Compilación y ejecución

Àlex Osés Laza, Albert Segarra Roca

Algorismia

FIB • UPC

11 de enero de 2016

1. Organización de ficheros

El proyecto se organiza en 6 carpetas principales:

- **src/algoritmos**: Contiene los ficheros fuente de los algoritmos implementados. Podrás ver que estos ficheros fuente contienen unas directivas de compilador del tipo `#if _STATS_ ...`. Las hemos usado para poder generar dos ejecutables distintos: uno para medir el tiempo de ejecución, y otro para medir las estadísticas. Lo hemos hecho de esta forma para que la toma de estadísticas no influyera en el tiempo de ejecución. De este modo, hay trozos de código (los que calculan estadísticas) que sólo se compilan si así se indica en el comando de compilación. Dado que estas directivas dificultan la lectura del código hemos incluido una segunda carpeta que contiene las versiones de los algoritmos sin las directivas/cálculos de estadísticas, esta se llama src/algoritmos-sin-stats
- **src/algoritmos-sin-stats**: Explicada en el punto anterior
- **src/headers**: Esta carpeta contiene dos cabeceras con código que todos los algoritmos comparten, y que no tiene que ver con la implementación del algoritmo en sí, como son la lectura y escritura (fichero `src/headers/io.hpp`) o la medición de tiempo (fichero `src/headers/cronometro.hpp`)
- **src/archivos**: Aquí se almacenan los archivos (`arxiu1 y arxiu2`) que se usan como diccionario y texto respectivamente, y que son generados por el ejecutable `generador-archivos`.
- **src/salidas**: Aquí se guardan las salidas de los algoritmos cuando estos se ejecutan con el programa `ejecutar`. Esto incluye las estadísticas, el tiempo de ejecución y el resultado del algoritmo.
- **src/datos**: Aquí se guardan los datos recopilados de la ejecución de los programas, que se recaban mediante el programa `generador-datos.py`.

2. Compilación

Para compilar todos los programas necesarios, incluyendo los ficheros fuente de la carpeta `algoritmos`, tanto en su versión de medición de tiempo como en su versión de estadísticas, y el generador de ficheros de entrada usaremos el `Makefile`. Simplemente ejecutando el comando `make` se nos generarán todos los ejecutables. También se pueden compilar los programas individualmente ejecutando `make nombre-programa` donde `nombre-programa` es el nombre del fichero fuente del programa sin la extensión `.cc`. Esto generará los ejecutables en la misma carpeta en la que estén situados los ficheros fuente, y con nombre = `nombre-programa`.

Los ficheros `generador-datos`, `ejecutar` y `ejecutar-rapido` no requieren de compilación, ya que son interpretados, el primero por `Python` y los otros dos por `Bash`.

Podemos dejar el proyecto en el estado inicial mediante el comando `make ultraclean`, que borra todos los ficheros ejecutables generados, todas las salidas generadas, todos los datos generados y todos los archivos generados. Por otro lado, el comando `make clean` sólo eliminará los ejecutables generados.

3. Generación de archivos

Para generar los archivos (diccionario y texto, llamados respectivamente `arxiu1` y `arxiu2`), debemos primero compilar el programa `generador-archivos.cc` y luego ejecutarlo con el siguiente comando:

```
./generador-archivos tamaño-diccionario proporcion relacion-tamaño-texto-diccionario
```

En resumen, el programa recibe 3 argumentos, el primero es el tamaño del diccionario, el segundo es la proporcion de elementos del diccionario que estarán tambien en el texto (de 0 a 1) y el tercero es el tamaño del texto en función del tamaño del diccionario (mínimo 2). Como ya hemos dicho, los dos ficheros resultantes se guardan en la carpeta `archivos`·

4. Ejecución

Para ejecutar los programas lo podemos hacer de 3 formas distintas:

- Con el programa `ejecutar`
- Con el programa `ejecutar-rapido`
- Manualmente

Los dos primeros son esencialmente iguales. La única diferencia es que el primero comprueba que el resultado que da el algoritmo que se ejecuta sea correcto, comparándolo con la salida del programa `algoritmo-seguro.cc`, que es muy sencillo y utiliza el diccionario de la stl, mientras que el segundo no lo hace, y por lo tanto es más rapido.

Para utilizarlos, primero tienen que estar generados los ficheros `arxiu1` y `arxiu2` en la carpeta `archivos`. Entonces solo hace falta pasarle al programa como parametro el nombre del algoritmo a ejecutar, por ejemplo:

`ejecutar tabla-hash-lista`

Y el script ejecuta el programa en versión estadísticas y versión medición de tiempo, y da como salida la información de estadísticas recabada y la información del tiempo de ejecución. La salida del programa se guarda en salidas/nombre-algoritmo/output.out y salidas/nombre-algoritmo/output-stats.out respectivamente para la versión de medición de tiempo y la version de estadísticas. También se guardan en esa carpeta los ficheros en formato `json` con las estadísticas y la información del tiempo de ejecución.

El formato de la salida del programa es una serie de 0 y 1 que indican si el elemento de la correspondiente posición en el texto `arxiu1` pertenece al diccionario (1) `arxiu2` o no (0).

La lista de algoritmos ejecutables es la siguiente:

- **busqueda-binaria**
- **filtro-bloom**
- **tabla-hash-lista**
- **tabla-hash-hopscotch**
- **trie**

5. Generación de datos

La generación de datos se hace mediante el ejecutable `generador-datos.py`. Para ejecutarlo no es necesario pasarle ningun parametro. Los resultados se guardan en la carpeta datos. Es importante tener en cuenta que la ejecución tarda mucho tiempo, aproximadamente 1h/1h y media, dado que las pruebas son largas. Si consideras que ese tiempo es demasiado alto, puedes probar de ejecutar el fichero oculto `.generador-datos-light`, en el que el exponente del tamaño del diccionario se ha reducido de 6 a 5 en todos los casos para que la ejecución sea mas rápida, aunque los datos no serán los mismos que generamos nosotros para el documento, ya que para eso usamos el programa `generador-datos.py` (con tamaño exponente 6). El tiempo de ejecución en este caso es de aproximadamente 20minutos.