VNIVERSITAT DE VALÈNCIA

Laboratorio de PROGRAMACIÓN
Grado en Ingeniería Multimedia (1º)
Curso 2021-22

Práctica Nº 1: Algoritmos de Búsqueda
Periodo de realización: Semana del **21 a 25/02/2022**

Material a entregar

- **(Repositorio) Código fuente necesario para la práctica**.
- **(Tarea AV) Documentación del código**.

Introducción

La búsqueda es un proceso de análisis de la información ampliamente utilizado, en especial cuando se dispone de un volumen elevado de datos sobre un determinado tema (censo de habitantes de una ciudad, datos sanitarios de la población, actividades de una red social, etc.). Es importante que el algoritmo empleado sea rápido, ya que es una tarea que se puede ejecutar muy frecuentemente (miles/millones de veces) y sobre colecciones de datos muy grandes (miles/millones de registros).

En esta práctica vamos a abordar el cálculo empírico/experimental de costes de diversos algoritmos de búsqueda. En concreto, se pretende obtener resultados experimentales del coste de la implementación de los algoritmos típicos de búsqueda sobre vectores: (1) Secuencial (sin parada), (2) Secuencial con parada, (3) Secuencial con centinela y (4) Búsqueda binaria.

A modo de ejemplo, trabajaremos con un vector con información de personas a las que se les ha administrado la vacuna contra Covid-19[1], que puede representar, por ejemplo, el registro de vacunación de una comunidad autónoma. Sobre este conjunto de datos se realizará un proceso "intensivo" de búsquedas para determinar el estado de vacunación de (miles) pacientes de un hospital[2]. En este problema se aplicarán los 4 algoritmos que se desean analizar y se medirá el número los pasos ejecutados para realizar este proceso y el coste medio de cada búsqueda de cada algoritmo. Para medir los pasos solo se tendrán en cuenta las operaciones "críticas", que son las que tienen lugar dentro de los bucles de búsqueda de cada uno de los algoritmos.

Para realizar los ejercicios se tomará como material inicial el programa "**`pr1_v0_9.cpp`**" que se puede descargar desde el Aula Virtual de la asignatura. Este programa incluye la definición del tipo de datos a utilizar (*`TPersona`*), la versión preliminar de los algoritmos de búsqueda que se desean comparar y el mecanismo de lectura de datos desde un archivo externo. A partir de aquí se debe:

1. Leer el código proporcionado e interpretar su significado y funcionamiento.
2. Realizar las modificaciones indicadas en los siguientes ejercicios hasta llegar al cálculo práctico de costes de algoritmos.

[1] Estos datos son evidentemente falsos, pues los datos personales de salud están catalogados con nivel de seguridad alto y no son públicos.
[2] Este proceso es habitual, por ejemplo, al realizar ingresos de nuevos pacientes en un hospital.

Ejercicios

NOTA: Los comentarios etiquetados como SCV indican las tareas a realizar para llevar el seguimiento de revisiones del código generado.

Tarea 1: Empezara a trabajar con el respositorio del SCV

Antes de iniciar las tareas de programación debes preparar tu ordenador para trabajar con el sistema de control de versiones (SVC). Lee atentamente el documento "SVC_GuiaEstudiante" disponible en Aula Virtual. Sigue las instrucciones indicadas en ese documento para Instalar en tu ordenador la aplicación GIT-GUI y clonar el repositorio que te corresponde para realizar las prácticas. Una vez realizado el proceso anterior, si todo ha funcionado correctamente, deberás tener en la carpeta de trabajo de tu ordenador una estructura con 8 subcarpetas, una por práctica, donde deberás almacenar los archivos que generes en cada una de las prácticas de la asignatura.

Descarga ahora de Aula Virtual el programa "**pr1_v0_9.cpp"** y los archivos de datos de la práctica. Copia todos estos archivos en la carpeta "Pr1" del repositorio creado en tu ordenador. Para hacer esto no es necesario utilizar el programa GIT-GUI. La gestión de carpetas y archivos debes hacerla normalmente utilizando el explorador de archivos de tu sistema operativo (Windows, Linux, Mac).

Si abres el programa GIT-GUI y el repositorio de la asignatura en tu ordenador verás que hay cambios: los nuevos archivos copiados. Sigue las instrucciones (documento "SVC_GuiaEstudiante") para hacer un *commit* de esta revisión del proyecto, con el mensaje "Versión inicial" y súbelo al servidor con *push*. Ten en cuenta que debes realizar un único *commit*, que incluya a todos los archivos copiados, y una única operación *push*. Para ello, todos los archivos que aparecen en el apartado "Unstaged Changes" deben pasar a "Staged Changes", añadir el mensaje indicado, después realizar *commit* y, a continuación, *push*.

Como el repositorio es accesible por los dos miembros de la pareja de prácticas, es suficiente con que uno de los miembros haya hecho esta tarea para que el otro pueda ver los cambios al clonar el repositorio en su ordenador.

Tarea 2

Renombra ahora el programa **"pr1_v0_9.cpp"** como "**pr1.cpp**". En su estado original el programa permite:

1. Leer y cargar en un vector la información de personas vacunadas disponible en el archivo **"vacunaciones_small.dat"**.
2. Mostrar por pantalla los datos cargados en el vector y verificar que corresponden con los disponibles en el archivo.

Sin embargo, para que el programa compile sin errores será necesario modificar (con DevC++) las funciones de búsqueda incluidas en el programa, ya que en su estado actual dan lugar a errores de compilación motivados porque comparan elementos completos del vector (todo el registro) con un identificador de personas, que es la clave que se desea utilizar para buscar. Comparaciones del estilo `v[i]==id` comparan variables de diferente tipo (y generan los errores), puesto que `v[i]` es un registro (*TPersona*) e `id` es un identificador (*string*). Son este tipo de comparaciones las que deben ser adaptadas al problema concreto que se debe resolver (buscar personas por identificador). Además, debes prestar especial atención a la definición del centinela empleado en la función

BusquedaSecuencialCentinela, que deberá ser una *TPersona* con el **`identificador`** buscado (el resto de los campos no importan).

Compila y ejecuta el programa tras las modificaciones y verifica que los datos mostrados corresponden con los disponibles en el archivo.

SCV: Utilizando GIT-GUI, abre el repositorio (si no lo tuvieras abierto) y refresca la venta de cambios con F5. En el apartado "Unstaged Changes" debe aparecer el fichero "`pr1.cpp`". Sigue las instrucciones del documento "SVC_GuiaEstudiante" para hacer *commit* (solo una vez) de estos cambios con el mensaje "`Completada Tarea 2`". Si vas a continuar trabajando no hace falta que hagas *push*.

Tarea 3

Ahora debes añadir al programa el proceso de búsqueda en el vector de las personas con los siguientes identificadores: `0002JTX`, `1111VAC`, `0006HYR`, `1543GYM` y `0014FGV`. Para cada identificador se deben aplicar los 4 métodos de búsqueda y mostrar sus resultados en pantalla: (1) el identificador buscado, (2) si ha sido encontrado o no y (3) cuántos pasos se han realizado con cada método de búsqueda utilizado[3].

Debes tener en cuenta que ya existe una función *RealizarBusquedas* que ejecuta sucesivamente los 4 algoritmos de búsqueda sobre el mismo identificador. Por lo tanto, habrá que llamarla adecuadamente desde el programa principal. En ese programa (función *main*) se ha declarado, pero comentado, la variable *`contar`*, que permite almacenar los contadores de pasos de los 4 algoritmos (fíjate en el tipo de datos que se ha declarado). Descomenta esa variable y úsala para almacenar los pasos de los algoritmos.

Verifica que los resultados obtenidos para los 5 identificadores de prueba son correctos, de acuerdo con el contenido del archivo.

SCV: Utilizando GIT-GUI, abre el repositorio (si no lo tuvieras abierto) y refresca la venta de cambios con F5. En el apartado "Unstaged Changes" debe aparecer el fichero "`pr1.cpp`". Sigue las instrucciones del documento "SVC_GuiaEstudiante" para hacer *commit* (solo una vez) de estos cambios con el mensaje "`Completada Tarea 3`". Si vas a continuar trabajando no hace falta que hagas *push*.

Tarea 4

Modifica las funciones de búsqueda del programa resultante de la tarea anterior para que devuelvan, en lugar de un valor booleano, la posición en la que se encuentra el elemento. Si el elemento no se encuentra en el vector las funciones devolverán el valor -1. Especial atención requiere la función *BusquedaSecuencial*.

De la misma manera, debes modificar el valor de retorno de la función *RealizarBusquedas* para que sea coherente con el cambio realizado en las otras funciones y también devuelva la posición y no el valor bool actual.

[3] Fíjate que las funciones proporcionadas ya calculan y devuelven el número de pasos ejecutados dentro de los bucles de búsqueda de cada uno de los algoritmos, tal como se ha indicado en la introducción.

Verifica los resultados de las 4 funciones con el siguiente banco de pruebas:

		Pasos			
Clave	**Posición**	**B.Secuencial**	**B.Sec.Parada**	**B.Centinela**	**B.Binaria**
0002JTX	2	31	6	4	10
1111VAC	-1	30	30	20	20
0006HYR	5	31	15	10	10
1543GYM	-1	30	30	20	20
0014FGV	9	31	27	18	15

Las diferencias entre algoritmos no son, en absoluto, significativas. Fundamentalmente porque se trata de una prueba con un número muy pequeño de elementos en el vector (solo 10).

SCV: Utilizando GIT-GUI, abre el repositorio (si no lo tuvieras abierto) y refresca la venta de cambios con F5. En el apartado "Unstaged Changes" debe aparecer el fichero "`pr1.cpp`". Sigue las instrucciones del documento "SVC_GuiaEstudiante" para hacer *commit* (solo una vez) de estos cambios con el mensaje "`Completada Tarea 4`". Si vas a continuar trabajando no hace falta que hagas *push*.

Tarea 5

Modifica el programa para que permita obtener resultados del promedio de pasos realizados por cada algoritmo de búsqueda al buscar en un vector de mayor tamaño y con un mayor número de búsquedas. Para ello, el programa debe:

1. Leer del archivo "**`vacunaciones_big.dat`**" (que tiene más de 5.000 personas). Este archivo representa (hipotéticamente) al colectivo de personas vacunadas según el registro de la Conselleria de Sanitat de la Comunitat Valenciana.
2. Buscar (con los 4 métodos) todos los identificadores de personas contenidos en el archivo "**`verificar.dat`**", que, por ejemplo, representan a los pacientes que un hospital ha citado para atender durante la próxima semana y cuyo estado de vacunación debe conocer.
3. Proporcionar un resumen de datos sobre los resultados del proceso. El formato de salida y el banco de pruebas que debe permitir validar tu programa es el siguiente (la talla es el número de elementos almacenados en el vector de búsqueda):

(página siguiente)

```
-------------------------
RESULTADOS Talla = 5083
-------------------------
Pacientes buscados = 1442
        -Vacunados: 1302
                -Solo 1 dosis: 206
                -Solo 2 dosis: 628
                -3 dosis: 468
        -No Vacunados: 140
Media de pasos realizados por los 4 algoritmos:
        - Busqueda Secuencial: 15249.9
        - Busqueda Secuencial con Parada: 8282.71
        - Busqueda Secuencial con Centinela: 5521.81
        - Busqueda Binaria: 52.8779
-------------------------------
```

Fíjate que cada vez que se realiza una búsqueda debes analizar si el paciente ha sido encontrado (vacunado) o no, si ha sido encontrado debes identificar cuántas dosis ha recibido para obtener los indicadores totales. De la misma manera, cada búsqueda requiere contabilizar los pasos realizados por cada algoritmo para poder calcular el promedio final, que dependerá del número de búsquedas realizadas.

Para esta tarea se deben suprimir todos los mensajes que muestran las funciones cuando se busca una persona o cuando se encuentra. Es una información de depuración que ya no es relevante y que confunde al usuario del programa.

SCV: Utilizando GIT-GUI, abre el repositorio (si no lo tuvieras abierto) y refresca la venta de cambios con F5. En el apartado "Unstaged Changes" debe aparecer el fichero "`pr1.cpp`". Sigue las instrucciones del documento "SVC_GuiaEstudiante" para hacer *commit* (solo una vez) de estos cambios con el mensaje "`Completada Tarea 5. Versión final`". Como has finalizado el trabajo, debes hacer *push* para subir los cambios al repositorio.

Tarea 6: Generación de documentación

El código realizado deberá estar correctamente documentado con Doxygen, de acuerdo con la guía de estilo. El contenido de los comentarios debe ser coherente con las operaciones realizadas por el programa al finalizar el proceso y, por lo tanto, se deben ir actualizando al realizar las modificaciones indicadas durante el proceso de desarrollo de la práctica.

Una vez realizada la versión final del programa, se deberá ejecutar Doxygen para generar toda la documentación definitiva. La documentación deberá ser generada exclusivamente en formato en *html* (utiliza como referencia, convenientemente personalizado para esta práctica, el archivo de configuración de Doxygen disponible en el Aula Virtual). Solo la documentación deberá subirse a Aula Virtual (como archivo comprimido) una vez concluido el trabajo.

El programa NO se debe subir a AV, ya que será evaluado directamente desde el repositorio de control de versiones.

La documentación generada con Doxygen NO debe subirse al repositorio de control de versiones. Se debe mantener exclusivamente en el ordenador personal.