Introduzione

Le strutture dati e gli algoritmi sono qualcosa di più di semplici concetti astratti. Imparando a padroneggiarli potrete scrivere codice più *efficiente*, ovvero software che "gira" più velocemente e spreca meno memoria. Questo è un grosso problema per le applicazioni software odierne, che devono operare sempre più spesso su piattaforme mobili e gestiscono quantità di dati sempre maggiori.

Il problema della maggior parte delle risorse disponibili su questi argomenti, però, è che sono... ehm... ottuse. La maggior parte dei testi impiega a piene mani il gergo matematico: chi non è un matematico, può faticare a capire i concetti esposti. Anche i libri che affermano di rendere "facili" gli algoritmi sembrano presupporre che il lettore abbia una laurea in matematica avanzata. Per questo motivo, troppi lettori si allontanano da questi concetti ritenendo di non essere all'altezza per capirli.

La verità è che tutto ciò che riguarda le strutture dati e gli algoritmi si basa sul buon senso. La notazione matematica stessa non è che un linguaggio, e tutto, in matematica, può essere spiegato anche con una terminologia basata sul puro buon senso. In questo libro utilizzo il linguaggio del buon senso (e molte, molte immagini) per spiegare questi concetti in modo semplice e, spero, divertente.

Una volta acquisiti questi concetti, sarete in grado di scrivere codice efficiente, veloce ed elegante. Sarete in grado di valutare i pro e i contro delle varie alternative e di prendere decisioni informate su quale sia il codice migliore per una data situazione.

In questo libro ho fatto il possibile per rendere più concreti questi concetti, con idee immediatamente utilizzabili. Certo, leggendo le pagine scoprirete alcune nozioni di computer science davvero interessanti. Ma con questo libro intendo prendere concetti apparentemente astratti e renderli pratici. Quando avrete finito di leggerlo, scriverete sicuramente codice migliore e software più veloce.

A chi è rivolto questo libro?

Credo che questo libro sia ideale per diversi tipi di pubblico.

- Per gli studenti di computer science alla ricerca di un testo che spieghi le strutture dati e gli algoritmi in un linguaggio semplice. Questo libro può servire come supplemento a qualunque testo "classico".

- Per gli sviluppatori alle prime armi, che conoscono i concetti di base della programmazione, ma desiderano apprendere i fondamenti della computer science per scrivere codice migliore ed estendere le proprie conoscenze e competenze di programmazione.
- Per gli sviluppatori autodidatti, che non hanno mai studiato formalmente computer science (o che ormai hanno dimenticato tutto!) e desiderano sfruttare la potenza delle strutture dati e degli algoritmi per scrivere codice più scalabile ed elegante.

Chiunque siate, ho cercato di scrivere questo libro in modo che possa essere letto e apprezzato da persone di tutti i livelli.

Una nota sul codice

Allegri, Pythonisti! Tutto il codice di questo libro è scritto in Python. Mi sforzo di seguire gli standard PEP 8 e di scrivere il codice che funzioni in modo equivalente sia in Python Versione 2 sia in Python Versione 3.
Detto questo, voglio sottolineare che i concetti contenuti in questo libro si applicano a praticamente tutti i linguaggi di programmazione e mi aspetto che questo libro sarà letto anche da persone che non hanno molta familiarità con Python. Per questo motivo, a volte ho evitato alcuni idiomi Python che avrebbero potuto confondere coloro che usano altri linguaggi. È stato un esercizio difficile usare codice Python che risultasse "accessibile" anche ai programmatori non Python, ma spero di aver mantenuto un equilibrio che soddisfi la maggior parte dei lettori.
Praticamente tutto il codice contenuto in questo libro può essere scaricato dal sito di Apogeo all'indirizzo `https://bit.ly/apo-algodati`. Questo repository di codici contiene anche vari test automatizzati. Vi incoraggio a verificarlo poiché i test non compaiono nel libro.
Nei paragrafi di "implementazione" troverete alcuni frammenti di codice più lunghi. Vi incoraggio a studiare questi esempi di codice, ma senza imporvi di comprendere ogni singola riga prima di procedere al paragrafo successivo del libro. Se sentite che questi lunghi pezzi di codice vi stanno bloccando, potete anche semplicemente scorrerli (o anche saltarli).
Infine, devo farvi notare che il codice contenuto in questo libro non è affatto "pronto per la produzione". Il mio obiettivo principale era quello di chiarire il concetto in questione e, sebbene abbia anche provato a rendere completo il codice, non ho tenuto conto di tutti i casi limite. C'è sicuramente spazio per ottimizzarlo, quindi siete liberi di modificarlo a piacere.

Contenuti trattati

Come ormai avrete intuito, questo libro parla di strutture dati e di algoritmi. Più nel dettaglio, il libro è strutturato così.

- Nei Capitoli 1 e 2 spiego che cosa sono le strutture dati e gli algoritmi ed esploro il concetto di complessità temporale, utilizzato per determinare la velocità di un algoritmo. Nel farlo, parlo anche molto di array, set e ricerca binaria.
- Nel Capitolo 3 tratto la notazione Big O e la spiego in termini semplici. Uso questa notazione un po' in tutto il libro, quindi questo capitolo è piuttosto importante.
- Nei Capitoli 4, 5 e 6 approfondisco la notazione Big O e la utilizzo per migliorare la velocità del codice. Nel farlo, tratto vari algoritmi di ordinamento, tra cui Bubble Sort, Selection Sort e Insertion Sort.
- Nel Capitolo 7 applico tutto ciò che ho spiegato in precedenza sulla notazione Big O e analizzo il tema dell'efficienza del codice di tutti i giorni.
- Nei Capitoli 8 e 9 tratto alcune altre strutture dati, tra cui le tabelle hash, gli stack e le code, mostrando come esse possono incidere sulla velocità e sull'eleganza del codice e come possiamo usarle per risolvere problemi concreti.
- Il Capitolo 10 introduce la ricorsione, un concetto fondamentale nel mondo della computer science. La analizzo per scoprire come possa essere un ottimo strumento per risolvere determinate situazioni. Il Capitolo 11 insegna a usare la ricorsione, un argomento non facile, perché può creare un po' di confusione.
- Il Capitolo 12 mostra come ottimizzare il codice ricorsivo e impedire che vada fuori controllo. Il Capitolo 13 mostra come utilizzare la ricorsione come strumento di base per algoritmi estremamente veloci come Quicksort e Quickselect, per portare a un livello superiore le vostre competenze nel campo dello sviluppo di algoritmi.
- I Capitoli da 14 a 18 esplorano alcune strutture dati basate sui nodi, tra cui la lista concatenata, l'albero binario, lo heap, il trie e i grafi e mostrano come ciascuno di essi possa essere la soluzione ideale per determinate applicazioni.
- Il Capitolo 19 esplora i problemi legati alla complessità spaziale, che è importante quando si programma per dispositivi con quantità di spazio di memoria di massa relativamente piccole o quando si ha a che fare con i big data.
- Il Capitolo 20 esplora varie tecniche pratiche per ottimizzare l'efficienza del codice e vi offre nuove idee per migliorare il codice.
- Un'appendice con le soluzioni a tutti gli esercizi è disponibile online all'indirizzo `https://bit.ly/apo-algodati`.

Come leggere questo libro

Vi consiglio di leggere questo libro in ordine. Alcuni libri possono essere letti un po' a piacere, saltando i capitoli, *ma questo libro non è così*. Ogni capitolo presuppone che abbiate letto i capitoli precedenti e il libro è strutturato con cura in modo che possiate estendere le vostre conoscenze man mano che procedete.
Detto questo, ci sono alcuni capitoli nell'ultima metà del libro che non dipendono del tutto l'uno dall'altro. Lo schema seguente illustra con precisione quali capitoli sono prerequisiti per altri capitoli.

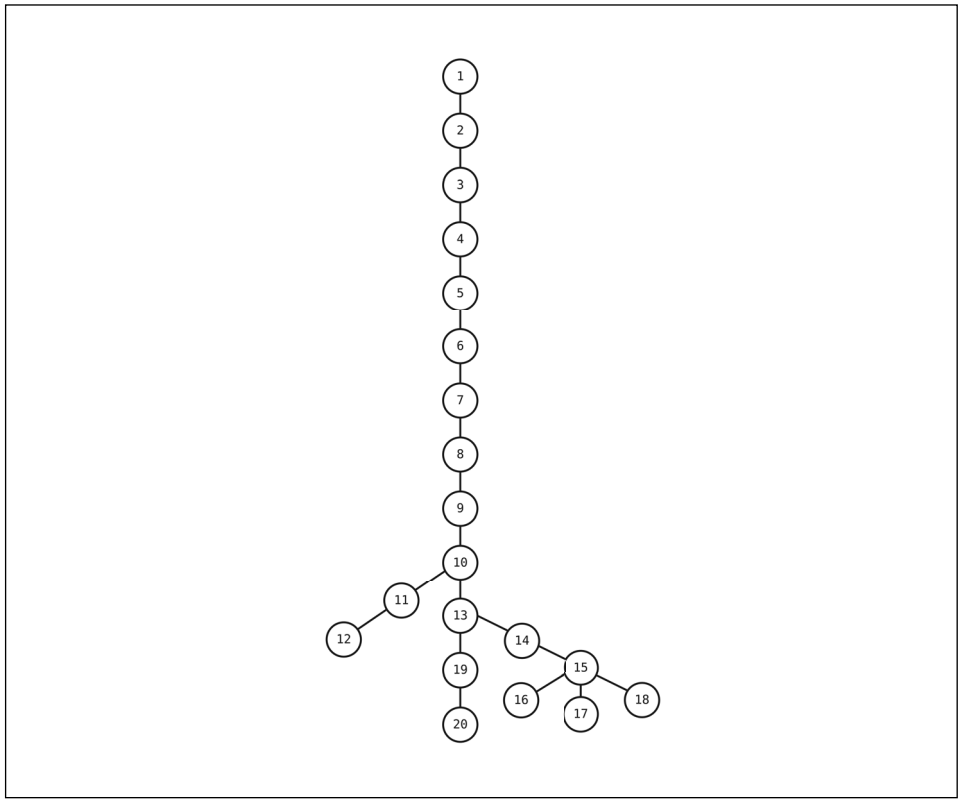

Figura I.1

Per esempio, teoricamente potreste saltare dal Capitolo 10 al Capitolo 13, se volete (a proposito, questo schema si basa su una struttura dati chiamata albero, di cui vi parlerò nel Capitolo 15).

Un'altra nota importante: per rendere massimamente accessibile questo libro, non sempre rivelo tutto di un determinato concetto quando lo presento per la prima volta. A volte, il modo migliore per affrontare un concetto complesso consiste nel rivelarne solo una piccola parte e riservare gli aspetti più complessi a un secondo tempo, quando le basi si possono dare per acquisite. Se in un certo punto definisco un termine particolare in un certo modo, non prendete tale definizione come quella di un libro di testo, finché non avrete completato l'intera parte dedicata a quell'argomento.

Spiegare richiede di scendere a compromessi: per rendere il libro "digeribile", ho scelto di semplificare molto alcuni concetti all'inizio e di chiarirli nel tempo, piuttosto che garantire che ogni frase sia sempre completamente e accademicamente accurata. Ma non preoccupatevi troppo: alla fine vedrete l'immagine intera e accurata.

Ringraziamenti

Anche se scrivere un libro può sembrare un compito solitario, questo libro non esisterebbe senza le molte persone che mi hanno supportato in questo mio viaggio. Vorrei ringraziare personalmente ognuno di voi.

Alla mia meravigliosa moglie, Rena: grazie per il tempo e il sostegno emotivo che mi hai dedicato. Ti sei occupata di tutto mentre io mi rintanavo come un recluso e scrivevo.

Ai miei adorabili figli, Tuvi, Leah, Shaya, Rami e Yechiel: grazie per la pazienza che avete dimostrato mentre scrivevo il mio libro sugli "agorimmi". E… sì, finalmente l'ho finito.

Ai miei genitori, Howard e Debbie Wengrow: grazie per aver suscitato il mio interesse per la programmazione e per avermi incoraggiato a perseguirlo. Non sapevate che procurarmi un libro introduttivo alla computer science per il mio nono compleanno avrebbe gettato le basi per la mia carriera e ora per questo libro.

Ai genitori di mia moglie, Paul e Kreindel Pinkus: grazie per il vostro continuo sostegno, a me e alla mia famiglia. La vostra saggezza e il vostro calore hanno significato tanto per me.

Quando ho inviato per la prima volta il mio manoscritto a Pragmatic Bookshelf, ho pensato che fosse "buono". Tuttavia, grazie all'esperienza, ai suggerimenti e alle richieste di tutte le meravigliose persone che lavorano lì, il libro è diventato qualcosa di molto, molto migliore di quanto avrei potuto scrivere da solo.

In primo luogo, vorrei ringraziare la mia redattrice, Katharine Dvorak, per aver contribuito a portare rapidamente al traguardo questa edizione, migliorando allo stesso tempo la qualità del libro.

Vorrei anche ringraziare Tammy Coron, caporedattore di Pragmatic Bookshelf, per la supervisione di questo progetto(e grazie per aver generato tutte le immagini specifiche di Python).

Grazie a Dave Rankin, CEO di Pragmatic Bookshelf, per aver guidato la migliore casa editrice per cui potessi scrivere e per aver supportato e condiviso la mia visione: creare più edizioni di questo libro, specifiche per i vari linguaggi.

A Brian MacDonald, l'editor delle prime edizioni, grazie per avermi mostrato come dovrebbe essere scritto un libro. Questo libro ha la tua impronta dappertutto.

Alla talentuosa sviluppatrice e artista Colleen McGuckin, grazie per aver preso i miei brutti schizzi e averli trasformati in bellissime immagini digitali. Questo libro non sarebbe nulla senza le spettacolari immagini che hai creato con tanta abilità e attenzione ai dettagli.

Ho avuto la fortuna di avere le recensioni di molti esperti. Il vostro feedback è stato estremamente utile e ha fatto sì che questo libro fosse il più accurato possibile. Vorrei ringraziare tutti voi per il vostro contributo.

I revisori delle prime edizioni sono stati: Alessandro Bahgat, Ivo Balbaert, Rinaldo Bonazzo, Alberto Boschetti, Mike Browne, Craig Castelaz, Jacob Chae, Javier Collado, Zulfikar Dharmawan, Ashish Dixit, Dan Dybas, Emily Ekhdal, Mohamed Fouad, Derek Graham, Neil Hainer, Peter Hampton, Rod Hilton, Jeff Holland, Jessica Janiuk, Aaron Kalair, Stephan Kämper, Grant Kazan, Arun S. Kumar, Sean Lindsay, Nigel Lowry, Joy McCaffrey, Dary Merckens, Nouran Mhmoud, Kevin Mitchell, Daivid Morgan, Brent Morris, Jasdeep Narang, Emanuele Origgi, Stephen Orr, Kenneth Parekh, Jason Pike, Sam Rose, Ayon Roy, Frank Ruiz, Brian Schau, Tibor Simic, Matteo Vaccari, Mitchell Volk, Stephen Wolff e Peter W. A. Wood.

I revisori di questa edizione Python sono stati: Connor Baskin, Katy Douglas, Tzvi Friedman, Joey Gallotta, Rod Hilton, Paa JAKE, Patrick Nikolaus, Nathan Pena, Terry Peppers, Lyndon Purcell, Cody Rutt, Brian Schau, Ahmad Shahba e Joe Yakich.
Oltre ai revisori ufficiali, vorrei ringraziare anche tutti i lettori della versione beta del libro, che hanno fornito i loro feedback mentre continuavo a scrivere e modificare il libro. I vostri suggerimenti e commenti e anche le vostre domande sono stati preziosi.
Vorrei ringraziare anche tutto lo staff, gli studenti e gli ex studenti di Actualize per il loro supporto. Originariamente, questo libro era un progetto Actualize e tutti voi vi avete contribuito in vari modi. Vorrei ringraziare in particolare Luke Evans per avermi dato l'idea di scrivere questo libro.
Grazie a tutti per aver reso questo libro una realtà.