

Comparación de Algoritmos de Ordenación

Vicente Javier Vidal-Abarca González

12 de junio de 2017

Resumen

Este documento, ha sido creado en \LaTeX , con el fin mostrar el conocimiento que poseo para crear un documento con dicha herramienta. Además, de guardar un registro de control usando GIT. Para demostrarlo, voy a realizar un pequeño código en python, que comparará el tiempo que tarda en ordenar una lista de un tamaño predefinido, usando tanto el algoritmo de Burbuja, como el Quicksort.

Índice

1. Introducción	3
1.1. Programas usados	3
2. GIT	4
2.1. git init	4
2.2. git add	4
2.3. git commit	4
2.4. git checkout	4
2.5. git log	4
2.6. git status	4
2.7. gitg	5
2.7.1. GitHub	5
2.7.2. Añadir el repositorio remoto	5
2.7.3. Guardar en el repositorio remoto	5
2.8. Otros	5
3. L^AT_EX	6
4. Python IDLE	7

1. Introducción

Bienvenido a este documento. Como bien esta explicado en el resumen, este documento ha sido realizado en \LaTeX , y con el mostraré el desarrollo de esta práctica.

1.1. Programas usados

En esta práctica, he usado 3 programas principalmente:

- GIT: Repositorio usado para almacenar las pruebas de control de la práctica.
- \LaTeX : Herramienta usada para crear este documento PDF eficientemente.
- Python IDLE: Entorno donde he desarrollado el código y las pruebas necesarias para realizar este proyecto.

2. GIT

Aquí se explicará los comandos mas utilizados, y como he planteado el desarrollo de este documento haciendo uso del control proporcionado por GIT.

2.1. git init

Usando el comando `git init <carpeta>` preparamos una carpeta con un proyecto de git vacío para comenzar a usar GIT.

En mi caso, he usado una carpeta nombrada por el nombre de “DIS”.

2.2. git add

Usando el comando `git add <archivo>` añadimos archivos al “index” (también conocido como HEAD). En este index, se hace una “previa” de como se encontrará la carpeta cuando hagamos un guardado real, usando el comando que se explicará mas adelante de `git commit`.

2.3. git commit

Usando el comando `git commit -m “Mensaje del commit”`, se realizan los cambios definitivamente en el repositorio de GIT. Con el parámetro `-m`, introducimos un pequeño texto para tener una orientación de que ha realizado dicho guardado.

2.4. git checkout

El comando `git checkout`, sirve para cambiar entre distintas ramas del proyecto.

Además, también sirve para crear ramas usando `git checkout -b <nueva_rama>`, o bien para borrarlas con `git checkout -d <rama_a_borrar>`.

2.5. git log

El comando `git log` muestra un pequeño resumen de los commits que han sido emitidos en la rama en la que te encuentres.

2.6. git status

El comando `git status` muestra el estado de los archivos que hay en la rama en la que te encuentres. Especifica si un archivo ha sido borrado, añadido o modificado desde el último commit realizado en esa misma rama.

2.7. gitg

Este comando lo usamos en el laboratorio de prácticas, y sirve para poder comprobar gráficamente como se encuentra el proyecto de GIT. Que ramas tiene, que archivos tiene cada rama, etc. Además de esto, también se puede hacer desde la herramienta gráfica los commits y otros comandos de GIT.

Sin embargo, yo no he podido usar esta herramienta gráfica, y por lo tanto he usado un repositorio remoto de github, creado específicamente para esto. Para realizar dicho repositorio:

2.7.1. GitHub

Primero, en la página de `github.com` he creado una cuenta (con nombre de usuario `karatekav`), y he creado un repositorio con el mismo nombre que tiene la carpeta que uso GIT, es decir, `DIS`.

2.7.2. Añadir el repositorio remoto

Una vez creado el repositorio, he añadido a la carpeta este mismo. Esto ha sido con el uso de:

```
git remote add origin https://github.com/<nombre_usuario>/<nombre_rep>.git
```

Que en mi caso específico ha sido:

```
git remote add origin https://github.com/karatekav/DIS.git.
```

2.7.3. Guardar en el repositorio remoto

Para guardar en el repositorio añadido como `origin`, es tan simple como realizar el comando de `git push`. Para ello:

```
git push origin <rama_a_guardar>.
```

Como la rama principal es `master`, la mayoría de guardados se realizaran con el siguiente comando:

```
git push origin master.
```

2.8. Otros

Además, he creado el archivo `.gitignore` para no guardar en GIT los archivos que son necesarios para la creación del PDF autogenerados por el compilador de `LATEX`.

3. L^AT_EX

En esta sección explicaré a grandes rasgos las cosas que más he utilizado en L^AT_EX.

He usado `\maketitle` con los parámetros de `\title`, `\author` y `\date` para crear la primera parte de este documento.

Además para poner todos estos comandos de L^AT_EX, estoy usando `\verb`. Si es para poner un comando de otras herramientas como GIT, he usado el comando de `\texttt{<comando>}`.

Con esto mencionado, y usando el bloque de `\begin{abstract} <resumen> \end{abstract}`, se crea el pequeño resumen que también se encuentra en la primera página de este documento.

Uso `\tableofcontents` para crear el índice. Además, he incluido el paquete `hiperref`, usando `[hidelinks]` para quitar el enlace en color rojo.

En general, uso `\newpage` para introducir un salto de página y `\bigskip` para crear una línea en blanco entre párrafos distintos, como se puede ver en esta parte del documento.

Uso los comandos de `\section{<nombre_sección>}`, `\subsection{<nombre_subsección>}` y `\subsubsection{<nombre_subsubsección>}` para crear los apartados que funcionarán con el índice.

Además de eso, para crear un apartado de una lista de objetos, usaremos la siguiente estructura: `\begin{itemize} \item <Nombre_item> \end{itemize}`, donde cada `\item <Nombre_item>` corresponde a una nueva línea que tendrá la lista.

Un ejemplo realizado en este documento es en el apartado de introducción, cuando se especifican las 3 herramientas principales usadas en esta práctica.

Además de estos comandos principales, quiero destacar el uso de herramientas externas (make con F12), y snippets que he ido personalizando, y que son muy útiles para este desarrollo. Por ejemplo, un snippet que es “sk”, que al usarlo se transforma en `\bigskip`.

4. Python IDLE

Python IDLE es el entorno donde he desarrollado el código en python donde voy a comparar los dos algoritmos de ordenación.

Los dos algoritmos que he utilizado son:

1. **Burbuja:** Revisa cada elemento de la lista que va a ser ordenada con el siguiente, intercambiándolos de posición si están en el orden equivocado.
2. **Quicksort:** Se basa en el algoritmo de divide y vencerás. Se basa en dividir el problema en dos mitades iguales, y hacer llamadas recursivas a si misma con cada una de estas partes, hasta llegar a un problema tan pequeño que sea directo.

Para comprobar la potencia de estos algoritmos, se crea una lista aleatoria de números, de 0 a n-1. Siendo n el número total de elementos de la lista.

Como tamaños de entrada (el número n comentado anteriormente), tenemos los siguientes:

500, 1000, 2000, 5000, 10000, 20000, 30000, 40000 y 50000.

Por último, usando la librería de python `matplotlib`, pintaré una gráfica de los resultados en tiempo obtenidos, usando la función `time` de la librería `time` de python.

5. Conclusión

La herramienta de L^AT_EX es bastante útil y con más experiencia y mucha práctica se pueden generar muy buenos documentos PDFs, y muy rápido.

Cabe destacar, que funciones del gedit aprendidas en clase, como por ejemplo los snippets y las external tools de gedit, son muy muy útiles y un gran descubrimiento para el desarrollo de cualquier proyecto.