Corso di Laurea in Informatica
Insegnamento di **Linguaggi dinamici**
Progetti d'esame validi per l'A.A. 2018/19

December 21, 2018

Compressore Lempel-Ziv-Welch Il progetto consiste nella stesura di un programma di compressione e di un corrispondente programma di decompressione (entrambi scritti in Python) eseguibili da linea di comando attraverso l'invocazione di opportuni script. L'algoritmo di compressione da utilizzare è quello ideato da Lempel e Ziv, nella versione data succesivamente da Terry Welch. Descrizioni precise di questo algoritmo sono reperibili in vari documenti di pubblico dominio. Esso è anche dettagliatamente descritto in appunti sulla compressione di testi preparati dal docente e disponibili nel repository *github* relativo all'insegnamento *Linguaggi dinamici* (file `appunti_compressione.pdf`).

La versione descritta (e anche ciò che è richiesto allo studente) non include il controllo del rapporto di compressione e, dunque, neppure la ricostruzione della tabella delle sottostringhe.

La sinossi dei comandi da implementare (ridotta rispetto a quella di `compress/uncompress` di Unix/Linux) è la seguente:

```
compress [ -v ] [ -r ] [file ...]
```

```
uncompress [-r] [file ...]
```

Con l'opzione `-v`, `compress` stampa, per ogni file specificato, un messaggio con la percentuale di spazio risparmiato oppure, nel caso in cui il file compresso abbia dimensioni non minori del file originale, un messaggio che notifica che non è stata effettuata compressione. Con l'opzione `-r`, nel caso in cui uno o più file siano directory, i programmi devono "scendere" nelle directory e comprimere/decomprimere tutti i file (regolari)/file compressi ivi presenti. I file compressi vengono sostituiti

con file compressi di estensione `.Z`. Nelle operazioni di compressione e decompressione vengono preservati i diritti sui file.

I programmi devono superare lo unit test "di sanità" presente nel repository *github* (file `LZW_unit_test.py`).

Simulatore di un sistema con q code e s serventi Il progetto consiste in una variante del simulatore visto a lezione. Gli arrivi dei job al sistema sono spaziati mediante utilizzo di deviazioni esponenziali negative di parametro λ_a. A differenza del simulatore presentato in classe, i job sono caratterizzati da una priorità. Le priorità sono identificate da un numero intero fra 1 e q, dove q è specificabile dall'utente; a valore più alto si intende priorità più bassa. Ad ogni job viene attribuita una priorità utilizzando deviazioni uniformi nell'intervallo di interi $[1, q]$. A seconda della priorità, il job viene inserito in una corrispondente coda. I job vengono mandati in esecuzione, non appena ci sono server disponibili, prima in base alla priorità e poi in base all'ordine di arrivo.

I tempi di esecuzione dei job sono stabiliti ancora utilizzando deviazioni esponenziali negative, questa volta con parametro λ_e. I server sono identici ed è contemplata pre-emption solo per i job di priorità 1 (considerati real-time job).

La durata della simulazione è misurata in *unità di tempo*. Quest'ultima nozione è in realtà solo implicitamente definita dal valore delle quantità λ_a e λ_e, che specificano, rispettivamente, frequenza degli arrivi e durata dei servizi, appunto "nell'unità di tempo". La durata della simulazione deve quindi essere sufficientemente ampia da poter raccogliere un numero statisticamente significativo di osservazioni.

Ricapitolando, i parametri in ingresso al simulatore devono essere:

1. frequenza degli arrivi λ_a;
2. numero delle priorità $q \geq 1$, $q \in \mathcal{Z}$;
3. velocità dei server, identica per tutti i server, e misurata come frequenza di servizio λ_e (nell'unità di tempo);
4. numero s dei server ($s \geq 1$);
5. durata della simulazione.

Il simulatore deve poi essere predisposto per calcolare le seguenti statistiche.

- media e deviazione standard della permanenza in coda dei job;

- media e deviazione standard della permanenza nelle varie code, a seconda della priorità.

Le statistiche devono essere raccolte per diversi valori dei parametri di ingresso. Fissate poi le seguenti quantità: $\lambda_a = n$ e $\lambda_e = 1$, si utilizzi il simulatore per determinare, al variare di n, intero positivo, il numero minimo di server che garantiscono, con grado di confidenza al 95%:

per i job real-time tempi di permenenza in coda inferiori al 10% dei tempi di interarrivo di tali job;

per gli altri job tempi di permanenza in coda inferiore all'unità.

Il programma deve superare gli unit test presenti nel repository *github* (file `Sim_unit_test.py`).

Progetto Interfaccia Web Il progetto consiste nello sviluppo di un applicativo Web che fornisca una interfaccia di accesso e visualizzazione ad un database pre-esistente (si utilizzi a tal fine il DB contenente dati sul campionato di calcio visto durante le lezioni).

Il progetto dovrà essere basato sul framework Django e collegarsi al DB esistente fornendo alcune funzionalità di base, il cui set minimo è specificato di seguito. Il sistema dovrà fornire una pagina di accesso principale che mostri l'elenco puntato delle funzionalità offerte, che saranno mostrate sotto forma di link che porterà alle relative pagine Web.

Il sistema minimo di funzionalità dovrà includere:

- Accesso al calendario delle partite
- Accesso ai risultati delle singole giornate di campionato
- Accesso alla schedina risultante per ogni giornata di campionato
- Accesso alla classifica finale
- Accesso a semplici statistiche per ogni squadra (es. numero di partite giocate, vinte, perse, punti, goal fatti e goal subiti)

Non vi sono requisiti di tipo grafico. Il progetto sarà giudicato in base alla correttezza delle funzionalità offerte e della logica di navigazione.

I progetti devono essere distribuiti come package Python 3 installabili con `setup` di `setuptools`. Eventuali script addizionali (certamente necessari per il progetto del compressore) devono essere "bundled" con il software

Python. A riguardo, supponiamo che tali script si trovino nella directory `project_root/scripts` (dove `project_root` è la directory home del progetto, contenente anche il file `setup.py`); in tal caso `setup.py` dovrà includere la seguente riga:

```
scripts=['scripts/script1', 'scripts/script2', ...]
```