Aquesta obra està subjecta a una llicència Attribution-NonCommercial-ShareAlike 3.0 Spain de Creative Commons. Per veure'n una còpia, visiteu http://creativecommons.org/licenses/by-nc-sa/3.0/es o envieu una carta a Creative Commons, 171 Second Street, Suite 300, San Francisco, California 94105, USA.

Pràctica 7: Estudi dels temps d'execució

Tecnologia de la Programació — iTIC

Aleix Llusà-Serra Sebastià Vila-Marta

7 de maig de 2012

Índex

1 Introducció

1.1 Objectius

Els objectius d'aquesta pràctica són:

- Consolidar els coneixement sobre cost en temps adquirits a les classes de teoria
- Conèixer algunes funcionalitats més de les moltes disponibles a la llibreria de `Python`.
- Consolidar l'ús de les eines de test i el disseny basat en tests.
- Consolidar l'ús de les eines de gestió de versions.

1.2 Condicions

- La pràctica té una durada de dues sessions de laboratori.
- Cal fer la pràctica amb l'equip de treball.
- Cal fer el desenvolupament usant control de versions amb subversion sobre el dipòsit `escriny3`. L'ús de l'eina serà part de l'avaluació de la pràctica.
- Aneu amb compte! les proves de temps heu de fer-les sempre en el mateix computador!

1.3 Lliurament

Caldrà lliurar el resultat de la pràctica a través de l'activitat escaient d'Atenea. El lliurament haurà d'incloure:

- Un informe escrit, en format pdf, en que es reflecteixin els resultats de la pràctica. Es valorarà la correctesa de l'informe (estructura, llenguatge, continguts, referències, etc.).

1.4 Context

La pràctica comença experimentant amb un algoritme d'ordenació que sotmet a diversos tests de temps. Posteriorment se'n fa una anàlisi teòrica en cas pitjor i es tracten les dades obtingudes experimentalment per veure si coincideixen amb el model teòric.

2 Preparació de l'entorn de treball

Prepareu l'entorn de treball on desenvolupareu el projecte. Useu, com en els darrers projectes el sistema de control de versions. En aquest projecte no us caldrà generar documentació automàtica però sí que us caldrà escriure un informe dels resultats.

Tasca 1 Prepareu l'entorn de treball per a la practica: doneu d'alta un nou directori de nom `p7` en el vostre dipòsit de versions i seguiu la mateixa estratègia de treball que en la pràctica anterior.

3 Ordenació de llistes

Una família molt important d'algoritmes és la constituida pels algoritmes d'ordenació. Essencialment podríem dir que són algoritmes que donada una llista d'elements són capaços d'ordenar-la seguint determinat criteri.

Dins d'aquesta família, un algoritme ben conegut és el *bubble sort*. El seu principi de funcionament és molt senzill. Imagineu la llista l amb elements que van de l_0 a l_k en un ordre arbitrari. Assumiu per simplicitat que $l_i \in \mathbb{N}$. Per ordenar l en sentit creixent cal aplicar el següent mètode:

1. Per a cada i des de 0 fins $k-1$, si $l_i > l_{i+1}$ intercanvieu els valors d'l_i i l_{i+1}.
2. Si heu fet algun intercanvi, repetiu el pas anterior. En cas contrari, la llista ja està ordenada.

Tasca prèvia 2 En un paper, mireu d'aplicar aquest algoritme a una llista petita tot intentant entendre el seu funcionament.

TASCA 3 Implementeu un petit programa en `Python` que ordeni una llista seguin aquest mètode.

ATENCIÓ: trobareu aquest algoritme implementat a molts llocs. No us demanem que el copieu, sinó que l'implementeu vosaltres mateixos.

A continuació definiu els doctests corresponents, assegureu-vos que són complets i funcionen correctament.

4 Eines de mesura

Per mesurar el comportament de l'algoritme que heu dissenyat és necessari que dissenyeu prèviament algunes eines. En essència han de servir per generar bancs de dades aleatòries.

Implementeu una funció per crear una llista d'n nombres naturals aleatoris amb valors entre 0 i $2n$ i uniformement distribuits. A tal efecte, serviu-vos del mòdul de la llibreria de Python anomenat `random`.

> TASCA PRÈVIA 4 Estudieu el mòdul `random` de la llibreria de `Python`. Proveu els seus mètodes i feu-vos a la idea de com s'utilitza.

Implementeu dues funcions més. Una per fer un reset del rellotge i una altra per escriure quan temps (en segons) ha passat des del darrer reset del rellotge.

> TASCA PRÈVIA 5 Pel rellotge inspireu-vos en els exemples de teoria i feu servir la documentació del mòdul `time` si és necessari.

TASCA 6 Implementeu en el mòdul `eines` les funcions per a crear llistes i per a gestionar el rellotge. A continuació definiu els doctests corresponents, assegureu-vos que són complets i funcionen correctament.

5 Experimentació

5.1 Primeres mesures

En aquesta tasca es tracta de prendre mesures sobre el funcionament de l'algoritme que heu dissenyat. Primerament, modifiqueu el vostre algoritme d'ordenació per tal que:

1. Generi una llista de 100 elements usant el mòdul `eines`.

2. L'ordeni 10 vegades i mesuri el temps que triga en fer 1 ordenació: mesureu el temps de les 10 ordenacions i dividiu per 10. Tingueu en compte que cal ordenar exactament la mateixa llista cada vegada que es fa una ordenació. Altrament la mesura no serviria.

Ara repetiu el procés anterior $k = 100$ vegades per cent llistes diferents però sempre de mida $n = 100$. Amb això obtindreu Una mostra de 100 dades diferents $M_n = \{t_1, \ldots, t_k\}$. Estudieu els principals indicadors descriptius d'aquesta mostra.

Feu ara el mateix procés per $n = 1000$ i $k = 1000$. Estudieu també M_{1000}.

TASCA 7 Implementeu els experiments descrits anteriorment en el programa principal. A tal efecte dividiu convenientment en funcions el vostre programa. Observeu que ens diuen les mostres sobre el funcionament de l'algoritme d'ordenació.

5.2 Estudiem el comportament en funció d'n

Modifiqueu de nou el programa anterior. Ara fixarem k a 100 i anirem variant el valor de n. Per cada valor de n obtindrem una mostra M_n però únicament ens interessarà el valor de $\bar{M}_n$.

El programa modificat ha de crear un fitxer de dades anomenat `dades.dat` en que va enregistrant, per cada valor de n, una línia en que consten el valor de n i el de $\bar{M}_n$.

Finalment el programa en qüestió ha de generar dades pels següents valors de n:

1. n de 0 a 900 amb increments de 100.
2. n de 1000 a 9000 amb increments de 1000.
3. n de 10000 a 90000 amb increments de 10000.
4. n de 100000 a 500000 amb increments de 100000.

TASCA 8 Comproveu que les dades obtingudes siguin raonables. Llegiu-les usant `oocalc` i feu-ne una gràfica amb n en l'eix de les abscisses. Diríeu que és una funció lineal? Quina mena de temps s'està mesurant?

6 Estudi teòric

6.1 Calculem el cost teòric

Analitzeu el cost teòric en cas pitjor de l'algoritme que heu dissenyat. De quin ordre és aquest algoritme? Què significa això exactament?

6.2 Ajustem el model

Ara que sabem quin és el cost teòric el que farem és ajustar les dades obtingudes, que tenim emmagatzemades en el fitxer `dades.dat`, al polinomi corresponent.

A tal efecte cal que useu un mòdul de Python anomenat `numpy` que potser haureu d'instal·lar prèviament (A Debian el paquet s'anomena `python-numpy`). Aquest mòdul conté una funció anomenada `polyfit` que us permetrà ajustar les dades que teniu al model teòric.

TASCA 9 Mireu-vos la documentació de la funció `polyfit`, [Com12], fixeu-vos especialment en l'exemple. Amb l'ajuda d'aquesta funció, ajusteu les dades obtingudes al model teòric.

7 Redacteu l'informe

Finalment, redacteu amb cura un informe en que es descrigui l'experiment fet, les condicions, materials i resultats. Afegiu-hi també el gràfic del model ajustat tal i com es fa en l'exemple que trobareu a la documentació de `polyfit`.

Referències

[Com12] The SciPy Community. *NumPy and SciPy Documentation.* numpy.polyfit. Anglès. Versió 1.7dev. 2012. URL: `http://docs.scipy.org/doc/numpy/reference/generated/numpy.polyfit.html` (visitat el 7 de maig de 2012).