UNIVERSITÀ DEGLI STUDI DI ROMA TRE
FACOLTÀ DI SCIENZE M.F.N.

Un problema di cammino minimo ed una sua applicazione al World Wide Web

Sintesi della tesi di Laurea in Matematica
di Daniela Fabrianesi
Relatore: Dott. Marco Liverani

Le reti di calcolatori, ed in particolare quelle basate sul protocollo TCP/IP, hanno assunto negli ultimi anni un'importanza sempre maggiore fino ad avere un impatto consistente nella vita di tutti i giorni. Questo tipo di tecnologie sono presenti fin dagli anni '70, ma mai come negli ultimi $5-6$ anni hanno riscosso tanto successo. Il fattore determinante in questa esplosione di popolarità è sicuramente la rete Internet con i suoi servizi di posta elettronica ed il World Wide Web.

L'obiettivo di questa tesi, e la parte originale del lavoro, è stato quello di progettare e realizzare uno strumento *software* che, costruendo prima un grafo che rappresenti la struttura dei collegamenti di una porzione del World Wide Web e poi, ricercando i cammini minimi tra ogni sua coppia di vertici, potesse rivelarsi un utile strumento nell'analisi e nella progettazione (o nel mantenimento) di grandi siti web. Infatti, dalla conoscenza di cammini

minimi si possono ottenere delle informazioni interessanti: come il "grado di correlazione" tra le pagine web, dato che è probabile che documenti con un contenuto semantico vicino siano collocati a "breve distanza" tra loro nella struttura del sito web, o come l'irraggiungibilità di una pagina a partire da un'altra.

Il primo capitolo del nostro lavoro presenta sinteticamente i principali temi riguardanti le reti di calcolatori, viene approfondito lo studio del protocollo di comunicazione TCP/IP ed in particolare ciò che riguarda l'indirizzamento delle risorse sulla rete ed il DNS. Tutto questo ci è servito a realizzare con maggiore consapevolezza la parte di *networking* del programma: infatti, per visitare il World Wide Web è necessario sviluppare una procedura automatica che si connetta ai diversi server sparsi sulla rete e ne estragga l'informazione desiderata. A tal fine il programma sviluppato sfrutta la tecnologia dei *socket*. La principale fonte di documentazione per questa parte di lavoro sono stati alcuni testi di *networking* ([2], [4]), che presentano l'argomento con un approccio piuttosto tecnico ed ingegneristico.

Il secondo capitolo fornisce una panoramica, anche in questo caso molto sintetica, vista la vastità dell'argomento, sul World Wide Web. Partendo dal concetto di ipertesto distribuito in rete, si affrontano il linguaggio di marcatura usato per codificare tale ipertesti (HTML), il protocollo di comunicazione di alto livello (HTTP) utilizzato per rendere accessibili le risore su una rete ampia ed eterogenea come, ad esempio, la rete Internet, e i meccanismi di indirizzamento delle risorse (URL e URI). Conoscere gli elementi principali del linguaggio HTML e la grammatica usata per comporre le URL è servito per implementare nel programma la parte di *parsing* ed analisi delle pagine scaricate mediante l'uso del protocollo HTTP e per l'individuazione e la ricostruzione delle URL che costituiscono gli indirizzi delle pagine collegate e che nel grafo vengono rappresentati come vertici adiacenti al vertice corrente. Per questo argomento le fonti bibliografiche sono stati alcuni testi di taglio divulgativo ([1], [5], [6], [7], [8]), altri di carattere più tecnico [14] ed un insieme di articoli ([3], [9], [10], [11], [12], [13], [15]), noti come RFC (*Request For Comments*), che forniscono una definizione accurata e formale di ogni aspetto che riguarda i protocolli "aperti" utilizzati in rete.

Il terzo capitolo ha un taglio più teorico e formale e ci permette di introdurre alcuni concetti elementari della teoria dei grafi utilizzati ampiamente anche nel capitolo successivo. Quindi viene introdotto il problema della ricerca di un cammino minimo su grafi pesati e vengono presentati alcuni dei principali e più interessanti algoritmi per risolvere in un tempo accettabile il problema (tali algoritmi sono, quindi, caratterizzati da una bassa complessità computazionale).

Nel quarto ed ultimo capitolo viene presentato un algoritmo molto efficiente, dovuto ad Ausiello, Italiano, Spaccamela e Nanni, che grazie alla costruzione di strutture dati opportune, è in grado di ottimizzare l'aggiornamento delle informazioni relative ai cammini minimi tra tutte le coppie di vertici di un grafo, in seguito all'operazione di inserimento di nuovi spigoli. Questo algoritmo ci fornisce quindi uno strumento molto valido per mantenere aggiornate le informazioni di nostro interesse, senza dover visitare nuovamente il grafo. Infine, al termine del quarto capitolo, viene presentata la struttura del programma realizzato (il cui listato in linguaggio C per sistemi UNIX è riportato in appendice) e vengono evidenziati le maggiori criticità incontrate. Le fonti bibliografiche su cui si sono basati i capitoli tre e quattro sono principalmente alcuni testi di informatica teorica e di teoria degli algoritmi ([16], [17]) e gli articoli originali ([18],[19], [20]) in cui sono stati descritti alcuni degli algoritmi presentati.

Nella progettazione e nella realizzazione del nostro programma abbiamo fatto uso di strutture dati dinamiche in grado di supportare inserimenti ed eliminazioni di spigoli e/o di vertici di un grafo. Infatti, se il grafo rappresenta un ipertesto (una porzione del World Wide Web), l'operazione di aggiornamento di spigoli riflette i reali cambiamenti della rete dovuti ai collegamenti che vengono attivati o disattivati o all'inserimento di vertici che, nel nostro caso, rappresentano la creazione di nuove pagine web durante il periodo di vita di un sito.

Per rappresentare l'insieme delle pagine web prese in considerazione durante l'esecuzione dell'algoritmo realizzato abbiamo utilizzato i grafi. I grafi sono strutture dati molto diffuse in informatica e di conseguenza gli algoritmi per

lavorare con tali strutture sono di fondamentale importanza in questo campo. Richiamiamo definizioni e proprietà fondamentali della teoria dei grafi.

Definizione 1. *Un* **grafo orientato** G *è una coppia* (V, E)*, dove* V *è un insieme finito ed* E *è una relazione binaria su* V*. L'insieme* V *è chiamato l'***insieme dei vertici** *(o* **insieme dei nodi***) di* G *ed i suoi elementi sono detti* **vertici***(o* **nodi***). L'insieme* E *è chiamato l'***insieme degli spigoli** *(o* **insieme degli archi***) di* G *ed i suoi elementi sono detti* **spigoli** *(o* **archi***).*

Definizione 2. *In un* **grafo non orientato** $G = (V, E)$*, l'insieme degli spigoli* E *è costituito da coppie non ordinate di vertici, piuttosto che da coppie ordinate; cioè, uno spigolo è un insieme* $\{u, v\}$*, dove* $u, v \in V$ *e* $u \neq v$*. Per convenzione si usa la notazione* (u, v) *per uno spigolo, piuttosto che la notazione insiemistica* $\{u, v\}$*; inoltre* (u, v) *e* (v, u) *sono considerati essere lo stesso spigolo.*

Definizione 3. *Se* (u, v) *è uno spigolo di un grafo orientato, si dice che* (u, v) *è* **incidente** *o* **esce dal** *vertice* u *ed è* **incidente** *o* **entra nel** *vertice* v*. Se* (u, v) *è uno spigolo di un grafo non orientato, si dice che* (u, v) *è* **incidente sui** *vertici* u *e* v*.*

Definizione 4. *Se* (u, v) *è uno spigolo di un grafo* $G = (V, E)$*, si dice che il vertice* v *è* **adiacente** *al vertice* u*. Quando il grafo è non orientato la relazione di adiacenza è simmetrica, mentre quando è orientato la relazione non è necessariamente simmetrica. Se* v *è adiacente ad* u *in un grafo orientato talvolta si scrive* $u \rightarrow v$*.*

Definizione 5. *Un* **cammino** *di* **lunghezza** k *da un vertice* u *ad un vertice* u' *in un grafo* $G = (V, E)$ *è una sequenza* $\langle v_0, v_1, v_2, ..., v_k \rangle$ *di vertici tale che* $u = v_0$*,* $u' = v_k$ *e* $(v_{i-1}, v_i) \in E$ *per* $i = 1, 2, ..., k$*. La lunghezza di un cammino è il suo numero di spigoli. Il cammino* **contiene** *i vertici* $v_0, v_1, v_2, ..., v_k$ *e gli spigoli* (v_0, v_1)*,* (v_1, v_2)*,...,*(v_{k-1}, v_k)*. Se vi è un cammino* p *da* u *a* u'*, si dice che* u' *è* **raggiungibile** *da* u *tramite* p*; ciò talvolta si scrive come* $u \overset{p}{\rightsquigarrow} u'$*. Un cammino è* **semplice** *se tutti i vertici sono distinti.*

Definizione 6. *Si definisce la* **distanza di cammino minimo** $\delta(u, u')$ *da* u *a* u' *come il numero di spigoli del cammino più breve dal vertice* u *al vertice*

u', oppure ∞ se non esiste nessun cammino da u a u'. Un cammino di lunghezza $\delta(u, u')$ da u a u' è chiamato **cammino minimo** *da u a u'.*

Definizione 7. *In un grafo orientato, un cammino $\langle v_0, v_1, ..., v_k \rangle$ forma un* **ciclo** *se $v_0 = v_k$ ed il cammino contiene almeno uno spigolo. Il ciclo è* **semplice***, se $v_1, v_2, ..., v_k$ sono distinti. In un grafo non orientato, un cammino $\langle v_0, v_1, ...v_k \rangle$ forma un* **ciclo** *se $v_0 = v_k$ e $v_1, v_2, ..., v_k$ sono distinti. Un grafo senza cicli è* **aciclico***. Spesso si usano le prime lettere del nome inglese "direct acyclic graph" (***dag***) per indicare un grafo orientato aciclico.*

Definizione 8. *Un grafo non orientato è* **connesso** *se ogni coppia di vertici è collegata con un cammino. Le* **componenti connesse** *di un grafo sono le classi di equivalenza dei vertici sotto la relazione "raggiungibile da".*

Definizione 9. *Un grafo orientato è* **fortemente connesso** *se ogni coppia di vertici è collegata con un cammino. Le* **componenti fortemente connesse** *di un grafo sono le classi di equivalenza dei vertici sotto la relazione "mutuamente raggiungibile".*

Definizione 10. *Un grafo aciclico e non orientato è una* **foresta** *e un grafo connesso aciclico e non orientato è un* **albero (libero)***.*

Il seguente teorema raccoglie molte importanti proprietà che riguardano gli alberi.

Teorema 1. *Sia $G = (V, E)$ un grafo non orientato. Le seguenti affermazioni sono equivalenti.*

1. *G è un albero libero.*
2. *Due vertici qualsiasi in G sono connessi da un unico cammino semplice.*
3. *G è connesso, ma se un qualunque spigolo è rimosso da E, il grafo risultante è sconnesso.*
4. *G è connesso e $|E| = |V| - 1$.*
5. *G è aciclico e $|E| = |V| - 1$.*

6. *G è aciclico, ma se un qualsiasi spigolo è aggiunto ad E, il grafo risultante contiene un ciclo.*

Definizione 11. *Un* **albero radicato** *T è un grafo orientato senza cicli che soddisfa le seguenti tre proprietà:*

1. *C'è soltanto un vertice, detto* **radice***, con nessun spigolo entrante.*
2. *Ogni vertice, escluso la radice, ha esattamente uno spigolo entrante.*
3. *C'è un (unico) cammino dalla radice ad ogni vertice.*

Definizione 12. *Si consideri un vertice x di un albero radicato T con radice r; qualunque vertice y sull'unico cammino da r a x è chiamato* **antenato** *di x e se y è antenato di x, allora x è un* **discendente** *di y (ogni vertice è sia un antenato che un discendente di se stesso). Se y è un antenato di x, e $x \neq y$, allora y è un* **antenato proprio** *di x e x è un* **discendente proprio** *di y. Il* **sottoalbero con radice in** *x è l'albero indotto dai discendenti di x, radicato in x.*

Definizione 13. *Se l'ultimo spigolo di un cammino dalla radice r di un albero T ad un vertice x è (y, x), allora y è il* **padre** *di x e x è il* **figlio** *di y; la radice è l'unico vertice in T che non ha padre. Se due vertici hanno lo stesso padre, si dicono* **fratelli***. Un vertice senza figli si dice* **foglia***. Un vertice non foglia è un* **vertice interno***.*

Definizione 14. *Il numero di figli di un vertice x in un albero T è chiamato il* **grado** *di x. La lunghezza del cammino dalla radice r ad un vertice x è la* **profondità** *di x in T; la profondità più grande di un qualsiasi vertice in T è l'***altezza** *di T.*

Definizione 15. *Dato un grafo orientato $G = (V, E)$ ed un vertice $x \in V$, un* **albero dei cammini in avanti radicato in** *x è un albero $T_f(x) = (X, S)$ radicato in x tale che*

1. *X è l'insieme dei discendenti di x in G;*
2. *$S \subseteq E$.*

Allo stesso modo, un **albero dei cammini indietro radicato in** x *è un albero* $T_b(x) = (X, S)$ *radicato in* x *tale che*

1. X *è l'insieme degli antenati di* x *in* G*;*
2. $S \subseteq E^R$, *dove* $E^R = \{\, (y, x) \mid (x, y) \in E \,\}$ *è detto l'***insieme degli spigoli inversi**.

Definizione 16. *Un albero dei cammini in avanti (indietro)* $T = (X, S)$ *radicato in* x *è di* **lunghezza minima** *(o equivalentemente è un* **albero dei cammini minimi in avanti (indietro)***) se per ogni vertice* $v \in X$*, la profondità di* v *in* T *coincide con la lunghezza del cammino minimo da* x *a* v *in* G*.*

In un *problema di cammini minimi* viene fornito un grafo orientato e pesato $G = (V, E)$, con una funzione peso $w : E \to \mathbb{R}$ che associa ad ogni spigolo un peso a valore nei reali.

Definizione 17. *Il* **peso** *di un cammino* $p = \langle v_0, v_1, ..., v_k \rangle$ *è la somma dei pesi degli spigoli che lo costituiscono:*

$$w(p) = \sum_{i=1}^{k} w(v_{i-1}, v_i).$$

Definizione 18. *Il* **peso di cammino minimo** *da* u *a* v *è definito come*

$$\delta(u, v) = \begin{cases} min\{\, w(p) : u \overset{p}{\rightsquigarrow} v\} & \text{se se esiste un cammino da } u \text{ a } v, \\ \infty & \text{altrimenti.} \end{cases}$$

Definizione 19. *Un* **cammino minimo** *dal vertice* u *al vertice* v *è definito come un qualunque cammino* p *con peso* $w(p) = \delta(u, v)$*.*

I pesi degli spigoli possono essere interpretati come misure diverse da distanze: essi sono usati spesso per rappresentare tempi, costi, penalità, perdite o qualunque altra quantità che si accumula in modo lineare e che si vuole minimizzare.

Osserviamo che un grafo non pesato può essere considerato come un grafo in cui ogni spigolo ha peso unitario.

In questo lavoro abbiamo presentato un algoritmo che può essere così brevemente spiegato: partendo da una home page (URL) fornita in input dall'utente, utilizzando la tecnologia dei *socket*, automaticamente si connette al server remoto su cui è disponibile tale risorsa e, mediante l'uso del protocollo HTTP, ne estrae la pagina sorgente (in linguaggio HTML). Inoltre, introduce un vertice nel grafo che corrisponde alla pagina web appena recuperata. Analizzando poi il contenuto di tale pagina esegue un *pattern matching* per individuare tutti i link, ad altri documenti HTML, presenti in essa sotto forma di tag `A`, contenenti HTTP URL in forma assoluta o parziale. Inoltre, inserisce tali link nel grafo rappresentandoli come spigoli che collegano i vertici adiacenti al vertice corrente.
Quindi, per ogni link vengono ripetute le stesse operazioni, per un massimo di NMAX volte (dove NMAX è un numero prefissato sufficientemente grande). Completata questa prima fase di "visita", partendo dal grafo G da essa ricavato, scegliendo uno alla volta tutti gli spigoli[1] (link) e aggiornando tutte le strutture dati calcola, qualora esista, un cammino minimo per ogni coppia di vertici del grafo.
Va da sé che una volta visitato per la prima volta il grafo, l'aggiunta di un nuovo spigolo non richiede di visitarlo tutto di nuovo, ma semplicemente di eseguire l'aggiornamento delle strutture dati relative ai vertici del nuovo spigolo; questa operazione ha una complessità molto bassa, dell'ordine di $N \log_2 N$ (dove N è il numero delle pagine visitate, quindi necessariamente minore o uguale a NMAX). Ciò è di grande utilità, perché compiere una nuova visita del grafo (cioè rieseguire la prima fase dell'algoritmo) è un'operazione che potrebbe richiedere tempi anche molto lunghi, vista la lentezza delle connessioni di rete dovuta al notevole traffico di dati sulla rete Internet.

Per mantenere aggiornate le strutture dati, che ci permettono di individuare i cammini minimi tra tutte le coppie di vertici, abbiamo utilizzato l'algoritmo ottimale di Ausiello, Italiano, Spaccamela, Nanni [18].

[1] che supponiamo essere di peso unitario

L'idea base di tale algoritmo consiste nell'aggiornare, durante l'inserimento di spigoli, una matrice $N \times N$ delle distanze $D = (d_{uv})$ tale che l'elemento d_{uv} contenga la lunghezza del più breve cammino da u a v, per ogni coppia di vertici (u, v). Per fare ciò si associa ad ogni vertice $v \in V$ l'insieme dei suoi discendenti, detto $DESC(v)$, e quello dei suoi antenati, detto $ANC(v)$. Per recuperare facilmente le informazioni sui cammini minimi, $DESC(v)$ è organizzato come un albero dei cammini minimi in avanti radicato in v e $ANC(v)$ come un albero dei cammini minimi indietro radicato in v. Inoltre, si usano due matrici $N \times N$ di puntatori, $F = (f_{uv})$ e $B = (b_{uv})$ definite come segue. f_{uv} contiene un puntatore al vertice v nell'albero dei cammini minimi in avanti $DESC(u)$ se il vertice v è un discendente del vertice u, altrimenti contiene lo speciale valore $NULL$. Allo stesso modo, b_{uv} contiene un puntatore al vertice v nell'albero dei cammini minimi indietro $ANC(u)$ se il vertice v è un antenato del vertice u, altrimenti contiene lo speciale valore $NULL$.
Un cammino minimo da u a v può essere individuato esaminando l'elemento f_{uv}: se è $NULL$, allora non esiste nessun cammino da u a v, altrimenti f_{uv} ci permette di localizzare il vertice v in $DESC(u)$, l'albero dei cammini minimi in avanti con radice in u. Percorrere il cammino da v alla radice u in $DESC(u)$, seguendo i puntatori "inversi" che da ogni vertice portano al padre, costa al più $O(k)$ operazioni, dove $k < N$ è la sua lunghezza. Il cammino $p_{v,u}$ individuato con questa procedura su $DESC(u)$ corrisponde ad un cammino minimo da u a v in G.

La dinamicità dell'algoritmo di Ausiello, Italiano, Spaccamela, Nanni, ossia la capacità di aggiornare la soluzione del problema di cammino minimo dopo cambiamenti dinamici (inserimenti di nuovi spigoli) invece di ricalcolarla da capo ogni volta, rende il nostro algoritmo molto efficiente; infatti:

Teorema 2. *Le strutture dati proposte nell'algoritmo di Ausiello, Italiano, Spaccamela, Nanni supportano ogni operazione* minpath*, che restituisce il cammino minimo tra una coppia di vertici, in un tempo $O(k)$, dove $k < N$ è la lunghezza del cammino tracciato. Il tempo totale richiesto per aggiornare le strutture dati durante l'inserimento di al più $O(N^2)$ spigoli è $O(N^3 \log_2 N)$, dove N è il numero di vertici del grafo. La quantità di memoria occupata*

dalle strutture dati è $O(N^2)$.

Per concludere riportiamo la descrizione, in pseudocodice, del nostro algoritmo e ne analizziamo la complessità globale:

```
1.  Assegna una home page P in input
2.  Q={P}; V={P}; c=0
3.  fintanto che Q non e' vuoto e c<NMAX
4.     estrai il primo elemento P di Q; c=c+1
5.     cerca tutti i link P' presenti in P
6.     per ogni P' che non appartiene a Q
7.          inserisci P' in Q; inserisci P' in V
8.     elimina P da Q
9.  fintanto che V non e' vuoto
10.    estrai il primo elemento P da V
11.    per ogni link P' presente in P
12.         aggiorna le strutture dati relative ai vertici P
            e P' come descritto nell'algoritmo di Ausiello
13.    elimina P da V
14. restituisci un cammino minimo (se esiste) per ogni coppia
    di vertici P e P'
```

Analizziamo ora la complessità del nostro algoritmo. Considerato che al massimo vengono effettuate $O(N)$ iterazioni del ciclo delle linee $3-8$, che le operazioni di inserimento (linea 7), di estrazione (linea 4) e di eliminazione (linea 8) richiedono ciascuna un tempo $O(1)$ e che l'esecuzione della linea 5 richiede al più $O(N)$ operazioni, l'esecuzione del ciclo delle linee $3-8$ richiede tempo $O(N^2)$. Per maggiore precisione è bene osservare che le pagine visitate potrebbero contenere link diretti a pagine che non appartengono all'insieme $\{P_1, ..., P_N\}$ delle pagine visitate dall'algoritmo. Quindi i cicli presenti nelle righe 5 e 6 potrebbero venire iterati più di N volte, se la pagina visitata contiene un numero N' di link, che puntano anche a pagine "esterne", maggiore di N. Tuttavia, se N è sufficientemente grande, è possibile individuare una costante $h << N$ tale che $N' < hN$. Quindi la complessità asintotica

dell'algoritmo non cambia.
L'esecuzione del secondo ciclo delle linee $9-13$ richiede $O(N^3 \log_2 N)$ (come è dimostrato nel Teorema 2).
Quindi, l'algoritmo richiede complessivamente un tempo $O(N^3 \log_2 N)$.
La quantità di memoria occupata dalle strutture dati è $O(N^2)$ dato che vengono utilizzate tre matrici di dimensione $N \times N$ e $2N$ alberi, ciascuno di dimensione al più N.

Se avessimo risolto il problema di trovare i cammini minimi tra tutte le coppie di vertici in maniera statica e non dinamica, cioè ricalcolando da capo la soluzione di tale problema ogni volta che si inserisce un nuovo spigolo, avremmo impiegato un tempo di ordine molto più alto. Ad esempio, se avessimo usato l'algoritmo di Floyd-Warshall [16], il totale tempo richiesto per trovare una soluzione del problema durante l'inserimento di al più $O(N^2)$ spigoli sarebbe stato $O(N^5)$; mentre la quantità di memoria occupata sarebbe stata $O(N^3)$.

Bibliografia

[1] Andrea Aparo, *Il libro delle reti*, ADN Kronos Libri, 1995.

[2] Craig Hunt, *TCP/IP network administration*, O'Reilly & Associates, 1994.

[3] J. Reynolds, J. Postel, *Assigned Numbers*, RFC 1700, USC/Information Sciences Institute, October 1994.

[4] Richard W. Stevens, *UNIX – Sviluppo del software di networking*, Prentice Hall International, Gruppo Editoriale Jackson, 1991.

[5] Frank Halasz, Mayer Schwartz, *The Dexter Hypertext Reference Model*, Communications of the ACM, Vol.37, No.2, February 1994, pp. 30-39.

[6] Jeff Coklin, *A Survey of Hypertext*, MCC Tech. Rep., No. STP-356-86, Rev.2, December 1987.

[7] Yehoshua Perl, Murray Turoff, *A Theoretical Framework for Collaborative Hypertext*, Proceedings of the Second Conferance on Computer-Supported Cooperative Work (Portland, Oregon 1988).

[8] Janet Fiderio, Mark Frisse, Michael L. Begeman, Jeff Conklin, *Hypertext*, Byte, October 1988, pp. 234-268.

[9] Tim Berners-Lee, *Universal Resource Identifiers in WWW: A Unifyng Syntax for the Expression of Names and Addresses of Objects on the Network as used in the World-Wide Web*, RFC 1630, CERN, June 1994.

[10] T. Berners-Lee, D. Connolly, *Hypertext Markup Language-2.0*, RFC 1866, MIT/W3C, November 1995.

[11] T. Berners-Lee, L. Masinter, M. MxCahill, *Uniform Resource Locators (URL)*, RFC 1738, CERN, Xerox PARC, University of Minnesota, December 1994.

[12] T. Berners-Lee, R. Fielding, H. Frystyk Nielsen, *Hypertext Transfer Protocol-HTTP/1.0*, RFC 1945, MIT/LCS, UC Irvine, May 1996.

[13] R. Fielding, *Relative Uniform Resource Locators*, RFC 1808, June 1995.

[14] Ian S. Graham, *HTML SOURCE BOOK. A Complete Guide to HTML 3.0*, John Wiley & Sons, Inc., 1996 (cap 1_2_6_7_8).

[15] N. Borenstein, N. Freed, *MIME(Multipurpose Internet Mail Extensions) Part One: Mechanisms for Specifying and Describing the Format of Internet Message Bodies*, RFC 1521, Bellcore, Innosoft, September 1993.

[16] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, *Introduzione agli algoritmi*, Jackson Libri, 1998, Vol 1 (cap 5_7), Vol 2 (cap 18_23_25_26).

[17] Frank Harary, *Graph Theory*, Addison-Wesley Pub. Co., 1969.

[18] G. Ausiello, G. Italiano, A. Spaccamela, U. Nanni, *Incremental Algorithms for Minimal Length Paths*, Journal of Algorithms, No.12, 1991, pp. 615-638.

[19] G. Ausiello, G. Italiano, A. Spaccamela, U. Nanni, *On-line Computation of Minimal and Maximal Length Paths*, Theoretical Computer Science, No.95, 1992, pp. 245-261.

[20] Robert E. Tarjan, *Amortized Computational Complexity*, Siam Journal Algebraic Discrete Methods, Vol. 6, No.2, April 1985, pp. 306-318.

[21] Giuseppe F. Italiano, *Finding Paths and Deleting Edges in Directed Acyclic Graphs*, Information Processing Letters, No.28, 1988, pp. 5-11.