Práctico 13 - Búsqueda

Programación 1
InCo - Facultad de Ingeniería, Udelar

Para los ejercicios 1 a 5, considerar las siguientes declaraciones de tipos:

```
const
    MAX = . . .;  {entero mayor estricto que 1}
type
      ArregloEnteros = array [1..MAX] of integer;
      ListaEnteros = ^Celda;
      Celda = record
                elem : integer;
                sig : ListaEnteros;
              end;
```

1. Escriba la función `buscarElemento` que determina si el elemento `e` se encuentra en:
   (a) el arreglo `a`

   ```
   function buscarElemento(e:integer; a:arregloEnteros):boolean;
   ```

   (b) el arreglo ordenado de menor a mayor `a`

   ```
   function buscarElemento(e:integer; a:arregloEnteros):boolean;
   ```

   (c) la lista de enteros `l`

   ```
   function buscarElemento(e:integer; l:ListaEnteros):boolean;
   ```

   ¿Cambiaría su solución si la lista estuviese ordenada?

2. Escriba la función `unSoloMayor` que determina si existe un único elemento mayor que el valor `v` en:
   (a) el arreglo `a`

   ```
   function unSoloMayor(v:integer; a:arregloEnteros):boolean;
   ```

   (b) el arreglo ordenado de menor a mayor `a`

   ```
   function unSoloMayor(v:integer; a:arregloEnteros):boolean;
   ```

   (c) la lista de enteros `l`

   ```
   function unSoloMayor(v:integer; l:ListaEnteros):boolean;
   ```

3. Escriba la función `todosMayores` que determina si todos los elementos son mayores que el valor `v` en:
   (a) el arreglo `a`

   ```
   function todosMayores(v:integer; a:arregloEnteros):boolean;
   ```

   (b) el arreglo ordenado de menor a mayor `a`

   ```
   function todosMayores(v:integer; a:arregloEnteros):boolean;
   ```

   (c) la lista de enteros `l`

   ```
   function todosMayores(v:integer; l:ListaEnteros):boolean;
   ```

4. Escriba la función `cuantosMayores` que determina cuántos son mayores que el valor `v` en:
   (a) el arreglo `a`

   ```
   function cuantosMayores(v:integer; a:arregloEnteros):integer;
   ```

(b) el arreglo ordenado de menor a mayor `a`

```
function cuantosMayores(v:integer; a:arregloEnteros):integer;
```

(c) la lista de enteros `l`

```
function cuantosMayores(v:integer; l:ListaEnteros):integer;
```

5. Escriba la función `ultimaOcurrencia` que determina la última posición en la cual ocurre el elemento `e`. Si `e` no se encuentra, la función devuelve 0. Implemente para las siguientes estructuras:

(a) el arreglo `a`

```
function ultimaOcurrencia(e:integer; a:arregloEnteros):integer;
```

(b) el arreglo ordenado de menor a mayor `a`

```
function ultimaOcurrencia(e:integer; a:arregloEnteros):integer;
```

(c) la lista de enteros `l`

```
function ultimaOcurrencia(e:integer; l:ListaEnteros):integer;
```

6. Sean las siguientes declaraciones:

```
const
    MAXCOL = . . .; {entero mayor estricto que 0}
    MAXFIL = . . .; {entero mayor estricto que 0}
type
    MatrizEnteros = array [1..MAXFIL, 1..MAXCOL] of integer;
```

Escriba la función `buscarElementoMatriz` que determina si el elemento `e` ocurre en la matriz `m`. Puede invocar las funciones de los ejercicios 1 a 5. Para ello puede suponer que las constantes `MAXCOL` y `MAX` coinciden.

```
function buscarElementoMatriz(e:Integer; m:matrizEnteros):boolean;
```

7. Sean las siguientes declaraciones:

```
const
    MAX = . . .; {entero mayor estricto que 0}
type
      TComparacion = (mayor, menor, igual);
      TTexto = record
                     nombre : array [1 .. MAX] of char;
                     tope     : 0 .. MAX;
                 end;
      ArregloNombres = array [1..MAX] of TTexto;
```

(a) Escriba la función `comparacionNombres` que dados dos nombres `n1` y `n2` devuelva la comparación entre ellos. La comparación es en base al orden lexicográfico.

```
function comparacionNombres(n1, n2:TTexto):TComparacion;
```

(b) Escriba la función `buscarNombre` que devuelve `TRUE` si el nombre `n` ocurre dentro del arreglo `a` y `FALSE` en caso contrario. El arreglo de nombres `a` está ordenado en forma lexicográfica y creciente. ¿Realizará búsqueda lineal o binaria? ¿Por qué?

```
function buscarNombre(n:TTexto; a:arregloNombres):boolean;
```

(c) Para la parte anterior, ¿consideró si había nombres repetidos en el arreglo? Si no lo consideró, ¿qué cambia en su código si los hay? ¿Qué cambiaría si hay nombres repetidos y se le pide encontrar la primera ocurrencia de un nombre?

8. Sean las siguientes declaraciones:

```
const
    MAXE  = . . .; {entero mayor estricto que 0}
    MAXC = . . .;  {entero mayor estricto que 0}
    MAX = . . .; {entero mayor estricto que 0}

type
      TTexto = record
                     caracteres : array [1 .. MAX] of char;
                     tope     : 0 .. MAX;
                  end;

     TEstudiante = record
                     nombre     : TTexto;
                     generacion : integer;
                  end;

     TEstudiantes = record
                              estudiante : array [1 .. MAXE] of TEstudiante;
                              tope : 0 .. MAXE;
                           end;

      TCarrera = record
                        nombre : TTexto;
                        estudiantes : TEstudiantes;
                     end;

     TBedelias = array [1 .. MAXC] of TCarrera;
```

que representan los estudiantes que están inscriptos a una carrera de grado en facultad. Un elemento de tipo `TBedelias` tiene ordenadas las carreras en forma lexicográfica y creciente por su nombre. Luego, por cada carrera, la lista de estudiantes se encuentra ordenada por el año de ingreso en forma creciente.

(a) Escriba la función `buscarEstudiante` que determina si el estudiante de nombre `nombreE` está inscripto a la carrera de nombre `nombreC`. Para comparar los nombres de las carreras y de los estudiantes, puede usar la función definida en el ejercicio 7 (a).

```
function buscarEstudiante(nombreE:TTexto; nombreC:TTexto; bed:TBedelias):boolean;
```

(b) En la parte anterior, ¿qué tipos de búsquedas tuvo que realizar? Explique.

(c) Escriba el procedimiento `buscarEstudiantesGeneracion` que devuelva en `bGen` todos los estudiantes que ingresaron en el año `gen`. Si no hay estudiantes que ingresaron en tal año para una carrera, la lista de estudiantes para dicha carrera debe quedar vacía. ¿Qué tipo de búsqueda debe realizar?.

```
procedure buscarEstudiantesGeneracion(gen:integer; bed:TBedelias; var bGen:TBedelias);
```

9. Sean las siguientes declaraciones:

```
const
    MAX = . . .; {entero mayor estricto que 0}
type
      Rango = 1 .. MAX;
      ArregloEnteros = array [rango] of integer;
```

(a) Escriba la función `estaOrdenado` que determina si el arreglo `a` está ordenado en forma ascendente.

```
function estaOrdenado(a:arregloEnteros):boolean;
```

10. (a) Escriba un programa donde el usuario ingresa números enteros positivos, terminando la secuencia con el número `-1`. Puede asumir que como máximo el usuario ingresará una cantidad `N` de números, siendo `N` una constante declarada al inicio del programa. Luego, el usuario ingresa otra secuencia de números enteros positivos que termina en `-1`. El programa debe mostrar, por cada número de la segunda secuencia, si éste pertenece a la primera secuencia.

Ejemplo:

```
1 9 100 20 88 100 1 -1
6
El número 6 no pertenece a la secuencia.
9
El número 9 pertenece a la secuencia
100 2
El número 100 pertenece a la secuencia
El número 2 no pertence a la secuencia
-1
```

Analice las distintas formas de implementar dicho programa teniendo en cuenta los algoritmos de búsquedas y ordenación vistos en el curso y justifique la elección de los algoritmos empleados, de forma de tener en cuenta el tiempo de ejecución del programa.

(b) Modifique su programa para que imprima la cantidad de comparaciones que fueron necesarias para determinar si un número pertenece a la secuencia. ¿Cómo relaciona la cantidad de comparaciones con los algoritmos de búsqueda vistos en el curso y la forma de implementar su programa?