Dispense per il corso di algoritmica per il web

Sebastiano Vigna

7 novembre 2022

Contents

1 Notazione e definizioni di base

Il prodotto cartesiano degli insiemi X e Y è l'insieme $X \times Y = \{\langle x, y\rangle \mid x \in X \wedge y \in Y\}$ delle coppie ordinate degli elementi X e Y. La definizionesi estende per ricorsione a n insiemi. Al prodotto cartesiano $X_1 \times X_2 \times \cdots \times X_n$ sono naturalmente associate le *proiezioni* $\pi_1, \pi_2, \ldots, \pi_n$ definite da

$$\pi_i(\langle x_1, x_2, \ldots, x_n\rangle) = x_i \tag{1}$$

poniamo

$$X^n = \overbrace{X \times X \times \cdots \times X}^{n \text{ volte}} \tag{2}$$

e $X^0 = \{*\}$ (qualunque insieme con un solo elemento). La *somma disgiunta* degli insiemi X e Y é, intuitivamente, un'unione di X e Y che peró tiene separati gli elementi comuni, quindi evita i conflitti. Formalmente:

$$X + Y = X \times \{0\} \cup Y \times \{1\} \tag{3}$$

Di solito ometteremo, con un piccolo abuso di notazione, la seconda coordinata. Una *relazione* tra gli insiemi X_1, X_2, ..., X_n é un sottoinsieme R del prodotto cartesiano $X_0 \times X_1 \times \cdots \times X_n$. Se $n = 2$ si tende a scrivere $x\ R\ y$ per $\langle x, y\rangle \in \mathbb{R}$. Una relazione tra due insiemi é detta *binaria*. Se R é una relazione binaria tra X e Y, X é detto *dominio* di R, ed é denotato da $\mathrm{dom}(R)$, mentre Y é detto *codominio* di R, ed é denotato da $\mathrm{cod}(R)$. Il *rango* o *insieme di definizione* di R é l'insieme $\mathrm{ran}(R) = \{x \in X \mid \exists y \in Y, x\ R\ y\}$, e in generale puó non coincidere con il dominio di R. L'*immagine* di R é l'insieme $\mathrm{imm}(R) = \{y \in Y \mid \exists x \in X, x\ R\ y\}$, e in generale puó non coincidere con il codominio di R. Una relazione binaria R tra X e Y é *monodroma* se per ogni $x \in X$ esiste al piú un $y \in Y$ tale che $x\ R\ y$. É *totale* se per ogni $x \in X$ esiste un $y \in Y$ tale che $x\ R$, cioé se $\mathrm{ran}(R) = \mathrm{dom}(R)$. É *iniettiva* se per ogni $y \in Y$ esiste al piú un $x \in X$ tale che $x\ R\ y$. É *suriettiva* se per ogni $y \in Y$ esiste un $x \in X$ tale che $x\ R\ y$, cioé se $\mathrm{imm}(R) = \mathrm{cod}(R)$. É *biiettiva* se la relazione é sia iniettiva che suriettiva. Una *funzione* da X a Y é una relazione monodroma e totale tra X e Y (notate che l'ordine é rilevante[1]); in tal caso scriviamo $f : X \to Y$ per dire che f "va da X a Y". Se f é una funzione da X a Y é uso scrivere $f(x)$ per l'unico $y \in Y$ tale che $x\ f\ y$, diremo che f *mappa* x in $f(x)$ o, in simboli, $x \mapsto f(x)$. Le nozioni di dominio, codominio, iniettivitá, suriettivitá e biiettivitá vengono ereditate dalle relazioni. Se una funzione $f : X \to Y$ é biiettiva, é facile verificare che esiste una funzione inversa f^{-1}, che soddisfa le equazioni $f(f^{-1}(y)) = y$ e $f^{-1}(f(x)) = x$

[1]Secondo la mia interpretazione, una funzione é monodroma e totale perché una funzione é definita come una relazione in cui ogni elemento del dominio é mappato in uno e un solo elemento del codominio, dunque:

- monodroma garantisce che ogni elemento dell'insieme di definizione ha un'unica immagine.
- totale garantisce che $\mathrm{dom}(R) = \mathrm{ran}(R)$.

per ogni $x \in X$ e $y \in Y$. Una *funzione parziale* (che tecnicamente non é una funzione perché non é definita sull'interezza del suo dominio) da X a Y é una relazione monodroma tra X e Y; una funzione parziale puó non essere definita su elementi del suo dominio, fatto che denotiamo con la scrittura $f(x) = \perp$ ("$f(x)$ é indefinito" o "f é indefinita su x"), che significa che $x \notin \text{ran}(f)$. Date funzioni parziali $f : X \to Y$ e $g : Y \to Z$, la *composizione* $g \circ f$ di f con g é la funzione definita da $(g \circ f)(x) = g(f(x))$. Si noti che, per convenzione, $f(\perp) = \perp$ per ogni funzione parziale f. Dati gli insiemi X e Y, denotiamo con $Y^X = \{f | f : X \to Y\}$ l'insieme delle funzioni da X a Y. Si noti che per insiemi finiti[2] $|Y^X| = |Y|^{|X|}$.
Denoteremo con n l'insieme $\{0, 1, \ldots, n-1\}$.
Dato un insieme X, il *monoide libero* su X, denotato da X^*, é l'insieme di tutte le sequenze finite, (inclusa quella vuota, normalmente denotata da ε) di elementi di X, dette *parole* su X, dotate dell'operazione di concatenazione, di cui la parola vuota é l'elemento neutro. Denoteremo con $|w|$ il numero di elementi di X della parola $w \in X$. Dato un sottoinsieme A di X, possiamo associargli la sua *funzione caratteristica* $\chi_A : X \to 2$ definita da:

$$\chi_A = \begin{cases} 0, & \text{se } x \notin A \\ 1, & \text{se } x \in A \end{cases} \tag{4}$$

Per contro, a ogni funzione $f : X \to 2$ possiamo associare il sottoinsieme di X dato dagli elementi mappati da f in 1, cioé l'insieme $\{x \in X \mid f(x) = 1\}$; tale corrispondenza é inversa alla precedente, ed é quindi naturalmente equivalente considerare sottoinsiemi di X o funzioni di X in 2. Date due funzioni f, $g : \mathbb{N} \to \mathbb{R}$, diremo che f é di *ordine non superiore* a g, e scriveremo che $f \in \mathcal{O}(g)$ ("f é $\mathcal{O}$-grande di g") se esiste una costante $a \in \mathbb{R}$ tale che $|f(n_0)| \leq |ag(n_0)|$ definitivamente. Diremo che f é di *ordine non inferiore* a g, e scriveremo che $f \in \Omega(g)$ se $g \in \mathcal{O}(f)$. Diremo che f é *dello stesso ordine di* g e scriveremo $f \in \Theta(g)$, se $f \in \mathcal{O}(g)$ e $g \in \Omega(f)$.
Un *grafo semplice* G é dato da un insieme finito di vertici V_G e da un insieme di lati $E_g \subseteq \{\{x, y\} \mid x, y \in V_G \wedge x \neq y\}$; ogni lato é cioé una coppia non ordinata di vertici distinti. Se $\{x, y\} \in E_G$, diremo che x e y sono vertici *adiacenti* in G. Un grafo puó essere rappresentato graficamente disegnando i suoi vertici come punti sul piano, e rappresentato i lati come segmenti che congiungono vertici adiacenti. Per esempio, il grafo con insieme di vertici 4 e insieme di lati $\{\{0, 1\}, \{1, 2\}, \{2, 0\}, \{2, 3\}\}$ puó essere rappresentato come segue:

L'*ordine* di G é il numero naturale $|V_G|$. Una *cricca* o una *clique* di G é un insieme di vertici $C \subseteq V_G$ mutualmente adiacenti (nell'esempio in figura $\{0, 1, 2\}$ é una cricca). Dualmente, un *insieme indipendente* di G é un insieme di vertici $I \subseteq V_G$ mutualmente non adiacenti. Un *cammino* di lunghezza n in G é una sequenza di vertici $x_0, \ldots, x_n$ tale che x_i é adiacente a x_{i+1} con $(0 \leq i < n)$. Diremo che il cammino va da x_0 a x_n. Nell'esempio in figura, 0, 1, 2 é un

[2]Si noti che l'uguaglianza é vera in generale, utilizzando i cardinali cantoriani

cammino, 1, 3 non lo é.
Un grafo *orientato* G é dato da un insieme di nodi N_G e un insieme di archi A_G e da funzioni s_G, $t_G : A_G \to n_G$ (*source*, *target*) che spcificano l'inizio e la fine di ogni arco. Due archi a e b tali che $s_G(a) = s_G(b)$ e $t_G(a) = t_G(b)$ sono detti *paralleli*. Un grafo senza archi paralleli é detto *separato*. Il *grado positivo* o *outdegree* $d^+(x)$ di un nodo x é il numero di archi uscenti da x, cioé $|s_G^{-1}(x)|$. Dualmente, il *grado negativo* o *indegree* $d^-(x)$ di un nodo x é il numero di archi entranti in x, cioé $|t_G^{-1}(x)|$. In un grafo orientato G un *cammino* di lunghezza n é una sequenza di vertici e archi $x_0, a_0, x_1, a_1, \ldots, a_{n-1}, x_n$ tale che $s_G(a_i) = x_i$ e $t_G(a_i) = x_{i+1}$ per $0 \leq i < n$. Diremo che il cammino va da x_0 a x_n. Definiamo la relazione di *raggiungibilitá*: $x \rightsquigarrow y$ se esiste un cammino da x a y. La relazione di equivalenza $\sim$ é ora definita da $x \sim y \iff x \rightsquigarrow y \wedge y \rightsquigarrow x$. Le classu di equivalenza di $\sim$ sono dette *componenti fortemente connesse* di G, G é *fortemente connesso* quando é costituito da una sola componente.
La funzione $\lambda(x)$ denota il bit piú significativo dell'espansione binaria di x: quindi $\lambda(1) = \lambda(1_2) = 0$, $\lambda(2) = \lambda(10_2) = 1$ etc...
Per convenzione $\lambda(0) = -1$. Si noti che per $x > 0$ si ha $\lambda(x) = \lfloor \log x \rfloor$.

2 Crawling

Il *crawling* é l'attivitá di scaricamento delle pagine web. Un *crawler* é un dispositivo software che visita, scarica e analizza i contenuti delle pagine web a partire da un insieme di pagine dato, detto *seme*. Il crawler procede nel suo processo di visita seguendo i collegamenti ipertestuali conenuti nelle varie pagine.
Le pagine web durante il processo di crawl si dividono in tre:

- L'insieme delle pagine *visitate*, V, che sono giá state scaricate e analizzate;
- La *frontiera*, F, che é l'insieme delle pagine conosciute ma che non sono ancora state visitate;
- Línsieme U degli URL sconosciuti.

Le differenze tra l'attivitá di crawling e una banale visita all'interno di un grafo sono molto importanti, prima di tutto c'é il fatto che un crawl ha una dimensione ignota, non conosciamo $|V_G|$; secondariamente la frontiera é un enorme problema, in quanto la sua dimensione tende a crescere molto piú velocemente dell'insieme dei visitati.
In generale l'operazione di crawling parte caricando il seme in frontiera e, finché la frontiera non é vuota, viene estratto un URL dalla frontiera, secondo determinate politiche, l'URL viene visitato (e quindi scarica la pagina corrispondente), lo analizza derivandone nuovi URL tramite i collegamenti ipertestuali contenuti all'interno della pagina e sposta l'URL nell'insieme dei visitati. I nuovi URL vengono invece aggiunti alla frontiera se sono sconosciuti, e quindi non sono in $V \cup F$.
Diverse politiche di prioritizzazione della frontiera possono poi dare luogo ad approcci diversi al processo di crawling, posso per esempio estrarre prima degli URL a cui si arriva partendo da pagine che contengono determinate parole chiave.

2.1 Il crivello

Il crivello é la struttura dati di base di un crawler, questo accetta in ingresso URL potenzialmente da visitare e permette di prelevare URL pronti alla visita. Ogni URL viene estratto una e una sola volta in tutto il processo di crawling, indipendentemente da quante volte é stato inserito all'interno della struttura. In questo senso il crivello unisce le proprietá di un dizionario a quelle di una coda con prioritá e rappresenta al tempo stesso la frontiera, l'insieme dei visitati e la coda di visita. Combinare questi aspetti in una sola struttura é un lavoro complesso ma permette risparmi notevoli dal punto di vista pratico[3].
Una prima osservazione é che spesso, per mantenere l'informazione di quali URL sono stati giá visitati ($V \cup F$) é preferibile sostituire gli URL con delle *firme*, cioé con il risultato del calcolo di $h(u)$, dove h é una funzione di hash definita

[3]Si noti che é possibile riordinare ulteriormente gli URL *dopo* l'uscita dal crivello

sulle stringhe e restituisce un hash di dimensione arbitraria, per esempio 64 bit. Questo ha due grandi benefici:

- Risparmio di spazio non trascurabile, molti URL possono essere di grandi dimensioni e salvarli sempre in un centinaio[4] si rivela una buona fonte di risparmio
- Uniformiamo le lunghezze degli URL a un valore standard

Il drawback di una soluzione del genere é che accettiamo il fatto che vi siano delle collisioni, é dunque possibile che due URL diversi vengano mappati sullo stesso valore di hash. Questo fenomeno é inevitabile, peró se abbiamo una funzione di hash che lavora su un numero di bit abbastanza grande, la probabilitá di incontrare una collisione sará cosí bassa da essere trascurabile.
Github.com implementa una soluzione del genere, viene impiegato SHA-1 (funzione di hash a 160bit) per calcolare un hash dell'URL di ciascuna delle repository nei loro database, la probabilitá di collisione é cosí bassa che é sostanzialmente impossibile. Adesso sembra che vogliano muoversi verso SHA-256.
Supponiamo ora di avere in memoria n firme, la probabilitá che una nuova firma collida con una di quelle esistenti é n/u, dove u é la dimensione dell'universo delle possibili firme. Nel caso di un sistema a 64 bit $u = 2^{64}$, e quindi possiamo memorizzare 100 miliardi di URL con una probabilitá di falsi positivi nell'ordine di $10^{11}/2^{64} < 2^{37}/2^{64} = 1/2^{27} < 1/10^8$, quindi avremo meno di un errore ogni 100 milioni di URL.

[4] valore d'esempio arbitrario