

Análisis de algoritmos

Escuela Politécnica Superior, UAM, 2015–2016

Conjunto de Prácticas no. 3

Fecha de entrega de la práctica y corrección en el laboratorio

- Grupos del Lunes: 7 de Diciembre.
- Grupos del Martes: 8 de Diciembre.
- Grupos del Miércoles: 9 de Diciembre.
- Grupos del Viernes: 11 de Diciembre.

La documentación impresa se entregará al comienzo de la clase de prácticas.

En esta práctica se trata el desarrollo y el análisis de algoritmos de búsqueda sobre diccionarios. Se determinará experimentalmente el tiempo medio de búsqueda de los elementos en un diccionario que emplea como tipo de datos una tabla.

1. TAD Diccionario

Creación de un TAD diccionario definido en **busqueda.h** por las siguientes estructuras:

```
#define NO_ENCONTRADO -2

#define ORDENADO 1
#define NO_ORDENADO 0

typedef struct diccionario {
    int tamano; /* tamano de la tabla */
    int n_datos; /* numero de datos en la tabla */
    char orden; /* tabla ordenada o desordenada */
    int *tabla; /* tabla de datos
} DICC, *PDICC
```

La estructura **DICC** se utilizará para implementar un diccionario empleando una tabla (ordenada o desordenada) como tipo abstracto de datos. La estructura **TIEMPO** servirá para almacenar los tiempos de ejecución de un algoritmo sobre un conjunto de claves.

Implementad las siguientes rutinas en el fichero **busqueda.c**:

```
PDICC ini_diccionario (int tamano,char orden);
void libera_diccionario(PDICC pdicc);
int inserta_diccionario(PDICC pdicc,int clave);
int insercion_masiva_diccionario (PDICC pdicc,int *claves,int tamano);
int busca_diccionario(PDICC pdicc,int clave,int *ppos,pfunc_busqueda metodo);

int bbin(int *tabla,int P,int U,int clave,int *ppos);
int blin(int *tabla,int P,int U,int clave,int *ppos);
int blin_auto(int *tabla,int P,int U,int clave,int *ppos);
```

donde **pfunc_busqueda** es un puntero a la función de búsqueda, definido como:

```
typedef int (* pfunc_busqueda)(int*, int,int,int,int*);
```

que corresponde con las declaraciones de funciones de búsqueda **bbin**, **blin** y **blin_auto** del fichero **busqueda.h**.

- La función **ini_diccionario** creará un diccionario vacío (sin datos) del tipo indicado por sus argumentos, **tamano** indicará el tamaño inicial que deseamos darle al diccionario y **orden** indicará usando las constantes **ORDENADO** o **NO_ORDENADO** si se emplea una tabla ordenada o desordenada como estructura de datos.
- La función **inserta_diccionario** introduce el elemento clave en la posición correcta del diccionario definido por **pdicc** en función del valor del campo orden del diccionario. Si dicho campo toma el valor **NO_ORDENADO** la rutina insertará el elemento al final de la tabla. Si el campo toma el valor **ORDENADO**, la rutina insertará el elemento en la posición correcta que mantenga la tabla ordenada mediante una sola iteración del método de inserción, es decir, colocará el elemento al final de la tabla y lo colocará en el sitio correcto utilizando el siguiente trozo de código

```
A=T[U];
j=U-1;
mientras (j >= P && T[j]>A);
    T[j+1]=T[j];
    j--;
T[j+1]=A;
```

- La función **insercion_masiva_diccionario** introduce todas las claves del parámetro **claves** en el diccionario.
- La función **busca_diccionario** buscará una clave en el diccionario definido por **pdicc** usando la rutina indicada por **metodo**, la función devolverá la posición de la clave en la tabla por medio de la posición apuntada por **ppos**.
- La función **libera_diccionario** liberará toda la memoria reservada por las rutinas del TAD diccionario.
- Las rutinas **bbin**, **blin** y **blin_auto** implementarán los algoritmos de búsqueda binaria, búsqueda lineal y búsqueda lineal auto-organizada (cuando un elemento es encontrado en la tabla, se intercambia con el inmediatamente anterior, excepto si se trata del primer elemento de la tabla) respectivamente, las funciones devolverán la posición de la clave en la tabla o la constante **NO_ENCONTRADO** en el caso de que la clave no se encuentre en la tabla por medio de la posición apuntada por **ppos**.

Todas las funciones devolverán el número de Operaciones Básicas que han empleado o **ERR**, en función de que su desarrollo fuera el deseado o no, exceptuando **ini_diccionario** que devolverá **NULL** si ha ocurrido un error o bien el diccionario construido si todo se ha realizado correctamente.

El programa C de prueba para estas rutinas es **ejercicio1.c** que está incluido en **codigo_p3.zip**.

Modificar el programa **ejercicio1** para que permita comprobar el funcionamiento tanto de la búsqueda binaria como de búsqueda lineal sobre una tabla ordenada.

2. Comparación del rendimiento de búsqueda

En este apartado se deberán implementar una serie de funciones para realizar medidas de rendimiento de las funciones de búsqueda implementadas. Para ello se debe crear un nuevo módulo **tiempos.c** y **tiempos.h** donde poner todas estas rutinas así como la estructura **TIEMPO** y todas las rutinas de la práctica anterior (el fichero **tiempos.h** está incluido en **codigo_p3.zip**).

Para ello se debe implementar la función:

```
short tiempo_medio_busqueda(pfunc_busqueda metodo, pfunc_generador_claves generador,
    char orden,
    int tamano,
    double fclaves,
    int n_ciclos,
    PTIEMPO ptiempo);
```

donde:

- **orden** indica si se usarán tablas ordenadas en el TAD diccionario.

- **tamano** es el número de claves que debe contener el diccionario donde se realizan las búsquedas.
- **fclaves** es un factor que indica el número de claves que se buscarán en el diccionario. Se deben buscar $fclaves * \text{tamano_claves}$. Por ejemplo si **fclaves** es igual a 2 entonces se buscarán **tamano*2** claves.
- **n_ciclos** representa el número de veces que se repite la medición de tiempos (equivalente **n_perms** de la práctica 2). **ptiempo** es un puntero a una estructura de tipo **TIEMPO** que a la salida de la función contendrá el número de ciclos realizados en el campo **n_ciclos** (antes **n_perms**), el tamaño del diccionario en el campo **tamano**, el tiempo medio de ejecución (en segundos) en el campo **tiempo**, el número promedio de veces que se ejecutó la OB en el campo **medio_ob**, el número mínimo de veces que se ejecutó la OB en el campo **min_ob** y el número máximo de veces que se ejecutó la OB en el campo **max_ob**.

Además la rutina recibe un puntero a la función de búsqueda (parámetro **metodo**) y otro puntero a la función que se debe usar para generar las claves que deben ser buscadas (parámetro **generador**) y que ya se encuentra implementada en los ficheros **busqueda.c** y **busqueda.h** suministrados. La rutina **tiempo_medio_busqueda** devuelve **ERR** en caso de error y **OK** en el caso de que las búsquedas se realicen correctamente.

Además se debe implementar la función:

```
short genera_tiempos_busqueda(pfunc_busqueda metodo, pfunc_generador_claves generador,
                             int orden, char* fichero,
                             int num_min, int num_max,
                             int incr, double fclaves, int n_ciclos);
```

que sirve para automatizar la toma de tiempos. Esta función llama a la función anterior con tamaños de diccionario desde **num_min** hasta **num_max**, ambos incluidos, usando incrementos de tamaño **incr**. La rutina devolverá el valor **ERR** en caso de error y **OK** en caso contrario. Además debe guardar los resultados en el fichero **fichero** mediante la función ya implementada **short guarda_tabla_tiempos(char *fichero, PTIEMPO tiempo, int N)**. El resto de parámetros son equivalentes a la rutina anterior.

Con estas rutinas de medición de tiempos se deben realizar los siguientes pruebas:

1. Comparar tiempo medio y operaciones básicas medias de las búsquedas lineal y binaria. El método de búsqueda lineal se aplicará sobre diccionarios desordenados y la búsqueda binaria en diccionarios ordenados. Se deben buscar todas las claves contenidas en el diccionario una sola vez. Es decir $fclaves=1$ y la función de generación de claves debe ser generador claves uniforme incluida en **busqueda.h** y **busqueda.c**. Se debe hacer la comparación al menos para tamaños de diccionarios desde 1000 a 10000 elementos.
2. Sin embargo, no siempre las claves a buscar siguen una distribución uniforme sino que habrá unas claves que se buscarán con más frecuencia que otras. Bajo estas condiciones se puede mejorar el tiempo medio de extracción si hacemos que las claves más frecuentes se puedan obtener en tiempo cercano a $O(1)$. Un modo de realizar esto es mediante búsqueda lineal auto-organizada que utiliza las propias claves a buscar para ir colocando las claves más comunes al inicio de la lista. Para que la búsqueda auto-organizada tenga efecto se deberán buscar muchas claves para conseguir que la colocación de las claves en la lista alcance su orden óptimo.

Las listas autoorganizadas son particularmente útiles cuando la consulta a los datos siguen reglas como la de 80/20: el 80 % de las consultas preguntan por el 20 % de las claves, o cuando la frecuencia de los datos siguen distribuciones potenciales como la de Zipf, como ocurre con las palabras de un texto: hay pocas palabras con mucha frecuencia y muchas palabras con poca frecuencia, esto también parece suceder con los enlaces de una página web.

Para realizar las pruebas, se deberá trabajar con la función generadora de claves **generador_claves_potencial**, que genera claves con una distribución no uniforme. Con este generador de claves se deberá comparar en tiempo medio y número de operaciones básicas medias para la búsqueda auto-organizada (**blin auto**) en diccionarios desordenados y la búsqueda binaria en diccionarios ordenados. Se deberá probar para diccionarios de 100 elementos cambiando el valor del parámetro **fclaves** (que indica cuantas búsquedas se realizan) al menos para 1, 10, 100, 1000 y 10000.

El programa C de prueba para estas rutinas es **ejercicio2.c** incluido en **codigo_p3.zip**.

Cuestiones teóricas

1. ¿Cuál es la operación básica de **bbin**, **blin** y **blin_auto**?

2. Dar tiempos de ejecución en función del tamaño de entrada n para el caso peor $W_{SS}(n)$ y el caso mejor $B_{SS}(n)$ de **BBin** y **BLin**. Utilizar la notación asintótica (O, Θ, o, Ω , etc) siempre que se pueda.
3. Cuando se utiliza **blin_auto** y la distribución no uniforme dada ¿Cómo varía la posición de los elementos de la lista de claves según se van realizando más búsquedas?
4. ¿Cuál es el orden de ejecución medio de **blin_auto** en función del tamaño de elementos en el diccionario n para el caso de claves con distribución no uniforme dado? Considerar que ya se ha realizado un elevado número de búsquedas y que la lista está en situación más o menos estable.
5. Justifica lo más formalmente que puedas la corrección (o dicho de otra manera, el por qué busca bien) del algoritmo **bbin**.

Material a entregar en cada uno de los apartados

Documentación: La documentación constará de los siguientes apartados:

1. **Introducción:** Consiste en una descripción de tipo técnico del trabajo que se va a realizar, qué objetivos se pretenden alcanzar, qué datos de entrada requiere vuestro programa y qué datos se obtienen de salida, así como cualquier tipo de comentario sobre la práctica.
2. **Código impreso:** El código de la rutina según el apartado. Como código también va incluida la cabecera de la rutina.
3. **Resultados:** Descripción de los resultados obtenidos, gráficas comparativas de los resultados obtenidos con los teóricos y comentarios sobre los mismos.
4. **Cuestiones:** En el último apartado, incluir las respuestas en papel a las cuestiones teóricas anteriores.

Esta documentación se le entregará al profesor de prácticas, en el correspondiente día de entrega de la práctica. En portada deberá incluirse los nombres de los estudiantes.

Todos los ficheros necesarios para compilar la práctica y la documentación se guardarán en un único fichero comprimido, en formato zip, o tgz (tgz representa un fichero tar comprimido con gzip).

Adicionalmente, las prácticas deberán ser guardadas en algún medio de almacenamiento (lápiz usb, CD o DVD, disco duro, disco virtual remoto, etc) por el alumno para el día del examen de prácticas en Diciembre.

Ojo: Se recalca la importancia de llevar un lápiz usb **ademas de otros medios de almacenamiento como discos usb, cd, disco virtual remoto, email a direccion propia, etc**, ya que no se garantiza que puedan montarse y accederse todos y cada uno de ellos durante el examen, lo cual supondría la calificación de suspenso en prácticas.

Asimismo, una copia de los fuentes y la documentación deberán enviarse por el sistema electrónico de entrega de prácticas, como se indica a continuación.

Instrucciones para la entrega de los códigos de prácticas

La entrega de los códigos fuentes correspondientes a las prácticas de la asignatura AA se realizará por medio de la página web **www.uam-virtual.es**.