]){

        binarySearchMain();

    }

    public static void binarySearchMain(){

        Scanner input = new Scanner(System.in);

        DecimalFormat df = new DecimalFormat("#.##########s");

        String[] words = {

        //200words

//Las palabras fueron generadas aleatoriamente

            "abridgeable","acclimatise","albumenised","alembic","allegedly","aneurismal","anguish","antihalation","armillae","attired","autarchically","bahilist","baloney","becloud","bethany","billable","bombay","brahmaputra","breadnut","brewer","brucite","bulky","bullbat","bunco","calvina","cantala","capote","catechise","cephalothoraces","ceraceous","chaldaea","chapter","checksum","chordophone","choriomas","chowchilla","cisel","clingingness","consideration","costarring","costotome","counteractively","counterdeclaration","courtelle","cripplingly","curch","cyaneous","declamation","derbent","descalate","desirableness","desire","dhamma","discomfortable","doughiness","duka","eisteddfod","enrobed","ensphere","estaing","eucalyptic","eupneic","excavator","excretal","experiencing","eyespot","faade","festive","firman","francis","freeloader","gabo","glandule","gluer","groundwards","grummet","guiltier","hamperedly","hierarchical","hunchback","hypercarbia","illegitimation","immigrate","interchanging","interveinous","josiah","knickered","kudos","labdanum","laborless","languorously","lauwine","logroller","lunatical","lusatian","lyse","madag","malayan","mammonite","memphis","metioche","micronesian","midnight","milyukov","mirepois","mlaga","monotrichous","naperian","niersteiner","nondismemberment","nonextrinsic","nontan","obstructionistic","oping","organza","ostentatiousness","overnursed","paleozoic","paracystitis","patinated","peccantly","peeper","pelican","pentahydrate","pepperer","petto","precopying","prediscontented","premeditator","prophylactic","prussianizing","quiescently","racemization","rapidity","rater","reassignment","rebuke","reckon","reenlighten","refundment","remonstrant","rendering","repopularized","roscoe","rounce","sastruga","schizopod","scratchlike","seafighter","semidomestication","serialized","serrate","shrubby","singer","skinneries","somite","sorbic","sosnowiec","stealth","stoppably","strawworm","strongness","subtemperate","supererogating","superficies","suppository","tasteful","tenebrous","theophrastian","thermoelectricity","thurgau","thymi","titicaca","toluene","transitoriness","transnatural","tubifex","unadoring","unallured","unbombastic","uncheering","understandingness","undisparaged","unfilamentous","uniseptate","unmasticatory","unpaginated","unparching","unpardoning","unpicked","unscoring","unsecularised","upsetting","uranian","urticant","vanisher","vugg","wanderoos","wethersfield","zoophilous"

        //100words(size 3)

//      "ard","bag","big","bob","bre","bwg","cab","cad","cay","cbd","des","did","dom","dom","dot","dsc","emu","ene","ens","ezr","fad","fer","fez","fla","foh","gaz","gig","gin","gtc","hal","heh","him","hit","ife","ima","inc","jab","jat","jcd","jeh","jnd","lie","loo","ltd","mad","map","mar","mia","mos","mot","mst","nag","nev","nil","nos","num","oba","odd","oho","one","opp","otc","pav","plu","pos","pru","pte","pun","rab","raf","ray","ray","rco","rhg","sat","sax","sch","sci","scr","sen","shy","son","spt","sue","taj","tee","tef","tor","tow","tug","two","uru","ute","vat","wae","wye","yea","yew","yod","zee"

        //100words(size>20)

//       "aerobacteriologically","anatomicopathological","anticonstitutionalism","anticonstitutionalist","antidisestablishmentarianism","antimaterialistically","antinationalistically","antisupernaturalistic","ballistocardiographic","bioelectrogenetically","chlorotrifluoroethylene","chlorotrifluoromethane","chlorprophenpyridamine","counterrevolutionaries","cyclotrimethylenetrinitramine","demethylchlortetracycline","dendrochronologically","deoxyribonucleoprotein","desoxyribonucleoprotein","diaminopropyltetramethylenediamine","dichlorodifluoromethane","dichlorodifluoromethane","dichlorodiphenyltrichloroethane","dichlorodiphenyltrichloroethane","dicyclopentadienyliron","diphenylaminechlorarsine","disestablishmentarian","disestablishmentarianism","disproportionableness","electrocardiographically","electrodiagnostically","electroencephalograph","electroencephalographic","electroencephalographically","electroencephalography","electromyographically","electrophysiologically","establishmentarianism","hdmezov","hexachlorocyclohexane","hexahydroxycyclohexane","hexamethylenetetramine","humuhumunukunukuapuaa","hydrodesulphurization","hyperconstitutionalism","hyperenthusiastically","hyperpolysyllabically","indistinguishableness","intellectualistically","interdenominationalism","intertransformability","isopropylideneacetone","magnetohydrodynamically","magnetothermoelectricity","methyltrinitrobenzene","microspectrophotometric","microspectrophotometry","misapprehensiveranged","misapprehensiveranging","nitrotrichloromethane","noncharacteristically","nondistinguishableness","noninterchangeability","noninterchangeableness","overcommercialization","overimpressionability","overimpressionableness","overindividualistically","overindividualization","overindustrialization","overintellectualization","overintellectualizing","parathyroidectomizing","pentamethylenediamine","phenylethylmalonylurea","poliencephalomyelitis","pseudoanachronistical","pseudoanthropological","pseudoenthusiastically","pseudohermaphroditism","pseudointernationalistic","pseudophilanthropical","psychophysiologically","psychotherapeutically","representationalistic","spectrophotometrically","succinylsulfathiazole","succinylsulphathiazole","supercalifragilisticexpialidocious","superincomprehensible","superincomprehensibleness","temperameperamentally","thermophosphorescence","transubstantiationalist","triacetyloleandomycin","trichloroacetaldehyde","trichloronitromethane","trifluorochloromethane","trinitrophenylmethylnitramine","ultranationalistically",

    };

        String key = "abridgeable";

        int pos;

        long begin, end;

        System.out.println("\nUtilizar recursividad(1) o iteratividad(0)? : ");

        if(input.nextInt() == 1){

            begin = System.nanoTime();

            pos = recursiveBinarySearch(key, words, 0, words.length);

            end = System.nanoTime();

        }else{

            begin = System.nanoTime();

            pos = iterativeBinarySearch(key, words, 0, words.length);

            end = System.nanoTime();

        }

        double executionTime = (end - begin) / 1000000000.0;

        System.out.println("La llave se encuentra en la posicion: " + pos);

        String seconds = df.format(executionTime);

        System.out.println("Tiempo de Ejecucion: " + seconds + "\n");

        input.close();

    }

    public static int iterativeBinarySearch(String key, String[] words, int left, int right){

        int center = (left+right)/2;

        while (words[center].compareToIgnoreCase(key)!=0){

            if(left == right) return -1;

            else{

                if(words[center].compareToIgnoreCase(key) > 0) right = center -1;

                else left = center + 1;

            }

            center = (left+right)/2;

        }

        return center;

    }

    public static int recursiveBinarySearch(String key, String[] words, int left, int right) {

        int center  = (left+right)/2;

        if(words[center].compareToIgnoreCase(key) == 0){

            return center;  //Caso base

        }

        else if(left == right) return -1;

        else{

            if(words[center].compareToIgnoreCase(key) > 0) right = center -1;

            else left = center + 1;

        }

        return recursiveBinarySearch(key, words, left, right);

    }

}