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

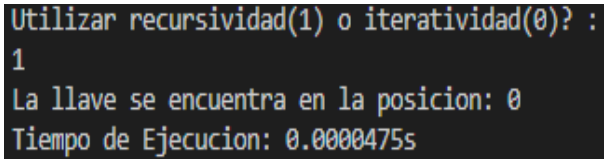
La búsqueda binaria es un algoritmo que encuentra la posición de un valor en un array ordenado. La búsqueda binaria funciona de la siguiente manera:

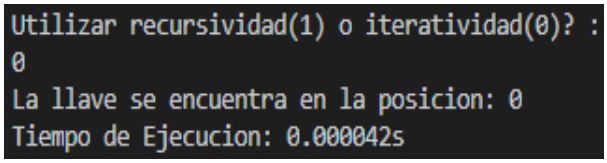
Calcular el centro de la lista, con la fórmula (Izquierdo + derecho) / 2. Izquierdo y derecho son las posiciones del elemento menor y mayor del vector. Encontrar el elemento central del arreglo, la llave se compara con el centro si es igual aquí termina la búsqueda. Si no es igual determinar si la llave 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

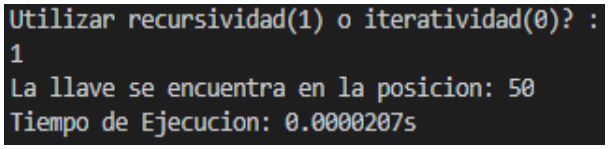
Utilizando el mismo conjunto de datos para ambas implementaciones de la búsqueda. Elabore una tabla con los siguientes casos. (Cadena de 100 palabras)

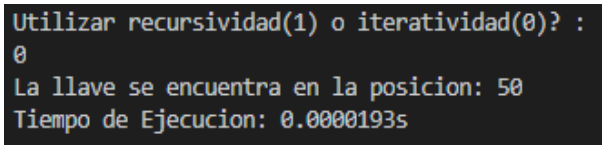
a) El tiempo para el peor de los casos



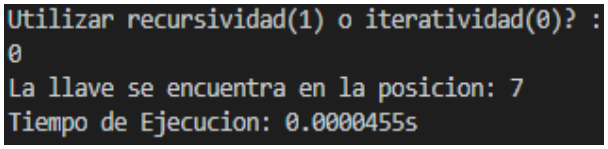


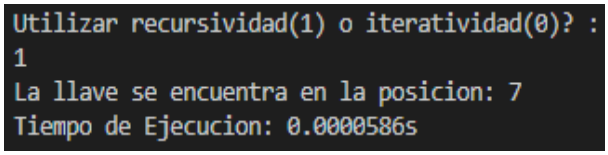
b) El tiempo para el mejor de los casos





c) El tiempo para cualquier otro caso





d) Numero de iteraciones:

7 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

	Promedio de 10	
Caso	**Recursivo**	**Iterativo**
Peor de los casos	53.91µs	48.51µs
Mejor de los casos	20.72µs	19.35µs
Cualquier otro caso	49.28µs	50.99µs
Cantidad de iteraciones	7	6

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","aneu
rismal","anguish","antihalation","armillae","attired","autarchically","bahilist",
"baloney","becloud","bethany","billable","bombay","brahmaputra","breadnut","brewe
r","brucite","bulky","bullbat","bunco","calvina","cantala","capote","catechise","
cephalothoraces","ceraceous","chaldaea","chapter","checksum","chordophone","chori
omas","chowchilla","cisel","clingingness","consideration","costarring","costotome
","counteractively","counterdeclaration","courtelle","cripplingly","curch","cyane
ous","declamation","derbent","descalate","desirableness","desire","dhamma","disco
mfortable","doughiness","duka","eisteddfod","enrobed","ensphere","estaing","eucal
yptic","eupneic","excavator","excretal","experiencing","eyespot","faade","festive
","firman","francis","freeloader","gabo","glandule","gluer","groundwards","grumme
t","guiltier","hamperedly","hierarchical","hunchback","hypercarbia","illegitimati
on","immigrate","interchanging","interveinous","josiah","knickered","kudos","labd
anum","laborless","languorously","lauwine","logroller","lunatical","lusatian","ly
se","madag","malayan","mammonite","memphis","metioche","micronesian","midnight","
milyukov","mirepois","mlaga","monotrichous","naperian","niersteiner","nondismembe
```

```
rment","nonextrinsic","nontan","obstructionistic","oping","organza","ostentatious
ness","overnursed","paleozoic","paracystitis","patinated","peccantly","peeper","p
elican","pentahydrate","pepperer","petto","precopying","prediscontented","premedi
tator","prophylactic","prussianizing","quiescently","racemization","rapidity","ra
ter","reassignment","rebuke","reckon","reenlighten","refundment","remonstrant","r
endering","repopularized","roscoe","rounce","sastruga","schizopod","scratchlike",
"seafighter","semidomestication","serialized","serrate","shrubby","singer","skinn
eries","somite","sorbic","sosnowiec","stealth","stoppably","strawworm","strongnes
s","subtemperate","supererogating","superficies","suppository","tasteful","tenebr
ous","theophrastian","thermoelectricity","thurgau","thymi","titicaca","toluene","
transitoriness","transnatural","tubifex","unadoring","unallured","unbombastic","u
ncheering","understandingness","undisparaged","unfilamentous","uniseptate","unmas
ticatory","unpaginated","unparching","unpardoning","unpicked","unscoring","unsecu
larised","upsetting","uranian","urticant","vanisher","vugg","wanderoos","wethersf
ield","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","a
ntinationalistically","antisupernaturalistic","ballistocardiographic","bioelectro
genetically","chlorotrifluoroethylene","chlorotrifluoromethane","chlorprophenpyri
damine","counterrevolutionaries","cyclotrimethylenetrinitramine","demethylchlorte
tracycline","dendrochronologically","deoxyribonucleoprotein","desoxyribonucleopro
tein","diaminopropyltetramethylenediamine","dichlorodifluoromethane","dichlorodif
luoromethane","dichlorodiphenyltrichloroethane","dichlorodiphenyltrichloroethane"
,"dicyclopentadienyliron","diphenylaminechlorarsine","disestablishmentarian","dis
establishmentarianism","disproportionableness","electrocardiographically","electr
odiagnostically","electroencephalograph","electroencephalographic","electroenceph
alographically","electroencephalography","electromyographically","electrophysiolo
gically","establishmentarianism","hdmezov","hexachlorocyclohexane","hexahydroxycy
clohexane","hexamethylenetetramine","humuhumunukunukuapuaa","hydrodesulphurizatio
n","hyperconstitutionalism","hyperenthusiastically","hyperpolysyllabically","indi
stinguishableness","intellectualistically","interdenominationalism","intertransfo
rmability","isopropylideneacetone","magnetohydrodynamically","magnetothermoelectr
icity","methyltrinitrobenzene","microspectrophotometric","microspectrophotometry"
,"misapprehensiveranged","misapprehensiveranging","nitrotrichloromethane","noncha
racteristically","nondistinguishableness","noninterchangeability","noninterchange
```

```
ableness","overcommercialization","overimpressionability","overimpressionableness
","overindividualistically","overindividualization","overindustrialization","over
intellectualization","overintellectualizing","parathyroidectomizing","pentamethyl
enediamine","phenylethylmalonylurea","poliencephalomyelitis","pseudoanachronistic
al","pseudoanthropological","pseudoenthusiastically","pseudohermaphroditism","pse
udointernationalistic","pseudophilanthropical","psychophysiologically","psychothe
rapeutically","representationalistic","spectrophotometrically","succinylsulfathia
zole","succinylsulphathiazole","supercalifragilisticexpialidocious","superincompr
ehensible","superincomprehensibleness","temperameperamentally","thermophosphoresc
ence","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);
    }
}
```