

Grado en Ingeniería InformáticaDepartamento de Ingeniería Informática



GUIÓN DE PRÁCTICAS 3

CONTROL DE ESTADOS REPETIDOS

 Crea una función dentro de busquedaAlum.c que busque un estado dentro de una lista de elementos (nodos que contienen estados). Si lo encuentra debe devolver el valor 1 y 0 en otro caso:

```
int buscaRepe(tEstado *s, Lista L1)
```

Funciones útiles para la implementación:

• En listaia.c dispones de varias funciones para extraer elementos, en este caso una opción es usar la función ExtraerElem:

```
//Pre: Lista C no est· vacla, i: es un lindice entre 0 y TamMax
//Post:Devuelve el elemento i-Esimo de la lista C
tElemento *ExtraerElem(Lista C,int i);
```

 Además en puzleAlum.c debe estar implementada la función iguales, que comprueba si dos estados son iguales:

```
// Devuelve 1 si dos estados son iguales y 0 en caso contrario.
int iguales(tEstado *s, tEstado *t);
```

- 2. Añade las siguientes instrucciones en la función int búsqueda() para terminar de codificar el control de estados repetidos (cada instrucción en el lugar adecuado):
 - InsertarUltimo(Actual, Cerrados);
 - repetido=buscaRepe(Actual->estado, Cerrados);
 - Lista Cerrados= (Lista) CrearLista(MAXI);
 - if (!repetido) { ... }

VARIANTES DE LA BÚSQUEDA EN PROFUNDIDAD

- Añade las funciones necesarias para añadir en el archivo BusquedaAlum.c las siguientes estrategias de búsqueda:
 - 1. Búsqueda en Profundidad con Control de Estados Repetidos
 - 2. Búsqueda con Profundidad Limitada, donde el límite sea un parámetro de entrada a la función
 - 3. Búsqueda con Profundidad Limitada Iterativa