Modulo III + Modulo IV

N. Galesi, S. Guerrini
Lab Prog. A.A. 2004/2005 – II Semestre

1 Modulo III

Il modulo III si occupa di calcolare il PageRank degli ipertesti. Il calcolo viene suddiviso in due fasi. Inizialmente, per ogni pagina calcoliamo un valore reale seguendo la formula del PageRank di Google. Quindi trasformiamo questi valori in un scala da 1 a 100.

Sia A un ipertesto e siano $T_1, \ldots, T_k$, tutti gli ipertesti che contengono almeno un link ad A. Il page rank $PR(A)$ di A si ottiene applicando la formula:

$$PR(A) = (1-d) + d \cdot \left(\frac{M(T_1, A) \cdot PR(T_1)}{C(T_1)} + \ldots + \frac{M(T_k, A) \cdot PR(T_k)}{C(T_k)} \right) \quad (*)$$

- dove d è un parametro che verrà fissato ad un valore di 0.85,
- $C(T_i)$ è il numero totale di link in uscita della pagina T_i (la somma delle molteplicità degli archi uscenti da T_i nel grafo delle pagine),
- $M(T_1, A)$ è il numero di link dalla pagina T_i alla pagina A (la molteplicità dell'arco da T_i ad A nel grafo delle pagine).

Possiamo pensare al PageRank di tutte le pagine come ad un vettore PR di reali di dimensione pari al numero di ipertesti. Il calcolo del PageRank di tutte le pagine web è quindi dato dalle soluzioni del sistema di equazioni ottenuto usando la precedente uguaglianza per tutti gli ipertesti.

Supponiamo di avere tre pagine A, B e C e che la pagina A abbia un link a B e a C, che B abbia un link a C e che C abbia un link ad A. Fissiamo per semplicità $d = 0.5$. Le tre equazioni che definiscono il PageRank sono:

$$\begin{aligned} PR(A) &= 0.5 + 0.5 * (PR(C)) \\ PR(B) &= 0.5 + 0.5 * (PR(A)/2) \\ PR(C) &= 0.5 + 0.5 * (PR(B) + PR(A)/2) \end{aligned}$$

In questo semplice sistema è facile trovare le soluzioni per $PR(A)$, $PR(B)$ e $PR(C)$. In generale però, dato l'alto numero di equazioni e la possibilità di circolarità dei link tra pagine web, non sarà sempre possibile trovare una soluzione risolvendo il sistema (si veda l'introduzione al progetto per ulteriori dettagli).

Per calcolare una soluzione al sistema, si userà quindi un algoritmo che calcola una soluzione *approssimata*. Tale algoritmo calcola iterativamente soluzioni del sistema sempre migliori e sempre più "vicine" alla soluzione.

PageRank per approssimazioni successive

Sia dato un valore $\epsilon << 1$, per esempio $\epsilon = 0,000001$. Sia PR il vettore dei PageRank. Un algoritmo per approssimazioni successive calcola una soluzione approssimata del sistema (un vettore) come segue:

1. inizialmente considera un vettore dato (per esempio, $PR = [1, 1, \ldots, 1]$);
2. usando tutte le equazioni del sistema $(*)$, continua a calcolare un nuovo vettore soluzione PR' fino a quando, per ogni coordinata, la differenza tra il valore di quella coordinata nel nuovo vettore e il valore di quella coordinata nel vettore precedente è sufficientemente piccola, ovvero è $<$ di ϵ. Formalmente, le iterazioni terminano quando:

$$\max_{i=1,\ldots,NUMPAGE}(|PR'[i] - PR[i]|) < \epsilon$$

1.1 Valori di Pagerank in scala

Avendo in genere molti ipertesti da analizzare, conviene normalizzare il PageRank riportandolo ad un valore intero compreso in una scala che va (per esempio) da 1 a 100.

1.2 Cosa fare nel modulo III

Nel Modulo III si deve calcolare il PageRank degli ipertesti usando l'algoritmo descritto prima. Le equazioni del sistema $(*)$ sono calcolate usando il grafo dei link, e quindi le informazioni contenute nelle liste `lista_in` e `lista_out` del record `struct info_ipertesto`. Per memorizzare i valori di `PR` delle pagine, si è aggiunto alla `struct info_ipertesto` (definita in `nodes.h`) il campo

```
double PR[2];
```

In questo campo si possono memorizzare il nuovo valore di `PR` che si sta calcolando e quello ottenuto all'iterazione precedente. Il procedimento approssimato di calcolo termina quando questi due valori differiscono per meno della costante `EPSILON`, definita nella nuova versione del file `nodes.h`.

Una volta calcolato `PR`, i valori devono essere riportati nella scala descritta e inseriti nel campo `pagerank`.

Il modulo, organizzato in un unico file di nome `modulo3.c` deve contenere una funzione

```
struct table *pagerank(struct table *, int)
```

che riceve in input il dizionario degli ipertesti calcolato nei moduli I e II e un intero contenente il massimo numero di iterazioni consentito. La funzione usa le informazioni contenute nel dizionario degli ipertesti e lo restituisce modificato dopo aver calcolato il PageRank di tutte le pagine usando l'algoritmo descritto prima. Restituisce NULL nel caso in cui nelle iterazioni consentite non sia stata trovata una soluzione approssimata.

2 Modulo IV

Il modulo IV si occupa di calcolare il *Dizionario Globale delle Parole* (DGP). Le entry del DGP sono tutte le differenti parole trovate in tutti gli ipertesti, insieme a certe informazioni sulle pagine in cui tali parole occorrono.

Per implementare il DGP useremo una *tabella hash*. Per costruire e gestire tale tabella useremo le dichiarazioni e i metodi sviluppati in `hashtab.h` e `hashtab.c`.

```
HASHTABLE diz_glob;
```

A differenza del Dizionario degli Ipertesti, per il DGP, il campo `key` del record `struct tabelem` conterrà la parola da inserire nel dizionario.

Il campo `info` nel DGP

Per ogni parola nel DGP abbiamo bisogno di poter accedere ai seguenti dati:

1. dati sugli ipertesti in cui tale parola occorre;
2. dati su tutte le occorrenze di quella parola negli ipertesti in cui la parola occorrre.

Tale informazione verrà quindi implementata mediante una lista in cui ogni elemento contiene

1. un puntatore al nome dell'ipertesto che contiene la parola,
2. un puntatore al record che contiene le informazioni sugli ipertesti, (`struct info_ipertesto`), e
3. un puntatore al record che contiene le informazioni riguardanti la data parola nel dato ipertesto, `struct elem_diz_loc`.

```
struct info_parola {
       char *it_name;
       struct info_ipertesto *it;
       struct elem_diz_loc *el_it;
       struct info_parola *next;
};
```

Quindi, nel caso del DGP, il campo `info` del record `tabelem` deve essere forzato al tipo `struct info_parola *` e contenere un puntatore alla lista delle informazioni sulla parola.

2.1 Cosa fare nel modulo IV

Il modulo si compone di due file. Il file `modulo4.c` in cui si deve implementare una funzione

```
struct dgp *crea_dgp(struct table *)
```

che riceve in input il dizionario degli ipertesti e lo analizza per costruire il dizionario globale delle parole, più tutte le altre funzioni che si ritengono utili, a parte quelle per la gestione della tavola hash che dovranno essere implementate a parte (si veda dopo). Il DGP viene implementato insieme con il dizionario degli ipertesti nella seguente struttura:

```
struct dgp {
       struct table *tb;
       HASHTABLE diz_glob;
}
```

Il secondo file da implementare nel modulo è il file `dgp.c` in cui sono implementate le funzioni

```
struct dg *dgp_init(struct table *t);
```

che si occupa di inizializzare il dgp;

```
struct tabelem *dgp_ins(struct table *dgp, char *word);
```

che si occupa di inserire un nuovo elemento nel dgp;

```
struct tabelem *dgp_search(struct dgp *dgp, char *word);
```

che si occupa di cercare un elemento nel dgp.

Le funzioni implementate in `dgp.c` sono l'interfaccia del modulo IV con il modulo I (nel quale è stata implementata la tavola hash). Il file `dgp.c` deve essere l'unica parte del modulo IV dipendente dall'implementazione dall'implementazione di `HASHTABLE`, più precisamente, deve essere l'unica parte del modulo in cui si richiamano le funzioni di creazione, inserimeno e ricerca della tavola hash implementate in `hashtab.c` e i cui prototipi sono stati forniti in `hashtab.h`.

I prototipi delle funzioni da implementare e le definizioni necessarie all'implementazione del DGP sono contenute nell'header file `dgp.h` fornito dai docenti.