Computer, pensiero computazionale e strutture dati

Autori
Silvio Peroni – silvio.peroni@unibo.it
Dipartimento di Filologia Classica e Italianistica, Università di Bologna, Bologna, Italia

Aldo Gangemi – aldo.gangemi@unibo.it
Dipartimento di Filologia Classica e Italianistica, Università di Bologna, Bologna, Italia

Matteo Pascoli - matteo.pascoli2@unibo.it
Dipartimento di Filologia Classica e Italianistica, Università di Bologna, Bologna, Italia

Sommario

In questo capitolo verrà fornita una definizione generale di *computer*, per poi discutere le caratteristiche principali che contraddistinguono il pensiero computazionale, presentando come le informazioni presenti nel quotidiano, che noi utilizziamo abitualmente, possano essere descritte in forma astratta mediante l'uso di opportune strutture.

Che cos'è un computer?

Il termine *computer* (*calcolatore*, in italiano) è in uso, oggi, per indicare una "macchina per l'elaborazione di dati rappresentati da caratteri alfanumerici variamente codificati, che vengono sottoposti a procedimenti aritmetici e logici, memorizzati in archivi e resi reperibili e trasmissibili" – dal dizionario di Repubblica.it. Tuttavia, la definizione originale dello stesso termine, in uso dal diciassettesimo secolo, è leggermente differente, visto che si riferisce a qualcuno che "esegue calcoli matematici" – da Wikipedia. In questo capitolo, quando useremo il termine "computer", considereremo sempre la sua accezione più generica, ovvero: qualsiasi agente (ovvero,

quell'entità in grado di agire se istruita appropriatamente, come una persona o una macchina) che sia in grado di fare calcoli e produrre una risposta (detta *output*) a partire da qualche informazione iniziale (detta *input*).

Computer *umani*, ovvero gruppi di persone che hanno eseguito lunghi calcoli per determinati esperimenti, sono stati impiegati molte volte nel passato. Per esempio, in astronomia, computer umani sono stati impiegati per calcolare le coordinate astronomiche di oggetti extraterrestri – come i calcoli effettuati da Alexis Claude Clairaut e colleghi per comprendere i vari passaggi della cometa di Halley. Come ulteriore esempio, computer umani sono stati usati anche da Napoleone Bonaparte quanto questo ha imposto la creazione di tabelle matematiche per convertire i valori descritti con il vecchio sistema di misura imperiale verso il nuovo sistema metrico (tutt'ora in uso) [Campbell-Kelly, 2009] [Roegel, 2010].

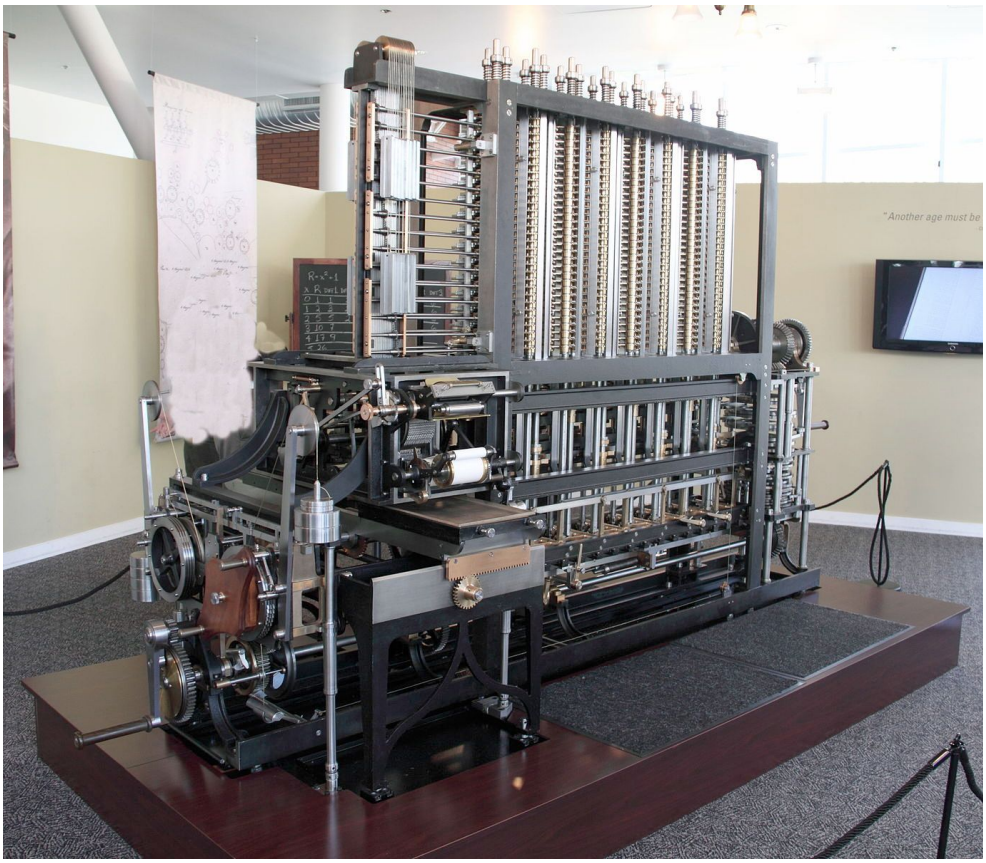

Figure 1. Macchina Differenziale Numero 2, costruita allo Science Museum di Londra e esposta al Computer History Museum di Mountain View (California).
Foto di Allan J. Cronin, sorgente: https://commons.wikimedia.org/wiki/File:Difference_engine.JPG.

Nel 1822, Charles Babbage, capendo la complessità di eseguire tutti questi calcoli a mano evitando l'introduzione di errori, iniziò lo sviluppo di un nuova, incredibile, macchina, chiamata Macchina Differenziale, mostrata in Figura 1. L'idea era quella di avere a disposizione una macchina che potesse gestire operazioni simili a quelle effettuate dai computer umani, ma in

modo che fossero eseguite automaticamente, velocemente, e senza errori. Babbage fu in grado di costruire solo un prototipo parziale della macchina e, dopo l'entusiasmo iniziale, fu demoralizzato dalla limitata flessibilità che offriva. Infatti, la Macchina Differenziale non era programmabile e, di conseguenza, era in grado di utilizzare solo un numero limitato di operazioni sull'input ricevuto – specificato, fisicamente, cambiando specifiche configurazioni della macchina.

In modo da sopperire a queste limitazioni, nel 1837, Babbage iniziò a progettare una nuova macchina, la Macchina Analitica, mostrata in Figura 2. Seppur nessun prototipo di questa macchina sia stato effettivamente costruito da Babbage, in linea di principio avrebbe dovuto permettere la creazione di qualunque calcolo procedurale, rendendola il primo computer meccanico e *general-purpose* della storia. Contrariamente al suo predecessore, la Macchina Analitica era in grado di ricevere in input istruzioni e dati mediante l'uso di schede perforate, senza obbligare l'utilizzatore, quindi, a compiere manipolazioni fisiche della macchina stessa per farla funzionare.

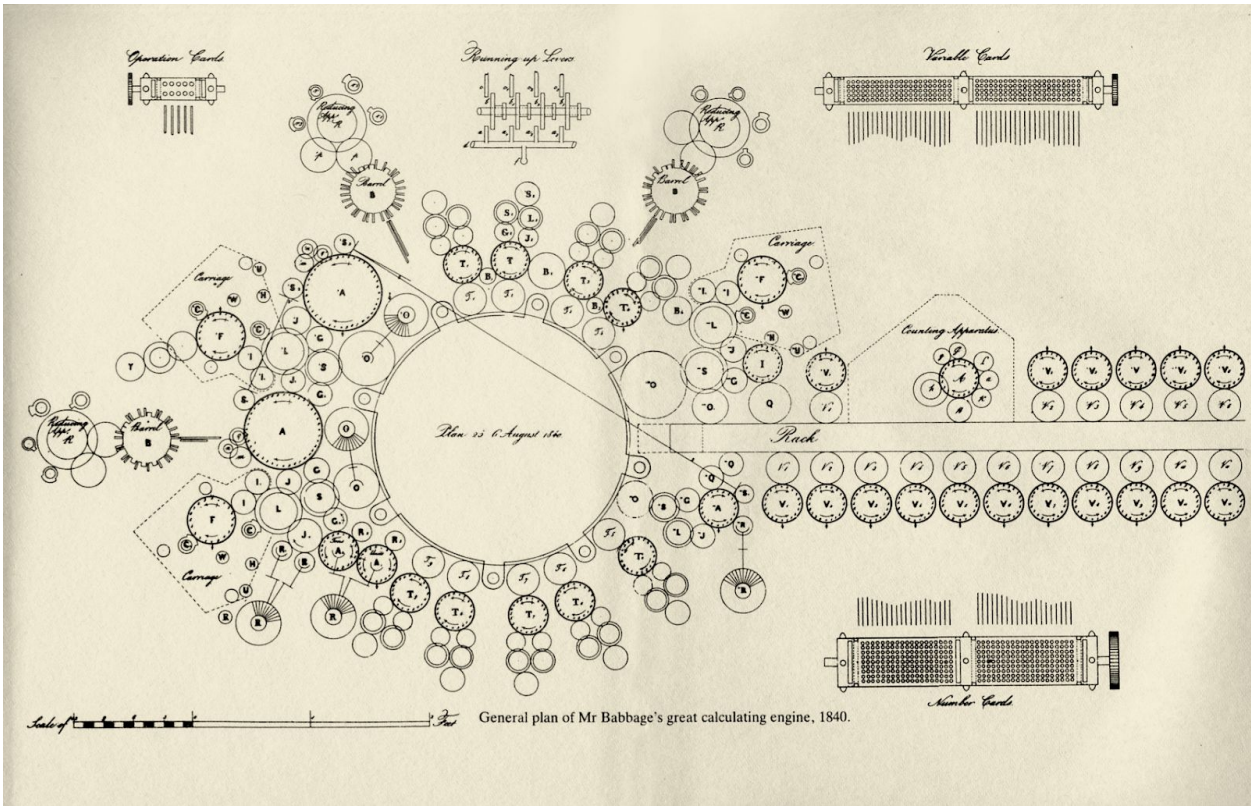

Figura 2. Una bozza di Babbage che descrive l'architettura principale della Macchina Analitica.
Sorgente: The Analytical Engine: 28 Plans and Counting, Computer History Museum.

C'è voluto più di un secolo per vedere sviluppate in una macchina fisica le idee presentate nella Macchina Analitica. Infatti, l'evoluzione della tecnologia computazionale ha avuto una brusca accelerata soltanto a seguito della Seconda Guerra Mondiale. In quei tempi, molti calcolatori

furono costruiti per ragioni militari, come, ad esempio, la Bomba (1940) (Figura 3), sviluppata da Alan Turing, che è stata il principale strumento che ha permesso a un gruppo di persone, rinchiuse nella base militare segreta britannica di Bletchley Park, di decifrare (grazie anche al lavoro pregresso fatto da crittologi polacchi come Marian Rejewski) le comunicazioni tedesche che erano state cifrate dalla macchina Enigma.

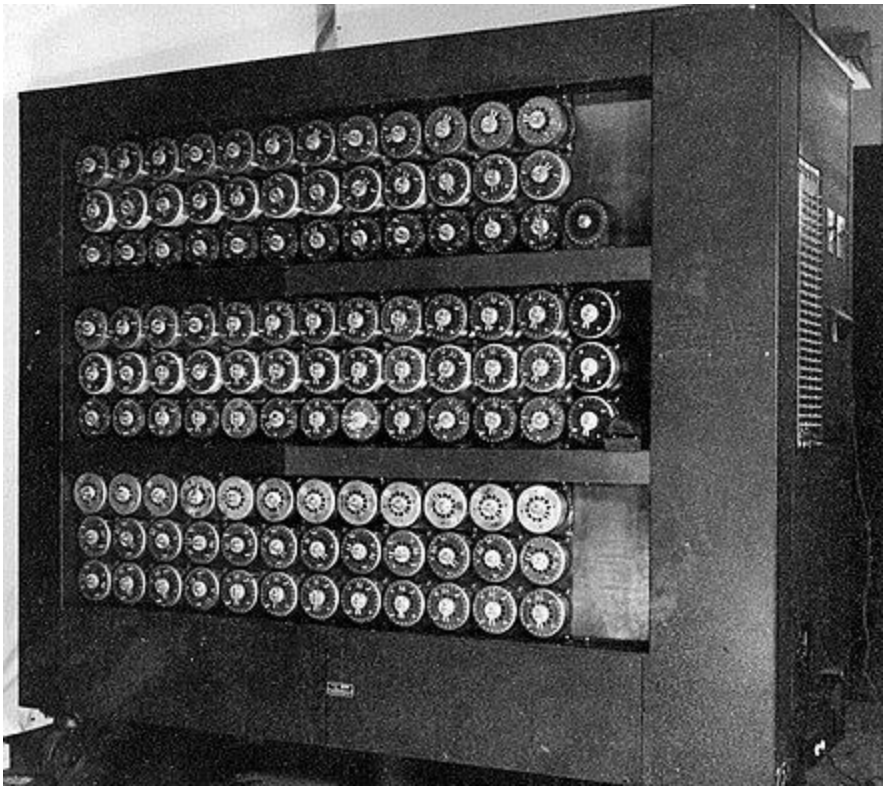

Figura 3. Una Bomba di Bletchley Park.
Sorgente: https://en.wikipedia.org/wiki/File:Wartime_picture_of_a_Bletchley_Park_Bombe.jpg .

Mentre la Bomba era una macchina estremamente efficace ed efficiente, era altresì parzialmente basata su componenti prettamente meccanici, e permetteva lo svolgimento di una sola operazione, anche se estremamente cruciale da un punto di vista squisitamente storico. Il primo computer interamente digitale, come pensato da Babbage con la sua Macchina Analitica, è stato sviluppato negli Stati Uniti d'America soltanto qualche anno dopo, nel 1946: l'Electronic Numerical Integrator and Computer (ENIAC), mostrato in Figura 4, che era completamente programmabile attraverso l'uso di cavi e interruttori. Questa invenzione è stata una delle più cruciali pietre miliari della storia dei computer elettronici, rappresentando una sorta di "punto fisso" nel tempo da cui tutti i moderni computer sono poi stati creati.

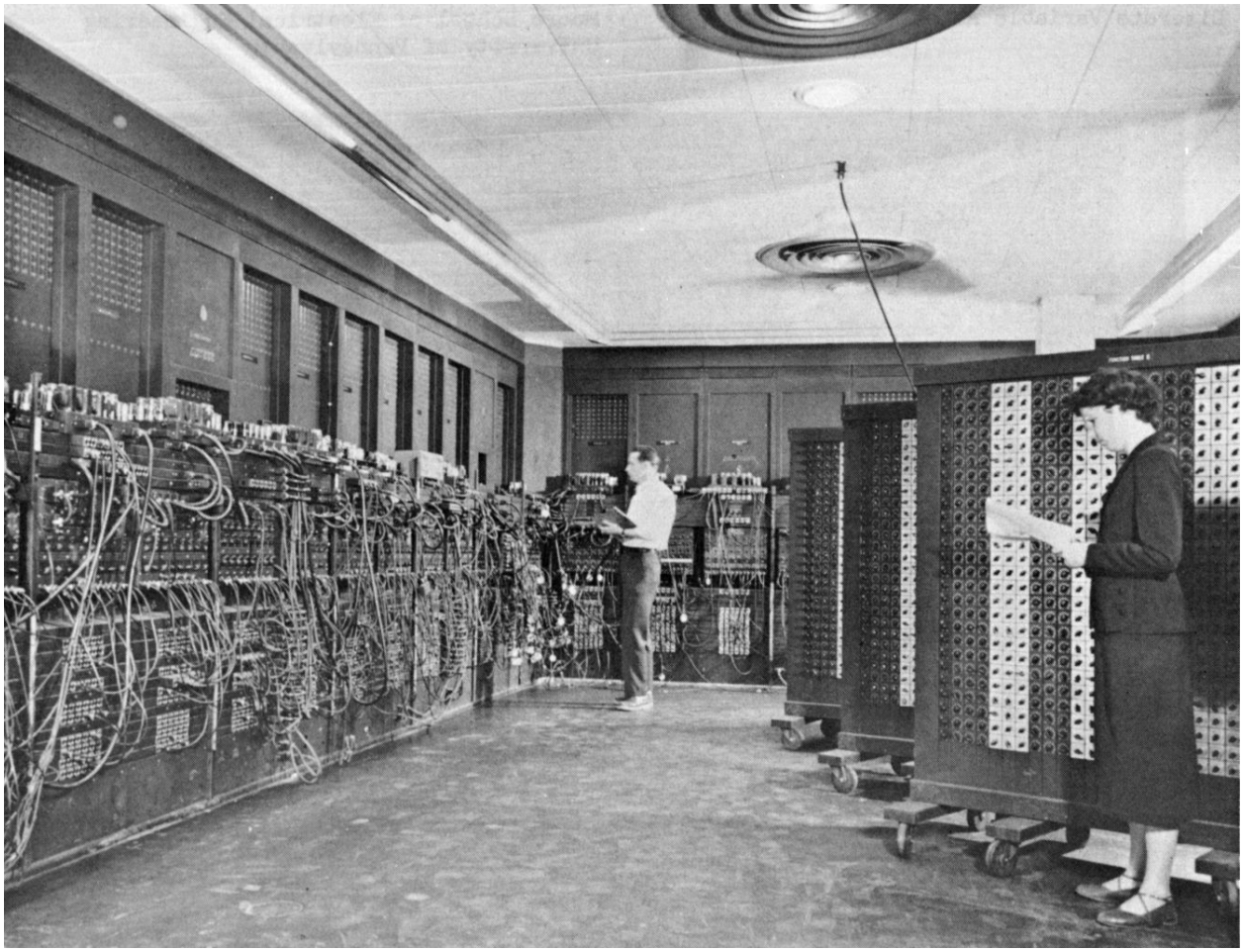

Figura 4. Una fotografia dell'ENIAC installato nel Ballistic Research Laboratory (Maryland).
Sorgente: https://en.wikipedia.org/wiki/ENIAC#/media/File:Eniac.jpg.

Pensiero computazionale

Spesso diciamo di *programmare* un computer – dove la parola "computer", in questo caso, si riferisce a un computer elettronico. Tuttavia, come anticipato nella definizione che abbiamo fornito in questo capitolo, un computer può essere sia una macchina sia un essere umano. Tuttavia, il verbo *programmare* non si presta particolarmente bene ad essere usato con computer umani – visto che non possiamo letteralmente programmare una persona (anche se qualche pseudoscienza, come la programmazione neuro-linguistica, sostiene di poterlo fare). Casomai diciamo che *parliamo con* una persona per *istruirla* sull'esecuzione di specifiche azioni attraverso l'uso di un particolare linguaggio (in questo caso naturale) che viene usato come canale di comunicazione. Di conseguenza, in questo contesto, si dovrebbero usare gli stessi verbi, ovvero *parlare* e *istruire*, anche quando ci si riferisce ad un computer elettronico perché, di fatto, è quello che succede. In pratica, scrivere un programma è esattamente questo: comunicare ad un computer elettronico utilizzando un linguaggio (in questo caso formale) che sia l'istruttore umano sia il computer stesso possano comprendere [Papert, 1980].

Una volta d'accordo su quale linguaggio usare per la comunicazione tra l'istruttore e il computer (a prescindere dal fatto che questo sia un umano o una macchina), dovremmo iniziare a pensare una sequenza di possibili istruzioni da comunicare che, se seguite sistematicamente, possano restituire un risultato atteso per risolvere un certo problema. In modo da raggiungere questo obiettivo, solitamente (e inconsapevolmente) proviamo a ricercare possibili soluzioni per il problema in questione confrontandolo con possibili situazioni che si sono già presentate in passato – e che, plausibilmente, abbiamo già risolto. L'idea è quella di trovare un *pattern* (schema ricorrente) che fornisca una possibile soluzione a un insieme di situazioni che, a livello astratto, sono del tutto omogenee, in modo da poter riusare la stessa strategia per raggiungere il nostro obiettivo, se questa è stata soddisfacente nel passato. Per esempio, alcune delle azioni che eseguiamo in un ufficio postale sono abbastanza simili a quelle che eravamo abituati a svolgere quando, da piccoli, aspettavamo il nostro turno per giocare con lo scivolo in un parco, come mostrato in Figura 5.

Figura 5. Due fotografie che descrivono una situazione simile, ovvero fare la coda, in due contesti differenti: un parco (sinistra) e un ufficio postale (destra).
La foto di sinistra è di Prateek Rungta, sorgente: https://www.flickr.com/photos/rungta/4409560365/. La foto di destra è di Rain Rabbit, sourgente: https://www.flickr.com/photos/37996583811@N01/6158491035/.

Considerando le situazioni e contesti appena riportati, possiamo definire il *pensiero computazionale* come un approccio per risolvere problemi, sviluppare sistemi e capire il comportamento umano che riprende i concetti fondamentali della computazione [Wing, 2008] – dove con la parola *computazione* si intende *calcolo*. Il pensiero computazionale definisce i processi mentali che coinvolgiamo quando formuliamo un certo problema ed esprimiamo le relative soluzioni usando un linguaggio che un computer (sia esso umano o macchina) può comprendere e, conseguentemente, eseguire.

Jeannette Wing fornisce una definizione aggiuntiva, per chiarire ancor meglio cosa si intenda per pensiero computazionale [Wing, 2008]:

> Computational thinking is a kind of analytical thinking. It shares with mathematical thinking in the general ways in which we might approach solving a problem. It shares with engineering thinking in the general ways in which we might approach designing and evaluating a large, complex system that operates within the constraints of the real world. It shares with scientific thinking in the general ways in which we might approach understanding computability, intelligence, the mind and human behaviour.[1]

La nozione principale dietro al pensiero computazionale è l'*astrazione*, ovvero l'abilità di esercitare pensiero astratto e di esibire abilità di astrazione – che comporta il processo di rimozione dei dettagli trascurabili di una situazione in modo da semplificarla, per così focalizzare l'attenzione sulle sue caratteristiche principali [Kremer, 2007]. Come già anticipato nell'esempio descritto in Figura 5, l'abilità di astrarre situazioni e nozioni è cruciale per automatizzare l'esecuzione di determinate operazioni attraverso l'uso di un computer che è responsabile dell'interpretazione di queste astrazioni.

Di solito, noi usiamo queste astrazioni in modo intenzionale o inconscio nella vita quotidiana, come quando usiamo un contenitore sia per metterci degli oggetti (per esempio un carrello della spesa), sia quando lo immaginiamo per metterci delle azioni da svolgere (per esempio una lista scritta o mentale delle cose da fare prima di pranzo), o addirittura quando vediamo cose dove non ci sono, come nel caso della pareidolia (Figura 6), perché alcune caratteristiche percettive hanno una forma simile a ciò che crediamo di, o vogliamo, vedere.

Figura 6. Due esempi di pareidolia: la prima è una fotografia del Monte Circeo, che da questa prospettiva evoca in alcuni la Maga Circe dormiente, la seconda è una fusione di muffin e facce di cani, creata automaticamente da una rete neurale, DeepDream.
Sorgenti: https://it.wikipedia.org/wiki/File:Circeo.JPG (foto di sinistra), https://it.wikipedia.org/wiki/File:Southwark_market_pies_(trippy).jpeg (foto di destra).

[1] Traduzione libera in italiano: "Il pensiero computazionale è un tipo di pensiero analitico. Condivide, con il pensiero matematico, il modo in cui possiamo approcciare un problema per trovarne la soluzione. Condivide, con il pensiero ingegneristico, il modo in cui possiamo affrontare la progettazione e la valutazione di grandi sistemi complessi che operano all'interno dei limiti del mondo reale. Infine, condivide, con il pensiero scientifico, il modo in cui possiamo trattare la comprensione della calcolabilità, l'intelligenza, la mente e il comportamento umano."

Uno degli obiettivi del pensiero computazionale è quello di *dare nuovamente forma* alle astrazioni che abbiamo già immagazzinato in passato come conseguenza della nostra esperienza personale – e che, spesso, riutilizziamo inconsciamente. Quindi, per essere nuovamente e interamente coscienti di queste astrazioni, dobbiamo ridefinirle usando un linguaggio appropriato per renderle comprensibili a un computer.

Questo richiede a volte una notevole capacità di introspezione e di riflessione sulle caratteristiche più importanti di un oggetto, di un discorso o di una situazione, non troppo diversamente da ciò che avviene nel pensiero filosofico o semiotico, ma sempre garantendo *precisione* e *adeguatezza* alla risoluzione di problemi e alla rappresentazione del loro contenuto. Questo indipendentemente dal fatto che vogliamo risolvere il problema di come elaborare un testo digitale, di come organizzare i dati relativi a un'indagine archeologica, o di come trovare il modo migliore per raccomandare un prodotto o un servizio.

A livello generale, l'obiettivo principale (dell'insegnamento) del pensiero computazionale è quello di permettere alle persone di pensare come se fossero *computer scientist*, anche quando bisogna affrontare attività del quotidiano. In futuro, il pensiero computazionale sarà parte integrante dell'educazione primaria delle persone [Wing, 2008], come la matematica e la fisica, e plasmerà il modo in cui le persone pensano e imparano [Papert, 1980] e di conseguenza, vivono.

In modo indiretto, questo sta già avvenendo attraverso la trasformazione dei sistemi di produzione e comunicazione: il Web e i social media, gli "orchestratori di reti" (come AirBnB e Uber), etc. Categorie fondamentali dell'esistenza umana: spazio, tempo, identità, immagine pubblica, collettività, pianificazione, relazioni interpersonali, etc. sono tutte influenzate dalla trasformazione socio-tecnica in maniera più veloce e profonda di quanto sia avvenuto negli ultimi secoli.

Strutture dati

Uno dei processi di base dell'attività di astrazione è quella di descrivere l'informazione presente in una certa situazione secondo un'organizzazione generica e riutilizzabile in più contesti. Per fare ciò, si usano quelle che comunemente sono chiamate strutture dati. Le strutture dati sono i modi in cui possiamo organizzare l'informazione e i dati da essere processati (input) e restituiti (output) da un computer, in modo da potervi accedere in modo efficiente ed efficace a livello computazionale. In pratica, una struttura dati è una sorta di contenitore dove possiamo posizionare alcune informazioni, e che fornisce dei metodi specifici per aggiungere e richiedere pezzi di questa informazione.

Tra le più semplici strutture dati abbiamo: le liste, le code, le pile, gli insiemi, i dizionari, gli alberi e i grafi. Tutte queste strutture verranno analizzate nelle sottosezioni che seguono, fornendo esempi delle loro applicazioni in scenari quotidiani.

Lista

Una lista è una sequenza di elementi *ordinati* e *ripetibili* che si possono *contare*, perché si può sapere quanti elementi essa contiene in un dato momento. I suoi elementi sono ordinati perché sono posizionati in uno specifico ordine di precedenza tra loro, che è preservato anche quando aggiungiamo e rimuoviamo determinati elementi. Inoltre, gli elementi in una lista sono ripetibili, visto che possono comparire più di una volta in una lista – ad esempio, se proviamo a creare una lista di caratteri della parola “pensiero”, la lettera “e” comparirà due volte, una in seconda posizione, e un’altra in sesta posizione.

Esistono diversi esempi tratti da situazioni reali di queste liste astratte. Per esempio, in Figura 7, sono mostrati un indice di un libro e una lista di riferimenti bibliografici in un articolo scientifico. Entrambi sono due oggetti concreti costruiti partendo dalla nozione astratta di lista.

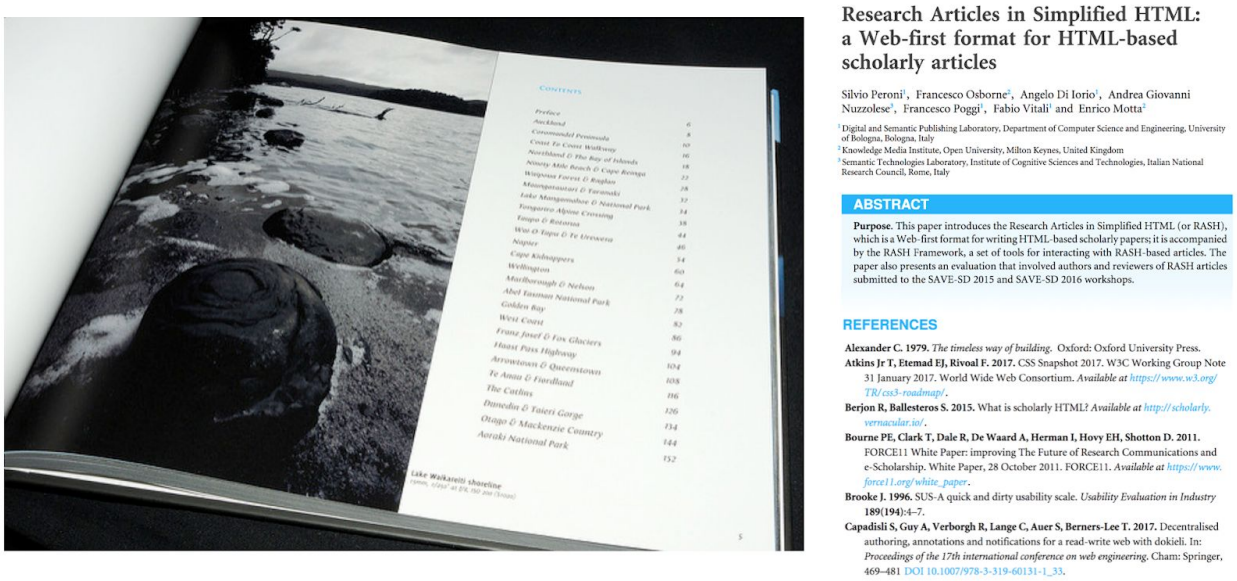

Figura 7. Due esempi di una lista espressa in oggetti reali: un indice di un libro (sinistra), e la lista di riferimenti bibliografici di un articolo di ricerca (destra).
Foto di sinistra di Marcus Holland-Moritz, sorgente: https://www.flickr.com/photos/mhx/4347706564/. Schermata di destra, sorgente: https://doi.org/10.7717/peerj-cs.132.

Pila

Una pila è una specie di lista vista da un particolare punto di vista, ovvero dal basso verso l’alto, e con uno specifico insieme di operazioni che si possono effettuare sugli elementi della pila.

Figura 8 mostra due esempi di pile in oggetti di tutti i giorni. In particolare, abbiamo una pila di sedie (a sinistra) e una pila di libri (a destra).

La caratteristica principale degli elementi di questa struttura riguarda le operazioni di aggiunta e rimozione, che seguono una strategia *last in first out* (*LIFO*) – ovvero, l'ultimo elemento che viene inserito è il primo ad essere rimosso. Infatti, l'ultimo elemento inserito nella struttura è posizionato in cima alla pila e, quindi, è anche il primo che verrà rimosso se richiesto. Inoltre, se si vuole rimuovere un elemento in mezzo alla pila, è prima necessario rimuovere tutti gli elementi che sono stati posizionati *dopo* questo, dal più recente al più vecchio.

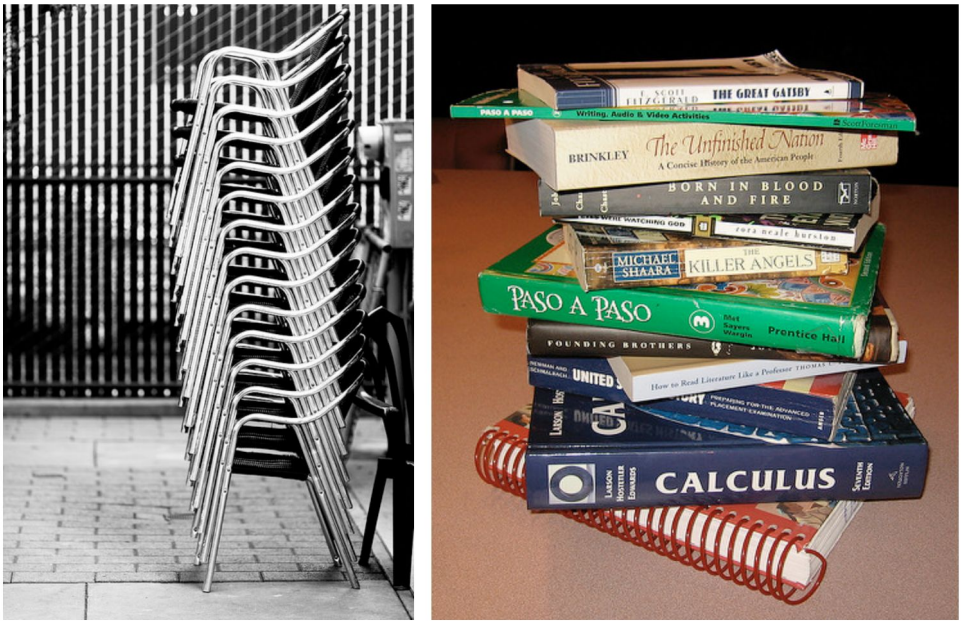

Figure 8. Due esempi di una pila espressa in oggetti reali: una pila di sedie (sinistra), e una pila di libri (destra).

Foto di sinistra di Jeremy Brooks, sorgente: https://www.flickr.com/photos/jeremybrooks/16410797960/. Foto di destra di Cary Lee, sorgente: https://www.flickr.com/photos/the1andonlycary/3310345438/.

Coda

Una coda è una specie di lista vista da un'altra prospettiva, ovvero da sinistra verso destra, e con uno specifico insieme di operazioni che possono essere effettuate sugli elementi che contiene. Figura 9 mostra due differenti esempi di code in situazioni del quotidiano: una coda di bambini (sinistra) e una linea di attesa di taxi (destra).

La caratteristica principale degli elementi di questa struttura riguarda le operazioni di aggiunta e rimozione degli elementi, che seguono una strategia *first in first out* strategy (*FIFO*) – ovvero, il primo elemento che viene aggiunto è anche il primo che viene rimosso a seguito di una richiesta. In pratica, il primo elemento inserito nella struttura è posizionato nella parte libera più a sinistra della coda e, di conseguenza, è anche il primo che verrà rimosso quando richiesto. In

modo del tutto simile alle pile, anche nelle code, se si vuole rimuovere un certo elemento in mezzo, è necessario prima rimuovere tutti gli elementi che sono stati aggiunti prima di esso.

Figura 9. Due esempi di una coda in oggetti del quotidiano: una coda di bambini che aspettano il loro turno per giocare con lo scivolo (sinistra), e una linea di sosta per taxi (destra).
Foto di sinistra di Prateek Rungta, sorgente: https://www.flickr.com/photos/rungta/4409560365/. Foto di destra di Lynda Bullock, sorgente: https://www.flickr.com/photos/just1snap/5141019486/.

Insieme

Un insieme è una collezione di elementi *non ordinati* e *non ripetibili* che si possono *contare*. I suoi elementi non sono ordinati perché l'ordine di inserimento di questi non prescrive nessuna relazione di cardinalità tra loro. Inoltre, sono non ripetibili perché lo stesso valore non può essere incluso due o più volte.

Figure 10. Due esempi di un insieme di oggetti reali: una classe di studenti (sinistra), e una collezione di colori in un bicchiere di plastica (destra).
Foto di sinistra di Uri Tours, sorgente: https://www.flickr.com/photos/northkoreatravel/10682515504/. Foto di destra di Mikel Seijas Alonso, sorgente: https://www.flickr.com/photos/xumet/2670267503/.

Ovviamente, esistono diversi esempi di questa struttura dati in oggetti e situazioni di vita quotidiana. Per esempio, Figura 10 mostra una classe di studenti e una collezione di colori. Entrambe sono oggetti concreti costruiti a partire dalla nozione astratta di insieme.

Dizionario

Un dizionario (o *array* associativo) è una collezione *non ordinata* di elementi definiti da *coppie chiave-valore* che si possono *contare*, dove la chiave *non è ripetibile*. I suoi elementi sono non ordinati perché l'ordine di inserimento non prescrive nessuna relazione di cardinalità tra gli elementi, in modo del tutto simile agli insiemi. Inoltre le chiavi di ogni coppia non sono ripetibili perché la stessa chiave non può essere usata due o più volte nel dizionario.

Ovviamente, esistono diversi esempi di questi dizionari astratti in oggetti di uso quotidiano. Per esempio, in Figura 11, sono mostrate una collezione di definizioni e una tabella di conversione di valuta. Entrambe sono oggetti concreti che sono stati costruiti partendo dalla stessa nozione astratta di dizionario.

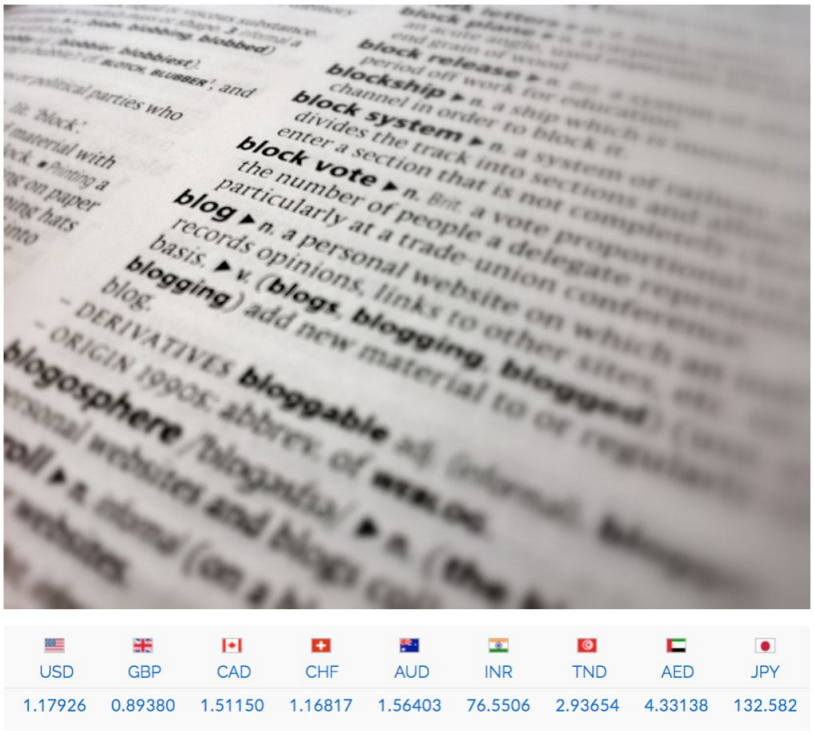

Figura 11. Due esempi di un dizionario espresso in oggetti del quotidiano: una collezione di definizioni (in alto) e una tabella di conversione da un euro a nove diverse valute (in basso). Foto in alto di Doug Belshaw, sorgente: https://www.flickr.com/photos/dougbelshaw/6877298592/. Schermata in basso ottenuta da http://www.xe.com/it/.

Albero

L'attività di marcatura di un testo – ovvero, l'annotazione che vi si può fare, riconoscendo i vari ruoli strutturali e semantici delle varie parti che lo compongono, come l'identificazione delle sezioni, capoversi, dialoghi, etc. – è un'attività che si compie sistematicamente, e in modo inconsapevole, ogni qual volta si prende un documento da analizzare o, molto più generalmente, da leggere. Per esempio, consideriamo il seguente estratto dal primo capitolo di *Alice's Adventure in Wonderland* di Lewis Carroll [Carroll, 1866]:

> Alice was beginning to get very tired of sitting by her sister on the bank, and of having nothing to do: once or twice she had peeped into the book her sister was reading, but it had no pictures or conversations in it, "and what is the use of a book," thought Alice, "without pictures or conversations?"
>
> So she was considering in her own mind, (as well as she could, for the hot day made her feel very sleepy and stupid,) whether the pleasure of making a daisy-chain would be worth the trouble of getting up and picking the daisies, when suddenly a white rabbit with pink eyes ran close by her.

Seppur sia totalmente implicito per chi legge, ogni parte del testo è in realtà organizzato in modo molto preciso. Per esempio, specifichi blocchi testuali del testo citato sono descritti in capoversi, che sono a loro volta organizzati in capitoli, che a loro volta compongono il libro. Inoltre, ogni capoverso può contenere altre strutture, come un dialogo di un certo personaggio. Tutte queste strutture sono mostrate in Figura 12, dove la struttura principale, chiamata *book*, è descritta da una sorta di scatola che contiene diverse scatole più piccole chiamate *chapter*, una per ogni capitolo. Ognuna di queste, a sua volta, contiene altre scatole chiamate *paragraph*, una per ogni capoverso, e così via. Questo approccio di racchiudere una parte del testo in una scatola etichettata definisce esattamente l'attività di marcatura, che è possibile definire esplicitamente mediante l'utilizzo di opportuni linguaggi di markup – alcuni dei quali verranno discussi in seguito.

Anche se potrebbe non sembrare estremamente chiaro ad una prima scorsa, l'organizzazione a scatole appena presentata descrive una precisa gerarchia tra loro, dove la più grande (ovvero *book*) ne contiene di più piccole (ovvero i vari *chapter*), queste a loro volta ne contengono di più piccole ancora (i *paragraph*), e così via. Quando siamo in presenza di queste organizzazioni gerarchiche che non si sovrappongono, possiamo usare una specifica struttura dati per descriverle in modo astratto: un *albero* – come mostrato in Figura 13.

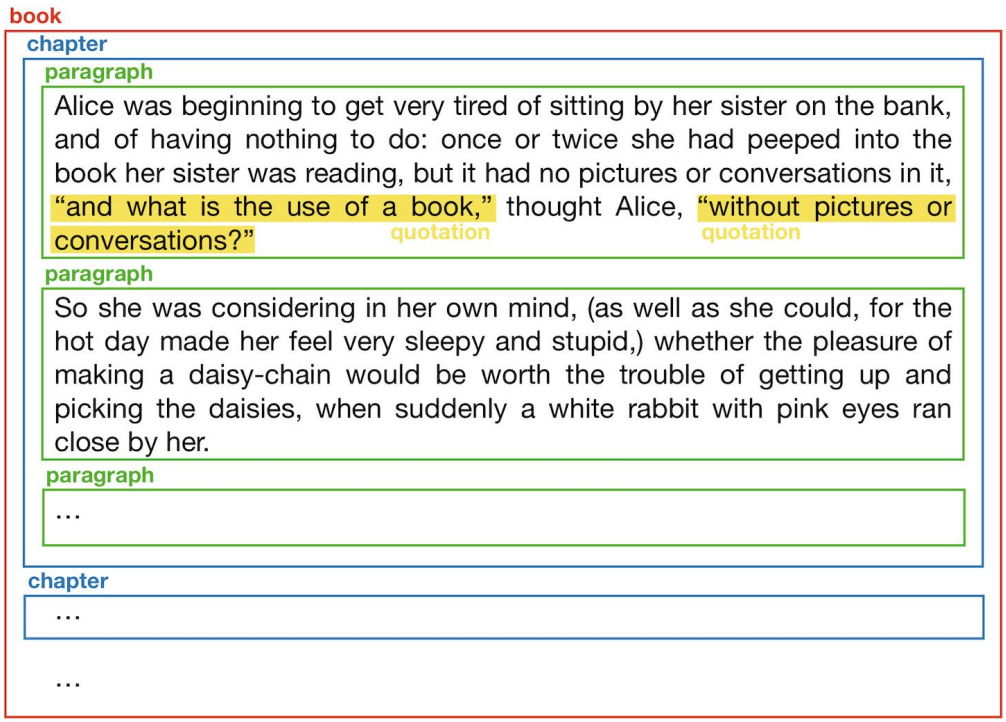

Figura 12. I primi due capoversi di *Alice's Adventure in Wonderland* marcati con strutture testuali di base: libro (*book*), capitolo (*chapter*), capoverso (*paragraph*), e dialogo (*quotation*).

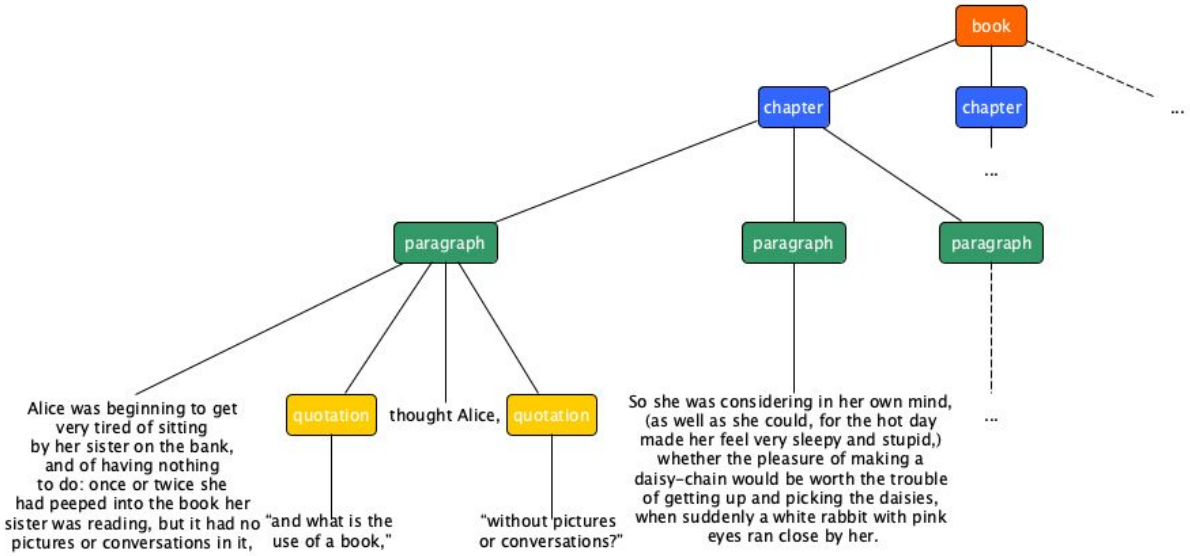

Figura 13. L'albero che descrive il contenimento delle varie strutture dell'incipit di *Alice's Adventure in Wonderland*.

Un albero è una struttura dati composta da un insieme di *nodi* collegati tra loro da una *relazione gerarchica genitore-figlio*. Come mostrato in Figura 14, i nodi di questa struttura dati vengono

disposti dall'alto verso il basso, contrariamente all'organizzazione dell'albero che siamo abituati ad osservare in natura.

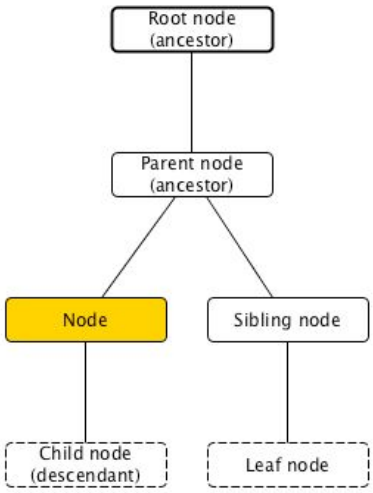

Figure 14. Un albero con la tipica nomenclatura dei suoi nodi a partire dal nodo centrale evidenziato in giallo. In questa figura, il bordo in grassetto è usato per identificare il nodo radice dell'albero (*root node*), che è l'unico senza alcun genitore, mentre il bordo tratteggiato è usato per indicare i nodi foglia dell'albero (*leaf nodes*), ovvero i nodi sprovvisti di figli.

Il nodo di origine, ovvero quello più in alto, è chiamato nodo radice (*root node*). Invece, i nodi che terminano l'albero, chiamati nodi foglia (*leaf nodes*), sono posizionati in basso nell'albero, rispetto al nodo radice. Prendendo in considerazione uno specifico nodo dell'albero, come quello evidenziato in giallo in Figura 14, possiamo definire tutti i restanti nodi come segue:

- il nodo genitore (*parent node*) è quello direttamente connesso quando ci si muove verso il nodo radice;
- un nodo figlio (*child node*) è uno di quelli direttamente connessi quando ci si muove lontano dal nodo radice;
- un nodo fratello (*sibling node*) è uno di quelli che condivide lo stesso nodo genitore;
- un nodo antenato (*ancestor node*) è uno di quelli raggiungibili seguendo ripetutamente le relazioni genitore-figlio andando verso il nodo radice;
- un nodo discendente (*descendant node*) è uno di quelli raggiungibili seguendo ripetutamente le relazioni genitore-figlio spostandosi lontano dal nodo radice.

Anche se abbiamo usato un esempio di contenimento a "scatole", le strutture ad albero possono rappresentare informazioni molto diverse, per esempio la prima frase dell'*incipit* di Alice in Wonderland può essere analizzato sintatticamente come in Figura 15, ma le relazioni non sono di contenimento, come avviene anche nel caso degli "alberi delle decisioni" (immaginate le istruzioni per usare un bancomat) o nel caso degli organigrammi gerarchici di un'azienda.

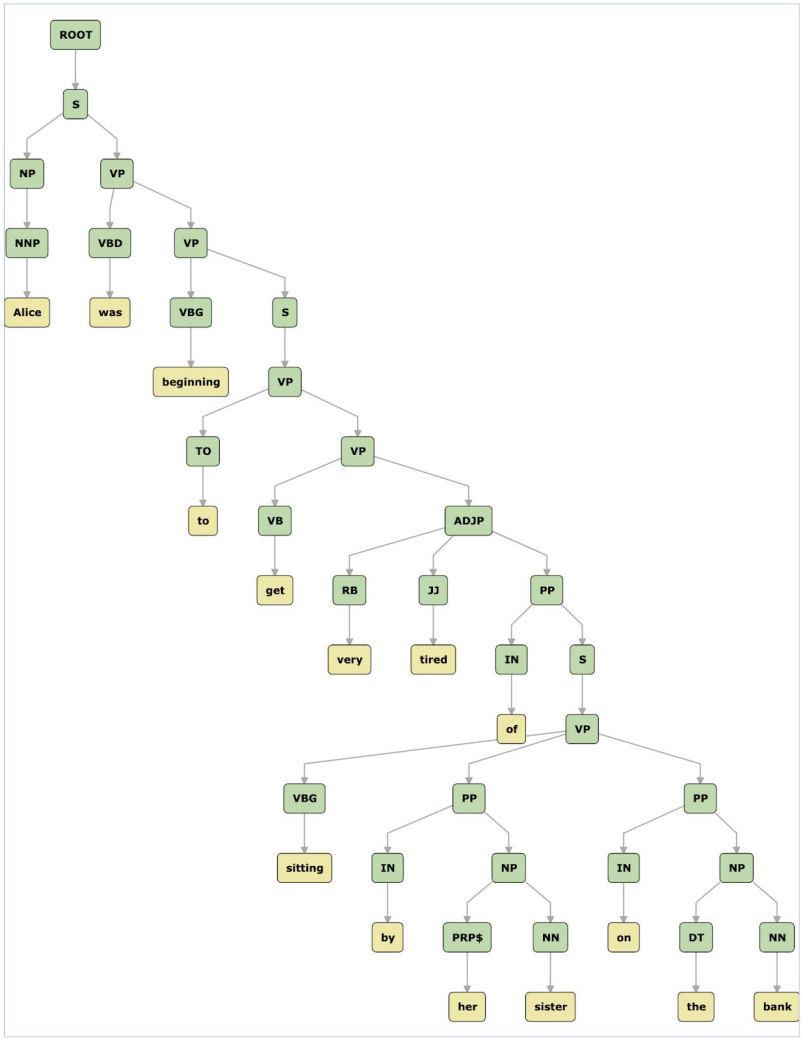

Figura 15. Un “albero di analisi a costituenti” della frase “Alice was beginning to get very tired of sitting by her sister on the bank”.
Sorgente: CoreNLP parser.

Grafo

L’origine della struttura a grafo deriva da un piccolo gioco, conosciuto ai tempi come un vero e proprio problema matematico, che riguarda sette ponti di una specifica città, Königsberg, illustrata in Figura 16. Il problema può essere enunciato come segue: è possibile fare una passeggiata in città attraversando ogni ponte una ed una sola volta? Molte persone hanno provato a proporre una soluzione a questo enigma, dimostrato (negativamente) da Eulero nel 1736 mediante una dimostrazione matematica formale [Euler, 1741].

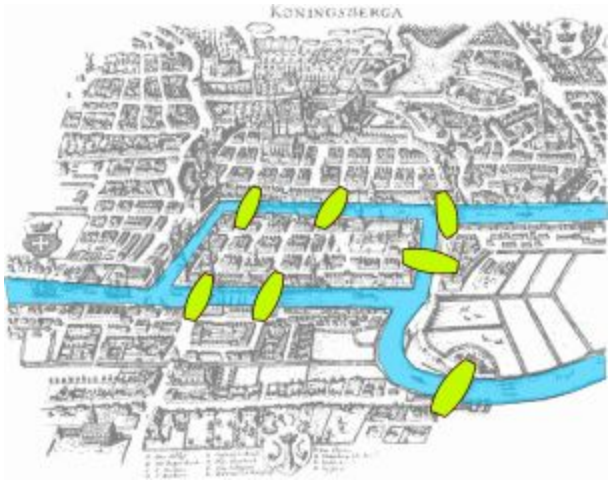

Figura 16. Una rappresentazione dei sette ponti di Königsberg.
Figura di Bogdan Giuşcă, sorgente: https://commons.wikimedia.org/wiki/File:Konigsberg_bridges.png.

Per questa dimostrazione, Eulero ha descritto, in modo astratto, le quattro terre che formavano Königsberg come *nodi* di una rete collegati da *archi*, dove ogni arco tra due nodi rappresentava un ponte tra due lembi di terra. Un'illustrazione di questa astrazione è mostrata in Figura 17. Usando questa rappresentazione astratta, poi conosciuta col nome di *grafo*, è stato in grado di dimostrare che non esiste alcuna soluzione al problema dei sette ponti di Königsberg.

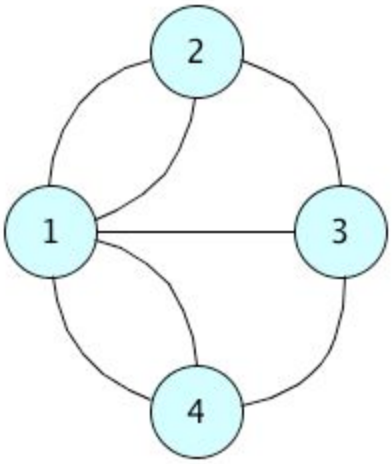

Figura 17. Una rappresentazione astratta dei sette ponti di Königsberg attraverso l'uso di un grafo.

La soluzione del problema era interamente basata sulla seguente intuizione. L'idea era che ogni nodo, ad eccetto del nodo di partenza e quello di arrivo, dovrebbero avere un numero pari di archi per essere raggiunti e poi abbandonati. Questa è un'implicazione pratica, derivata dagli spostamenti che una persona deve fare per arrivare in un lembo di terra e poi passare oltre. Infatti, ogni volta che si entra in un nodo bisogna percorrere un arco, e un altro arco è

comunque necessario per uscire da quel nodo. Quindi, per poter essere attraversati, ogni nodo che non sia quello di partenza o quello di arrivo deve avere necessariamente un numero pari di archi, in modo da essere transitato una o più volte. Tuttavia, tutti i nodi in Figura 17 hanno un numero dispari di archi, il che contraddice il precedente requisito. Di conseguenza, con questa configurazione, il problema dei sette ponti non può essere risolto.

I grafi sono una delle principali strutture dati in informatica e, in generale, del pensiero computazionale. Sono usati per descrivere, in termini astratti, molte situazioni del mondo reale come tragitti tra città, relazioni tra persone nei *social network*, l'organizzazione dei collegamenti ipertestuali tra pagine Web [Albert and Barabasi, 2002] e le relazioni concettuali nelle basi di dati o nei *knowledge graph* usati per esempio da Google (Figura 18), Amazon e Facebook per i loro servizi.

Questa struttura dati è interamente derivata dall'omonimo strumento matematico inventato da Eulero, e possono essere distinti in due macro-categorie: grafi non orientati (come quello usato da Eulero per risolvere il problema dei sette ponti di Königsberg, mostrato in Figura 17), dove un arco può essere attraversato in una direzione o nell'altra a piacere, ed i grafi orientati, dove ogni arco specifica esplicitamente la direzione di percorrenza, come mostrato in Figura 19.

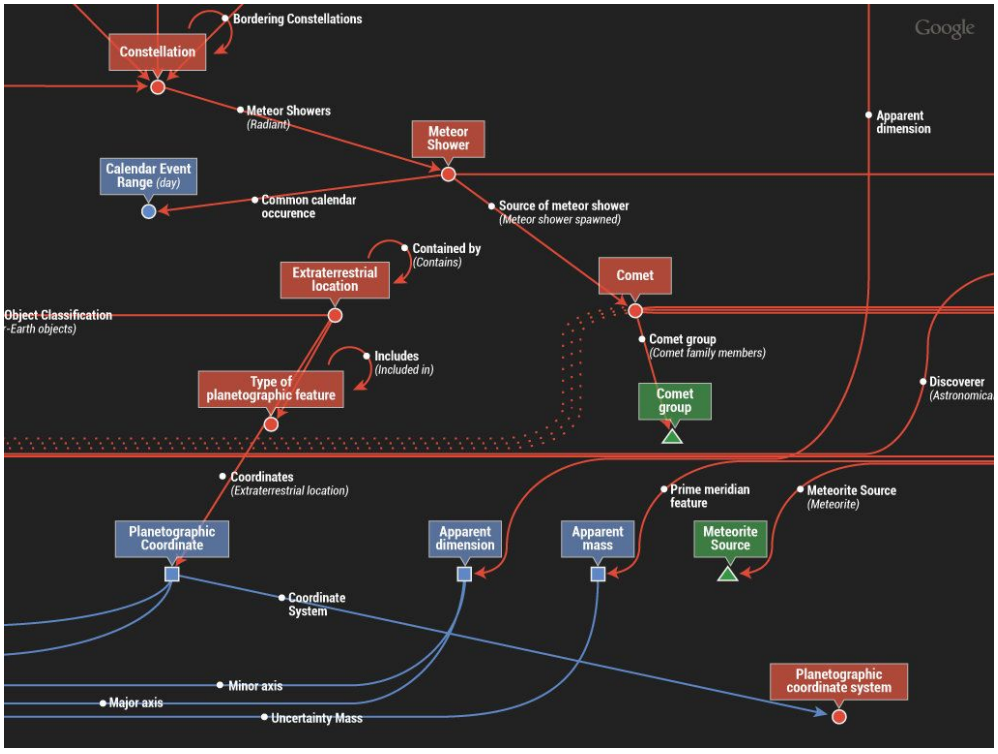

Figura 18. Un frammento del *knowledge graph* di Google.
Sorgente: Syracuse University Blog

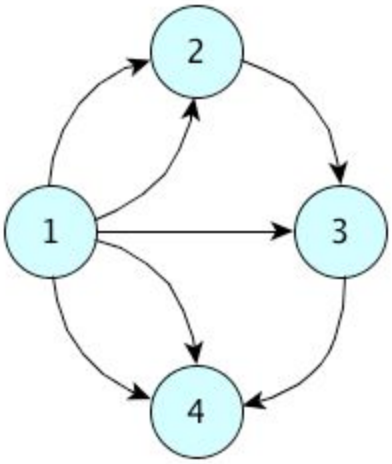

Figura 19. Un esempio di grafo orientato, in cui gli archi specificano la direzione di percorrenza attraverso una freccia – ad esempio, tra il nodo “1” e il nodo “2” ci sono due archi che possono essere percorsi solo dal primo nodo al secondo, ma non viceversa.

Bibliografia

Albert, R., Barabási, A.-L. (2002). Statistical mechanics of complex networks. Reviews of Modern Physics, 74 (47): 47-97. DOI: https://doi.org/10.1103/RevModPhys.74.47, anche disponibile al seguente URL: https://arxiv.org/pdf/cond-mat/0106096.pdf (controllato il 6 Gennaio 2019)

Campbell-Kelly, M. (2009). The Origin of Computing. Scientific American, 301 (September 2009): 62-69. DOI: https://doi.org/10.1038/scientificamerican0909-62, anche disponibile al seguente URL: http://www.cs.virginia.edu/~robins/The_Origins_of_Computing.pdf (controllato il 6 Gennaio 2019)

Carroll, L. (1866). Alice’s Adventures in Wonderland. Macmillan and Co. Available at https://en.wikisource.org/wiki/Alice%27s_Adventures_in_Wonderland_(1866) (last visited 8 December 2018)

Euler, L. (1741). Solutio problematis ad geometriam situs pertinentis. Commentarii academiae scientiarum Petropolitanae, 8 (1741): 128-140. http://eulerarchive.maa.org//docs/originals/E053.pdf (controllato il 6 Gennaio 2019)

Kramer, J. (2007). Is abstraction the key to computing? Communications of the ACM, 50 (4): 36-42. DOI: https://doi.org/10.1145/1232743.1232745, anche disponibile al seguente URL: https://www.ics.uci.edu/~andre/informatics223s2007/kramer.pdf (controllato il 6 Gennaio 2019)

Papert, S. (1980). Introduction: Computer for Children. In Mindstorms: children, computers, and powerful ideas: 3-18. New York, USA: Basic Books, Inc. ISBN: 0-465-04627-4. Disponibile al seguente URL: http://worrydream.com/refs/Papert%20-%20Mindstorms%201st%20ed.pdf (controllato il 6 Gennaio 2019)

Roegel, D. (2010). The great logarithmic and trigonometric tables of the French Cadastre: a preliminary investigation. Research Report. INRIA. https://hal.inria.fr/inria-00543946

Wing, J. M. (2008). Computational thinking and thinking about computing. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 366 (1881): 3717. DOI: https://doi.org/10.1098/rsta.2008.0118