Departamento de Ciencias de la Computación
UNIVERSIDAD DE CHILE

Auxiliar 10:
Busqueda y Ordenamiento

Algoritmos de Busqueda

Los algoritmos de busqueda son muy importantes para la computación, en la mayoría de las aplicaciones los utilizaremos, basta recordar la Tarea 2 de este curso.

Secuencial: Busca secuencialmente hasta encontrar el elemento. O(n).

Binaria: Busca partiendo del medio, “eligiendo” si el elemento buscado es más alto o más bajo. Requiere $O(\log_2(n))$

Hashing: Busca mediante una función que entrega el lugar donde está un cierto elemento dado ese mismo elemento. O(1)

Es claro cuales son las más rápidos y los más complejos. En orden de más complejo y más rapido a menos complejo y menos rapido son Hashing, Binaria, Secuencial.

En nuestro caso veremos los algoritmos de busqueda para arreglos de todo tipo.

Secuencial

Departamento de Ciencias de la Computación
UNIVERSIDAD DE CHILE

El algoritmo de busqueda secuencial busca en un arreglo siguiendo el indice, es decir busca secuencialmente. Este algoritmo se puede mejorar si el arreglo esta ordenado, pues podremos saber en algún momento si nos pasamos de donde deberia estar el objeto buscado.
Veremos la implementación en un arreglo de Strings.

```
static public int secuencial(String palabra,String arreglo[]){
    for(int i=0;i<arreglo.length;++i){
            int c=arreglo[i].compareTo(palabra);
            if(c>0)return -1;
            if(c==0)return i;
    }
    return -1;
}
```

Escribe los valores que tomaría c para la siguiente llamada.
String a[]={kiwi,manzana,melon,naranja,platano};
String b=”naranja”;
int i=secuencial(b,a);

Binaria

El algoritmo de busqueda binaria parte eligiendo el elemento del centro del arreglo, luego elige el elemento del centro de la primera o la segunda mitad según el elemento elegido anteriormente es mayor o menor (respectivamente) que el buscado. Es claro que se requiere un arreglo ordenado.

```
static public int binario(String palabra,String arreglo[], int n){
    int ip=0,iu=n;
    while(ip>=if){
           int i=(iu+ip)/2;
           int c=arreglo[i].compareTo(palabra);
           if(c==0)return i;
           if(c<0)ip=i+1;
           else iu=i-1;
    }
    return -1;
}
```

Escribe los valores que tomaría c para la siguiente llamada.
String a[]={kiwi,manzana,melon,naranja,platano};
String b=”naranja”;
int i=binaria(b,a,5);

Hashing

El algoritmo de hashing realiza un “hashing” a la palabra a buscar y este devuelve el indice donde debiera estar esta.

```
static public int hashing(String palabra,String arreglo[]){
    int i=palabra.hashCode();
    if(arreglo[i].equals(palabra))return i;
    return -1;
}
```

Digamos que el método hashCode() aplicado a un String entrega la suma de los char casteados a int, es decir para “pera” recibiriamos 112 + 101 + 114 + 97=424.

Qué harias si tienes dos palabras con el mismo hashCode (amor y roma)?

Ejercicio

Departamento de Ciencias de la Computación
UNIVERSIDAD DE CHILE

Escriba una función que realice una busqueda (binaria) en un arreglo de Strings entre dos indices.

```
static public int binaria(String pal,String arr[],int ip,int iu){
   if(ip>iu)return -1;
   int i=(ip+iu)/2;
   int c=pal.compareTo(arr[i]);
   if(c==0)return i;
   if(c<0)return binaria(pal,arr,ip,i-1);
   else return binaria(pal,arr,i+1,iu);
}
```

Propuesto:
Crear un objeto Fecha que tenga las tres busquedas antes explicadas para arreglos de Fecha.

Algoritmos de Ordenamiento

Como vimos anteriormente, los algoritmos de busqueda necesitan algunas veces de elementos ordenados, es por eso que junto a ellos existen diversos algoritmos de ordenamiento que son igual o más importantes que los de busqueda. Estos son:

Exhaustivo : Ordena los elementos a la fuerza, con dos arreglos eligiendo el menor de los no seleccionados.

Selección y Remplazo: Como el exhaustivo pero con un arreglo y eligiendo el menor de los seleccionados para luego reemplazarlo en la posición correcta.

BubbleSort: En cada una de las pasadas “hacer subir la burbuja (el mayor) al último lugar”.

Inserción: Tomamos cada elemento y lo insertamos ordenado en el sector de los ordenados.

Estos algoritmos son de orden cuadrático.

Exhaustivo

El ordenamiento exhaustivo elige el menor de un arreglo y lo transfiere a la primera posición de otro arreglo. Luego marca el que seleccionó y realiza lo mismo en los elementos restantes.

```
static public void exhaustivo(String arreglo[],int n){
    String aux[]=new String[n];
    for(int i=0;i<n;++i){
        int im=indiceMenor(arreglo,n);
        aux[i]=arreglo[im];
        arreglo[im]=null;
    }
    for(int i=0;i<n;++i)
    arreglo[i]=aux[i];
}
```

1) Ordena el siguiente arreglo mediante el método exhaustivo siendo lo más explicito posible.
String a[]={platano,kiwi,sandia,naranja,melon,pera,manzana};

2) Escribe el método indiceMenor(String a[],int n)

Selección y Remplazo

El ordenamiento selección y remplazo es muy parecido al exhaustivo pero ocupa menos memoria pues no necesita de un arreglo auxiliar.

```
static public void remplazo(String arreglo[],int n){
    for(int i=0;i<n;++i){
        int im=indiceMenor(arreglo,i,n);
        String aux=arreglo[im];
        arreglo[im]=arreglo[i];
        arreglo[i]=aux;
    }
}
```

Si bien la implementación es más corta, no aporta mucho a la rapidez del ordenamiento.

1) Ordena el siguiente arreglo mediante el método de selección y remplazo siendo lo más explicito posible.
String a[]={platano,kiwi,sandia,naranja,melon,pera,manzana};

2) Escribe el método indiceMenor(String a[],int ip,int iu)

3) Escribe el método indiceMayor(String a[],int n) explicado en auxiliar

Bubble Sort

Departamento de Ciencias de la Computación
UNIVERSIDAD DE CHILE

El ordenamiento de la burbuja tiene ese nombre por la analogía entre el mayor de un arreglo subiendo por este y una burbuja subiendo por un fluido. Para que el mayor suba haremos comparaciones sucesivas entre pares consecutivos del arreglo.

```
static public void bubbleSort(String arreglo[],int n){
    if(n<2)return;
    for(int i=0;i<n-1;++i)
    if(arreglo[i].compareTo(arreglo[i+1])>0){
        String aux=arreglo[i];
        arreglo[i]=arreglo[i+1];
        arreglo[i+1]=aux;
    }
    bubblesort(arreglo,n-1);
}
```

1) Ordena el siguiente arreglo mediante el método de bubblesort siendo lo más explicito posible.
String a[]={platano,kiwi,sandia,naranja,melon,pera,manzana};

2) Cuantas comparaciones realiza en una llamada?, y en n?

Ejercicio en clase

Departamento de Ciencias de la Computación
UNIVERSIDAD DE CHILE

Tenemos un Objeto llamado Color que es Comparable. El método comparable debe entregar un int que es la resta de las normas. Un Color se define mediante tres colores (Red,Green,Blue) con números para cada uno desde el 0 al 255 donde 0 es ausencia y 255 es máxima presencia.

1) Escriba el método public int compareTo(Comparable x)

2) Escriba el método ordenar que implemente el método de la burbuja pero ordenando de m´as claro a más oscuro.

Inserción

El ordenamiento de inserción va tomando cada elemento del arreglo y lo inserta en el espacio de los ya ordenados.

```
public void ordenar(String[] arreglo, int ip, int iu) {
    for (int i=ip+1; i<=iu; i++){
        String aux = arreglo[i];
        int j=i;
        while(j>0 && A[j-1].compareTo(aux)>0){
            arreglo[j]=arreglo[j-1];
            j--;
        }
        arreglo[j] = aux;
    }
}
```

1) Ordena el siguiente arreglo mediante el método de inserción siendo lo más explicito posible.
String a[]={platano,kiwi,sandia,naranja,melon,pera,manzana};

2) Como lo hariamos si quisieramos ordenar de mayor a menor?

Ejercicio con nota

Departamento de Ciencias de la Computación
UNIVERSIDAD DE CHILE

Escribe la clase Alumno que implementa a Comparable que contiene los siguientes métodos:

public Alumno(String nombre,String fechaNac)
//Constructor del objeto Alumno, fechaNac es de la forma AAAAMMDD

public int compareTo(Comparable c)
//Método de Comparable que devuelve -1 si c es mayor que “this”, 1 si “this” es mayor y 0 si son iguales. La relación de orden es primero por la edad y luego por el nombre.

static public void ordenar(Alumno a[],int ip,int iu)
//Método estático que ordena los alumnos de mayor a menor, es decir en a[ip] debe quedar el mayor.

Departamento de Ciencias de la Computación
UNIVERSIDAD DE CHILE

FIN

Profesor: Andrés Muñoz
Auxiliares: Oscar Alvarez
Pedro Valencia

Realizado con **OpenOffice.org Impress**