Analisi di Quicksort*

Federico Fusco

Anno Accademico 2024/2025

In queste note studiamo la versione randomizzata dell'algoritmo di Quicksort in cui i pivot vengono scelti uniformemente a caso. Dopo aver richiamato lo pseudo-codice dell'algoritmo, analizzeremo in valore atteso ed in alta probabilità il suo tempo di esecuzione.

Assunzione 1. *Per semplicità assumiamo che gli elementi della lista siano totalmente ordinati. In particolare, eventuali "pareggi" tra elementi devono essere risolti in modo arbitrario ma consistente.*

1 Quicksort

Algorithm 1 Quicksort Randomizzato

1: **Input**: lista S di n elementi da ordinare
2: **if** $|S| \leq 1$ **then** ▷ Passo base della ricorsione
3: **Return** S
4: Sia x un elemento di S estratto uniformemente a caso. ▷ x è il pivot!
5: Inizializzare due liste vuote S_1 ed S_2
6: **for** y in S **do**
7: **if** $y < x$ **then**
8: $S_1 \leftarrow S_1 + y$
9: **else**
10: $S_2 \leftarrow S_2 + y$
11: Ordinare S_1 usando Quicksort Randomizzato ▷ Ricorsione su S_1
12: Ordinare S_2 usando Quicksort Randomizzato ▷ Ricorsione su S_2
13: **Return** $S_1 + [x] + S_2$

Una versione "equivalente" dello pseudo-codice proposto consiste nell'operare una permutazione casuale alla lista S e estrarre i pivot (linea 4 nello pseudo-codice) in maniera arbitraria.

1.1 Tempo di esecuzione atteso

La versione non randomizzata di Quicksort è caratterizzata da un tempo di esecuzione, nel caso peggiore, che è dell'ordine di $\Theta(n^2)$. Questo è dovuto al fatto che una scelta "sbilanciata" dei pivot può indurre l'algoritmo a generare un albero di chiamate ricorsive di altezza lineare nella dimensione dell'input. Scegliere i pivot in maniera uniforme aiuta a evitare questi casi limite: in media ci aspettiamo infatti che un pivot casuale divida il vettore a metà. Possiamo formalizzare questa intuizione nel seguente Teorema.

*Queste note sono probabilmente piene di imprecisioni varie. Segnalarle a federico.fusco@uniroma1.it. Per una trattazione più sistematica si rimanda ai seguenti manuali: Kleinberg and Tardos [2006], Mitzenmacher and Upfal [2005], Cormen et al. [2009].

Teorema 1. *Quicksort randomizzato ordina un vettore di n elementi in tempo $O(n \log n)$, in valore atteso rispetto alla sua randomizzazione interna.*

Dimostrazione. Per ogni input S, è possibile definire la variabile aleatoria T_S che rappresenta il tempo di esecuzione dell'algoritmo randomizzato su S. Vogliamo dimostrare che, per n sufficientemente grande, e qualsiasi input S di n elementi, il valore atteso della variabile aleatoria T_S sia $O(n \log n)$.

Fissiamo quindi un n sufficientemente grande e un qualsiasi input S di n elementi. Vogliamo dimostrare che $\mathbb{E}[T]$ [*] $\leq Cn \log n$, dove C è una qualche costante che non dipende da n e da S. Dallo pseudocodice è facile osservare che le uniche operazioni che possono essere ripetute più di $O(n)$ volte sono quelle contenute nel ciclo for: Quicksort randomizzato viene infatti chiamato ricorsivamente al più $O(n)$ volte e le operazioni contenute nelle altre linee impiegano tempo $O(1)$. Per concludere la dimostrazione è sufficiente quindi dimostrare che, durante tutta l'esecuzione dell'algoritmo, le operazioni compiute all'interno dei vari cicli for sono state $O(n \log n)$, in valore atteso. In particolare, contiamo il numero di volte X (che è una variabile aleatoria) di confronti (linea 7) che ci sono stati tra elementi di S e dimostriamo che sono al più $O(n \log n)$.

Immaginiamo di rinominare gli elementi di S in base al loro ordine: $S = [x_1, x_2, \ldots, x_n]$. Per ogni coppia di indici distinti i, j, definiamo la variabile aleatoria $X_{i,j}$ che vale 1 se gli elementi x_i e x_j vengono confrontati dall'algoritmo, e 0 altrimenti. Chiaramente X è dato dalla somma dei vari $X_{i,j}$. Per ogni coppia di indici i e j (senza perdita di generalità $i < j$), consideriamo l'insieme degli elementi $S_{i,j} = \{x_i, x_{i+1}, \ldots, x_j\}$ e immaginiamo di seguire il percorso di tali elementi durante un'esecuzione dell'algoritmo. Gli elementi in $S_{i,j}$ vengono divisi in due chiamate ricorsive distinte se e solo se uno di essi viene scelto come pivot; in particolare, x_i e x_j vengono confrontati se e solo se tale pivot è x_i o x_j. Concentriamoci su tale chiamata ricorsiva: condizionando rispetto all'evento che il pivot x sia in $S_{i,j}$ è chiaro che il pivot può essere un qualsiasi elemento di $S_{i,j}$, con probabilità uniforme. Abbiamo quindi che:

$$\mathbb{E}[X_{i,j}] = \mathbb{P}(X_{i,j} = 1) = \frac{2}{|S_{i,j}|} = \frac{2}{j - i + 1}, \tag{1}$$

dove abbiamo contato sia il caso in cui il pivot sia x_i che quello in cui sia x_j. Possiamo finalmente concludere la dimostrazione. Abbiamo infatti che:

$$\begin{aligned}
\mathbb{E}[X] &= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \mathbb{E}[X_{i,j}] && \text{(Linearità valore atteso)} \\
&= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j - i + 1} && \text{(Equazione 1)} \\
&= \sum_{i=1}^{n-1} \sum_{k=1}^{n-i} \frac{2}{k + 1} && \text{(Poniamo } k = j - i\text{)} \\
&\leq 2n \sum_{k=1}^{n} \frac{1}{k} \leq 2n \log n. && \text{(Teorema 3, Sezione A)}
\end{aligned}$$

□

1.2 Tempo di esecuzione in alta probabilità

Il risultato in valore atteso ci ha offerto una prima formalizzazione del fatto che Quicksort randomizzato si comporta molto meglio della sua variante deterministica. Tuttavia non ci offre

[*] Nota, per semplicità abbiamo omesso il pedice S da T.

molte garanzie su quello che potrebbe succedere su una specifica istanza dell'algoritmo. Usando tecniche di analisi probabilistica un po' più avanzate è possibile rafforzare questo risultato e dimostrare che Quicksort randomizzato ha tempo di esecuzione $O(n \log n)$ *con alta probabilità* (cioè con probabilità almeno $1 - \frac{c}{n}$, dove c è una costante universale). In particolare, in questa Sezione dimostriamo il seguente Teorema:

Teorema 2. *Quicksort Randomizzato ordina un vettore di n elemnti in tempo $O(n \log n)$ con alta probabilità.*

Prima di iniziare con la dimostrazione del Teorema, presentiamo una proposizione probabilistica che usiamo nella Sezione e il cui spirito può essere di interesse generale.

Proposizione 1. *Consideriamo il seguente processo: una moneta non bilanciata, che dà testa con probabilità $\frac{1}{3}$, viene lanciata finché non viene osservata testa per m volte. La probabilità che la moneta venga lanciata più di $18m$ volte è più piccola di e^{-2m}.*

Dimostrazione. Vorremmo usare Chernoff bound, ma non possiamo farlo direttamente perché il numero di volte che la moneta viene lanciata è la somma di m variabili aleatorie geometriche, non di Bernoulli. L'osservazione decisiva è che i due eventi di seguito sono equivalenti:

- Evento 1 (l'evento dell'enunciato): Lanciamo la moneta fino ad osservare m volte testa, e lanciamo la moneta più di $18m$ volte.
- Evento 2: Lanciamo la moneta esattamente $18m$ volte ed osserviamo un numero di teste che è minore strettamente di m.

Visto che i due eventi sono equivalenti, possiamo calcolare la probabilità del secondo evento, su cui possiamo applicare le disuguaglianze di Chernoff! Sia X_i la variabile aleatoria di Bernoulli che vale 1 se osserviamo testa all'i^o lancio, ed X la somma delle X_i. Usando la linearità del valore atteso abbiamo che $\mathbb{E}[X] = 6m$. Possiamo quindi usare la disuguaglianza moltiplicativa di Chernoff con $\delta = 5/6$ (rimandiamo alla Sezione B per l'enunciato preciso della versione che usiamo):

$$\mathbb{P}(X < m) \leq \mathbb{P}\left(X \leq \tfrac{1}{6}\mathbb{E}[X]\right) \leq e^{-\frac{25}{12}\mathbb{E}[X]} \leq e^{-2m}. \qquad \square$$

La dimostrazione del Teorema è preceduta da tre Lemmi preparatori che utilizzano la seguente interpretazione dell'algoritmo: ogni chiamata ricorsiva dell'algoritmo è associata al vettore che riceve in input ed è visualizzato come un nodo in un albero, la cui radice è la chiamata iniziale. Un nodo/chiamata di questo albero è detto *buono* se il pivot selezionato divide il relativo vettore (di cardinalità s) in due parti di cardinalità al più $\frac{2}{3}s$. Altrimenti il nodo è chiamato *cattivo*. Per visualizzare questi concetti rimandiamo alla Figura 1.

Lemma 1. *Consideriamo l'albero generato da una qualsiasi realizzazione di Quicksort randomizzato, e consideriamo un qualsiasi cammino radice-foglia p. Allora il numero di nodi buoni in p, chiamato N_p^b è al più $c \log_2 n$, dove $c = 1/(\log_2 3/2)$.*

Dimostrazione del Lemma 1. La radice contiene n nodi e, man mano che si scende lungo il cammino p, il numero di elementi contenuti nei nodi diminuisce. In particolare, ogni volta che si attraversa un nodo buono il numero di elementi diminuisce di un fattore moltiplicativo 2/3. Questo vuol dire che N_p^b rispetta la seguente disuguaglianza:

$$n \cdot \left(\frac{2}{3}\right)^{N_p^b} \geq 1 \implies N_p^b \leq c \log_2 n,$$

dove la costante c è quella riportata nell'enunciato del Lemma. $\square$

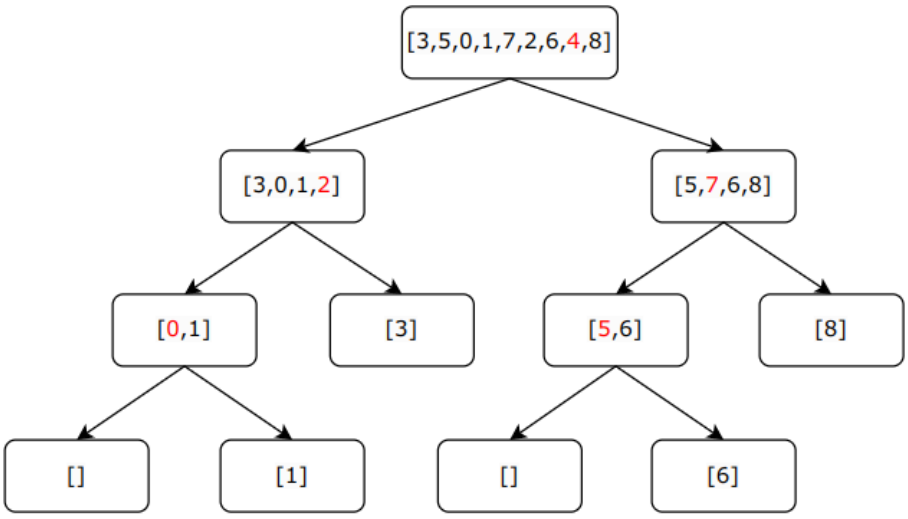


Figura 1: Albero delle chiamate ricorsive di un'esecuzione di Quicksort randomizzato sulla lista $[3, 5, 0, 1, 7, 2, 6, 4, 8]$. I pivot sono colorati in rosso.

Lemma 2. *Fissiamo un particolare cammino radice-foglia, e sia N_p il numero di nodi che lo compongono, allora*

$$\mathbb{P}\left(N_p \leq c' \log_2 n\right) \geq 1 - \frac{1}{n^2},$$

dove $c' = 18c$.

Dimostrazione del Lemma 2. In questa dimostrazione usiamo il risultato della Proposizione 1. Ogni cammino può essere infatti immaginato come una successione di "lanci di moneta" che corrispondono alla scelta del pivot; scegliere un buon pivot è equivalente ad ottenere testa (con probabilità $1/3$), mentre un cattivo pivot corrisponde ad ottenere croce (con probabilità $2/3$). Quindi possiamo direttamente applicare Proposizione 1, utilizzando come m il valore $c \log n$ che abbiamo ricavato dal Lemma precedente. □

Lemma 3. *Dimostrare che, con probabilità almeno $1 - \frac{1}{n}$, il numero di nodi sul percorso foglia-radice più lungo sia al massimo $c' \log_2 n$.*

Dimostrazione del Lemma 3. Per ogni cammino p, denotiamo con $\mathcal{E}_p$ l'evento "il cammino p ha al massimo $c' \log_2 n$ nodi". Stiamo cercando un limite inferiore alla probabilità dell'evento $\mathcal{E}$ definito come l'intersezione su tutti i cammini degli eventi $\mathcal{E}_p$. Siamo P l'insieme di tutti i cammini; abbiamo la seguente disuguaglianza che deriva dalle leggi di de Morgan e union-bound:

$$\mathbb{P}\left(\mathcal{E}\right) = \mathbb{P}\left(\bigcap_{p \in P} \mathcal{E}_p\right) = 1 - \mathbb{P}\left(\bigcup_{p \in P} E_p^C\right) \geq 1 - \sum_{i=1}^{n} \frac{1}{n^2} = 1 - \frac{1}{n}.$$

Nota, l'ultima disuguaglianza deriva da Lemma 2 e dal fatto che ci sono al più n percorsi distinti in qualsiasi albero. □

A questo punto abbiamo tutti gli ingredienti per dimostrare il Teorema.

Dimostrazione del Teorema 2. Il tempo di esecuzione di una qualsiasi realizzazione dell'algoritmo di Quicksort randomizzato è uguale al numero di chiamate ricorsive/nodi, moltiplicato per

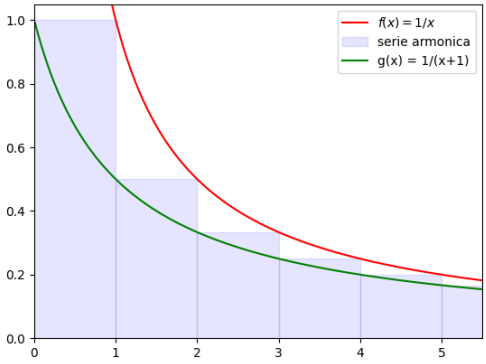
$O(n)$ (perché in ogni chiamata ricorsiva l'algoritmo compara tutti gli $O(n)$ elementi del vettore con il pivot). In Lemma 3 abbiamo dimostrato che, con probabilità almeno $1 - 1/n$, Quicksort viene chiamato ricorsivamente al più $O(\log n)$ volte, questo è sufficiente a concludere che, con la stessa probabilità, l'algoritmo termina in $O(n \log n)$. □

Riferimenti bibliografici

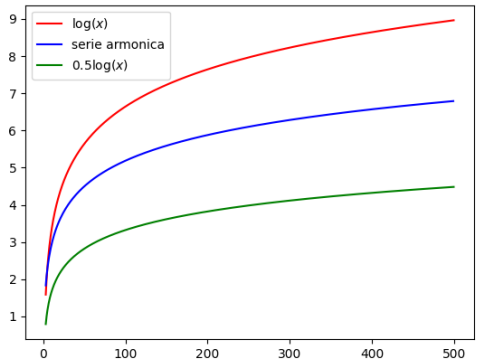
Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, 3rd Edition*. MIT Press, 2009.

Jon M. Kleinberg and Éva Tardos. *Algorithm design*. Addison-Wesley, 2006.

Michael Mitzenmacher and Eli Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, 2005.



(a) Confronto tra i "rettangoli", $\frac{1}{x}$ e $\frac{1}{x+1}$.



(b) Confronto tra la serie armonica, $\log x$ e $\frac{1}{2}\log x$

Figura 2: Figure di supporto per l'Appendice A

A Serie Armonica

La serie armonica è uno strumento matematico fondamentale in informatica. In particolare, è cruciale studiare la velocità con cui tale serie diverge.

Teorema 3. *Per ogni $n \geq 10$ le seguenti disuguaglianze sono verificate:*

$$\tfrac{1}{2}\log n \leq \sum_{k=1}^{n} \frac{1}{i} \leq \log n.$$

Dimostrazione. La serie ha una semplice visualizzazione geometrica (Figura 2a): possiamo immaginare infatti di sommare l'area di tanti rettangoli di altezza $\frac{1}{i}$ aventi come base $[i-1, i]$ e confrontare tale somma con l'integrale di $\frac{1}{x}$ e $\frac{1}{x+1}$. Partiamo da $\frac{1}{x}$, abbiamo la seguente disuguaglianza:

$$\sum_{k=1}^{n} \frac{1}{i} \leq 1 + \int_{1}^{n} \frac{1}{x} dx = 1 + \ln n = 1 + \frac{\log n}{\log e} \leq \log n,$$

dove l'ultima disuguaglianza può essere verificata analiticamente per ogni $n \geq 10$. Analogamente possiamo dimostrare l'altra disuguaglianza:

$$\sum_{k=1}^{n} \frac{1}{i} \geq \int_{0}^{n} \frac{1}{x+1} dx = \ln(n+1) \geq \frac{1}{2}\log n.$$

In Figura 2b viene rappresentato l'andamento effettivo della serie armonica e delle due funzioni $\log x$ e $\frac{1}{2}\log x$. □

B Disuguaglianza di Chernoff moltiplicativa

Nella dimostrazione della Proposizione 1 abbiamo usato la seguente versione moltiplicativa delle note disuguaglianze di Chernoff (per una dimostrazione si rimanda a Teorema 4.5. della seconda edizione di Mitzenmacher and Upfal [2005]).

Teorema 4. *Siano $X_1, X_2, \ldots, X_m$ variabili aleatorie indipendenti a valori in $\{0,1\}$, sia X la loro somma e $\mu = \mathbb{E}[X]$. Per ogni $\delta \in (0,1)$ la seguente disuguaglianza è verificata:*

$$\mathbb{P}(X \leq (1-\delta)\mu) \leq e^{-\mu\frac{\delta^2}{2}}.$$