Programmazione

Dipartimento di Matematica

Ing. Cristiano Gregnanin

Corso di laurea in Matematica

23 aprile 2015

Analisi prestazionale

La bontà di un algoritmo viene valutata analizzando due fattori:

- Quanto è veloce (complessità in tempo)
- Quante risorse richiede (complessità in spazio)

Solitamente, a meno di richieste esagerate, la complessità spaziale non è un problema, quindi bada molto più a quanto tempo ci vuole per eseguire un compito

Analisi prestazionale

La complessità temporale di un algoritmo però non è di semplice valutazione.
Tempo di esecuzione:

- dipende dalla potenza della macchina su cui è eseguito
- dipende dalla dimensione dell'input (molti dati molto più tempo)

La soluzione è stata di assegnare un costo alle operazioni e calcolare quindi la complessità totale come costo complessivo:

- l'operazione di confronto ha costo unitario
- tutte le altre hanno costo nullo (non è vero, ma hanno un costo irrisorio rispetto al confronto)

Analisi prestazionale

Inoltre non è possibile dare una misura puntuale del costo di un algoritmo, poiché molte volte esso dipende dallo spazio dell'input: si calcola quindi il costo in funzione della dimensione dell'input **costo = f(n)** , dove **n** è la dimensione dei dati su si opera.
Infine è necessario calcolare il costo nel caso migliore e nel caso peggiore :

- **o(f(n))** per la complessità nel caso **migliore** (costo minimo)
- **O(f(n))** per la complessità nel caso **peggiore**.

Analisi prestazionale

Esempio:
Dati n numeri, trovare il più grande
Algoritmo:
MAX = primo numero
Per ogni numero seguente, se è maggiore di quello in MAX, memorizzalo in MAX
Complessità: nel caso peggiore, il valore massimo si trova in ultima posizione; in questo caso sono necessari n-1 confronti. Quindi la complessità dell'algoritmo è **O(n)**. Tale funzione ($f(n) = n$) è detta anche lineare, dato che cresce linearmente con n.

Analisi prestazionale

In realtà, l'esempio precedente, avrebbe complessità $O(n-1)$, ma poiché **per** $n >> 1$ $O(n-1) \approx O(n)$, viene utilizzato $O(n)$ **come valore di complessità.**
Questo vale per tutte le funzioni $f(n)$ indicanti la complessità di un algoritmo:

- $O(2n+4)$, per $n >> 1$ si ha $O(n)$
- $O(2n2+n+4)$, per $n >> 1$ si ha $O(n2)$
- Ecc..

cioè viene individuata la funzione maggiorante.

Analisi prestazionale

La complessità dà un'idea della convenienza dell'algoritmo. Supponiamo il costo di una operazione pari ad 1 secondo, il tempo necessario per un algoritmo con le varie complessità è il seguente:

n - complessità	**(log(n))**	**O(n)**	**O(n^2)**	**O(2^n)**
10	3 sec	10 sec	1,5 min	17 min
20	4-5 sec	20 sec	7 min	291 giorni
30	5 sec	30 sec	15 min	191 anni

Come si vede, un problema con complessità polinomiale diventa molto velocemente intrattabile

Esempio: gestione di una biblioteca

- Ogni libro si trova su uno scaffale e può essere preso in prestito ed in seguito restituito.
- La biblioteca è dotata di uno schedario ed ogni scheda contiene:
  - Nome,cognome dell'autore (o autori)
  - Titolo
  - Data di pubblicazione
  - Numero scaffale in cui si trova
  - Numero d'ordine della posizione
- Le schede sono disposte in ordine alfabetico in base all'autore ed al titolo.

Esempio: gestione di una biblioteca

Algoritmo di ricerca di un libro in biblioteca:

1. Ricerca della scheda nello schedario
2. Lettura del numero di scaffale e posizione del libro
3. Ricerca dello scaffale indicato
4. Prelievo del libro e scrittura sulla scheda delle informazioni sul prestito: data, nome, ecc..

Ogni passo può (deve) essere descritto più dettagliatamente: sotto-algoritmo

Passo 1:

- Lettura della prima scheda dello schedario
- Se il nome dell'autore ed il titolo coincidono, ricerca conclusa
- Si ripete il passo 2 fino trovare la scheda cercata o a terminare le schede
- Se non viene trovata la scheda, la ricerca termina in modo infruttuoso

Esempio: gestione di una biblioteca

Il sotto-algoritmo per il passo 1, è corretto ma non efficiente:
ha una complessità lineare con il numero di schede, ossia nel caso peggiore tutte le schede nello schedario verranno analizzate.
Si può definire un altro algoritmo per il passo 1 più efficiente:

1. Se lo schedario è vuoto ricerca infruttuosa
2. Lettura scheda centrale
3. Se è quella cercata, ricerca conclusa con successo
4. Se il nome dell'autore e/o titolo è precedente, si ripete l'algoritmo sulla prima metà dello schedario
5. Se il nome dell'autore e/o titolo è successivo, si ripete l'algoritmo sulla seconda metà dello schedario.

Nel caso peggiore, questo algoritmo analizza $log2(n)$ schede:

- 16000 schede, alg1 = 16000 confronti
- 16000 schede, alg2 = 14 confronti

Esempio: gestione di una biblioteca

Nel secondo algoritmo presentato per automatizzare il passo 1, l'algoritmo *richiama se stesso* su un differente set di dati fino ad una terminazione positiva o negativa
Un algoritmo di questo tipo si dice **ricorsivo**: ovvero il problema viene scomposto in sotto problemi uguali ma su set di dati più piccoli, fino ad arrivare o alla soluzione o alla impossibilità di proseguire.
condizioni di stop:

- il sottoinsieme su cui viene richiamato l'algoritmo è vuoto
- la ricerca è andata a buon fine