Progetto Algorithms and Programming for Massive Data

Alberto Biliotti
Matricola: 7109894
Email: alberto.biliotti@stud.unifi.it

Hardware utilizzato per i test

Per testare il funzionamento del progetto ho usato un computer con 16 gigabyte di RAM, processore Intel i7 di undicesima generazione con frequenza di clock 2.30GHz e come sistema operativo Windows 11 Pro.

Domande scelte

Domanda 1: Which is the oldest venue?
Domanda 2: Compute the connected components and report for each component containing at least 30 publications the most used words in the titles (Possibly excluding conjunctions and adverbs)
Domanda 3: Who is the author who had the largest number of collaborations? If author A and B collaborated twice, this count 2.

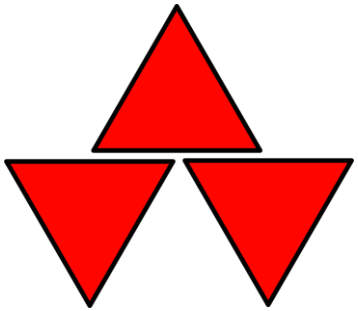
Lettura dei file, creazione e caricamento dei grafi

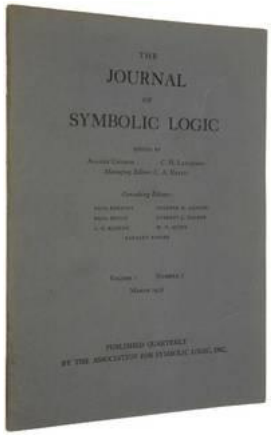
Ho creato un unico metodo, che dati in input il path (relativo o meno) del file csv desiderato e gli indici a cui si trovano le caratteristiche a cui siamo interessati delle pubblicazioni (titolo, autori e venue, visto che l'anno è sempre in prima posizione), legge tale file riga per riga, separando i campi del documento e memorizzando quelli indicati, costruendo poi il grafo desiderato insieme a due dizionari, uno per gli autori ed uno per le pubblicazioni, utili per i passaggi successivi. Nel grafo ho salvato i nodi delle pubblicazioni come coppie titolo-anno, mentre per gli autori ho semplicemente usato i nome.

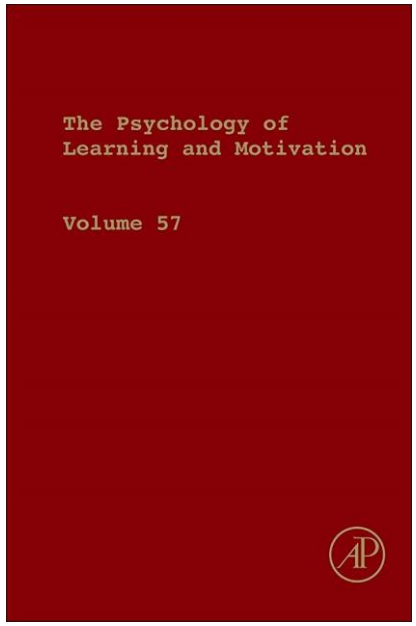
Trovare la venue più «antica»

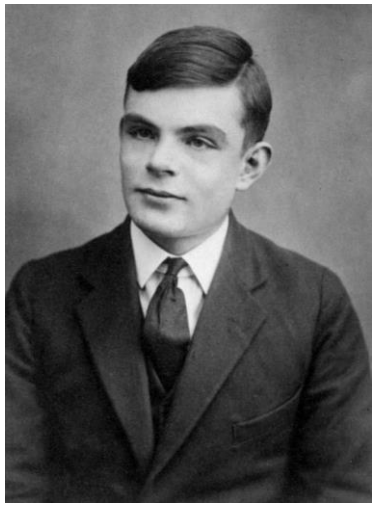
Per trovare la venue più antica non ho avuto bisogno neanche di creare il grafo: ho semplicemente memorizzato la venue della pubblicazione più antica che ho incontrato nel processo di creazione del grafo visto in precedenza, per poi restituirla insieme al grafo ottenuto. In questo modo il procedimento ha costo lineare rispetto alle pubblicazioni, ma tale costo è «incorporato» in quello necessario alla creazione del grafo.
Avrei potuto implementare direttamente una visita BFS per cercare la venue più antica, ma ho preferito evitare perché ho cercato di risparmiare più tempo possibile visto che l'esecuzione è molto lunga.

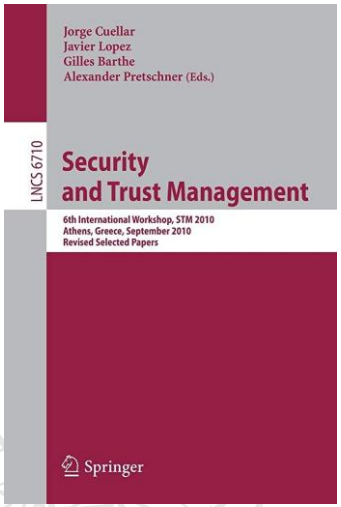
Venue più antiche trovate per tutti i dataset

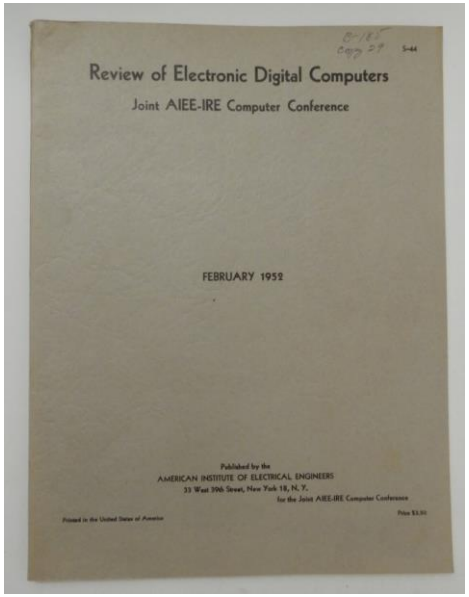












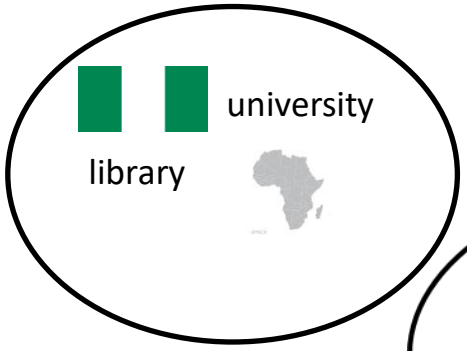
Ricavare le componenti connesse dei grafi

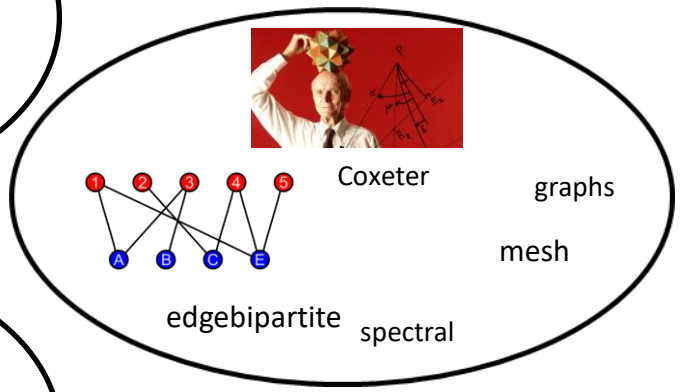
Per ricavare le componenti connesse del grafo è sufficiente eseguire una visita BFS su ciascun nodo non ancora visitato (nel nostro caso, avendo un grafo bipartito in cui ogni pubblicazione possiede almeno un autore posso considerare solo gli autori come nodi di partenza delle visite), per poi salvare ogni nodo raggiunto da tale visita nello stesso insieme, restituendo quindi la componente trovata. Tuttavia, visto che vengono richieste le componenti connesse fino ad un certo anno, se incontro una pubblicazione devo controllare che essa non sia più recente dell'anno specificato (è facile fare ciò considerando che ho definito i nodi delle pubblicazioni come tuple). Se trovo una pubblicazione più recente allora smetto di attraversare tale nodo e lo escludo dall'insieme dei nodi appartenenti alla componente connessa (indicando comunque di averlo già controllato).
Tale procedimento ha costo lineare rispetto al numero di archi, come una visita BFS «normale»

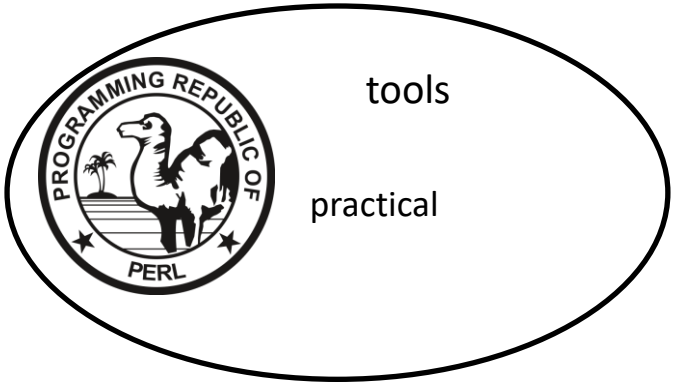
Trovare le parole più frequenti nelle componenti connesse

Una volta ricavate le componenti connesse fino all'anno desiderato, scarto quelle che contengono meno di 30 pubblicazioni. Concateno quindi tutti i titoli delle pubblicazioni appartenenti alla stessa componente connessa in unica stringa, da cui elimino le stopword (articoli, congiunzioni avverbi più comuni) e la punteggiatura, per poi contare, grazie alla classe «Counter», contenuta nella libreria «collections», la frequenza delle parole rimanenti. Infine, per ogni componente connessa, stampo le dieci parole più frequenti tra i titoli delle relative pubblicazioni.

Alcune parole frequenti «curiose» nelle componenti connesse fino al 2023







Trovare gli autori con più collaborazioni

Per fare ciò è stato sufficiente, per ogni autore presente nel dizionario degli autori, sommare il numero di autori nel grafo che sono «vicini» alle pubblicazioni a cui ha preso parte tale autore (escluso l'autore in analisi stesso) .
Per fare ciò ottengo i vicini del nodo associato all'autore, che corrispondono alle pubblicazioni a cui ha preso parte l'autore stesso, per poi andare a sommare in un contatore il numero di vicini di tale pubblicazione sottratto di uno. Restituisco quindi l'autore con il contatore delle collaborazioni massimo.
Tale procedimento ha ordine di complessità nel caso peggiore quadratica, tale situazione si verifica però solo se sto analizzando una cricca, ma per fortuna non è il nostro caso, in quanto le pubblicazioni in media hanno pochi autori.

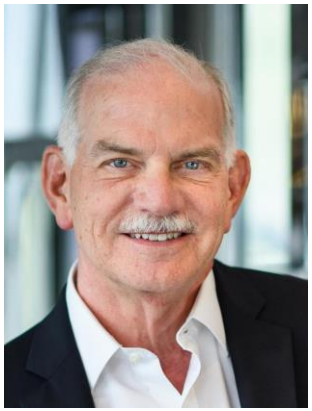
Autori con più collaborazioni fino ad oggi nelle varie tipologie di pubblicazioni

Martin W. Hartmann, tesi phd

Oscar Castillo, incollection

Yohannes Kurniawan, proceeding

Gunter Saake, libri



H. Vincent Poor, articoli

Radu Timofte, inproceeding

Costruzione del grafo unione

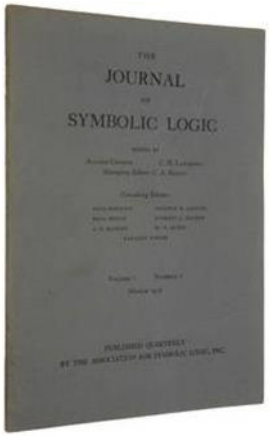
Per costruire il grafo unione sono ricorso ai dizionari degli autori e, partendo dal grafo degli articoli, che è quello più popoloso tra i sette, ho aggiunto gli archi ed i nodi corrispondenti a tutte le pubblicazioni e gli autori non inclusi nel grafo di partenza, fino ad ottenere il grafo unione desiderato.
Successivamente, ho ricavato anche il relativo dizionario degli autori unendo tutti i sette dizionari degli autori per poter riapplicare i metodi visti in precedenza.

Esercizi sul grafo unione

A questo punto posso riapplicare i metodi visti in precedenza sul grafo unione appena costruito. Tuttavia, in questo caso, i metodi impiegheranno molto tempo in più rispetto a prima, in quanto il programma inizia a saturare sempre più la RAM.
Notiamo però che il calcolo della venue più antica è in realtà molto semplice e veloce in quanto basterà trovare la meno recente tra le venue più antiche dei sette dataset visti in precedenza, con costo che risulta quindi essere costante.

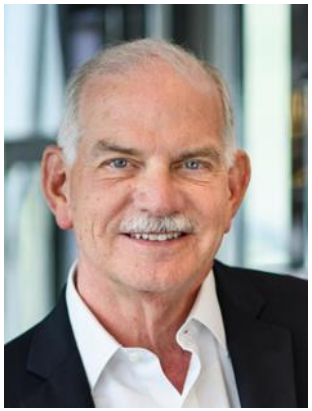
Risultati sul grafo unione (al 2023)

Venue più antica (1936):



Il grafo possiede 80 componenti connesse con almeno 30 pubblicazioni

Autore più collaborativo di sempre (7613 collaborazioni):



H. Vincent Poor

Grafo degli autori

Per rappresentare il grafo contenente come nodi esclusivamente gli autori ho deciso di ricorrere ad un multigrafo, in modo da poter avere la possibilità di poter inserire più di un arco tra due nodi (che in questo caso rappresentano esclusivamente gli autori), in modo da poter considerare anche collaborazioni tra due autori in pubblicazioni diverse.
Per riempire tale multigrafo ho utilizzato il dizionario delle pubblicazioni, aggiungendo al multigrafo un arco per ogni possibile combinazione di due autori che hanno partecipato alla stessa pubblicazione.

Trovare la coppia con più collaborazioni

A questo punto, per trovare la coppia di autori che ha collaborato a più pubblicazioni, è sufficiente, per ogni autore, cercare il vicino con cui condivide più archi, per poi confrontare il massimo numero di archi tra due nodi trovato con il massimo identificato per gli autori precedentemente visitati, restituendo infine tale valore massimo insieme alla coppia di autori associata.
Tale metodo ha complessità quadratica nel caso peggiore, ossia se sto analizzando un grafo clique, ossia in cui ogni nodo è adiacente ad un altro, tuttavia nel nostro caso il multigrafo è ben lontano dall'esserlo, in quanto in media un autore non collabora con un numero eccessivo di altri autori nel corso della propria carriera.

Coppia di autori con più collaborazioni reciproche

Makoto Takizawa

Tomoya Enokido

Con ben 515 collaborazioni reciproche

Conclusioni

L'intero progetto impiega circa 40/45 minuti per terminare con la mia configurazione, tale attesa è dovuta principalmente alla saturazione della RAM (che può essere aggravata anche dell'utilizzo di un IDE) dovuta alla notevole dimensione dei vari grafi e dei dizionari visti in precedenza, piuttosto che al tempo effettivamente necessario alla CPU per eseguire il codice.
Se avessi quindi a disposizione ulteriore RAM il programma terminerebbe ancora prima, ma con la RAM che ho a disposizione Python è costretto a memorizzare parte delle strutture dati analizzate in precedenza sul disco, facendo peggiorare notevolmente le prestazioni.

The End