

# Problemi, Algoritmi, Tempo e Spazio

## Luigi Salemi

La Complessità Computazionale (1) è quella branca dell'informatica che studia le risorse necessarie, in Tempo e Spazio, per la risoluzione di un Problema. I Problemi sono così riconducibili a differenti Classi di Complessità in funzione del miglior Algoritmo di cui disponiamo per risolverli.

Una definizione formale coinvolge la Macchina di Turing (2) che possiamo immaginare come il prototipo teorico di ogni computer, ma non è questo lo scopo di questa comunicazione che vuole fermarsi ad un livello informale e divulgativo.

Le Classi di Complessità sono una miriade (3) e di tutte si conosce la struttura di inclusione [L'Algoritmo che funziona per una Classe è utilizzabile per ogni Classe di minore Complessità e quindi la seconda è inclusa nella prima], quasi mai si sa se tale inclusione è "stretta" oppure no, ovvero se 2 Classi contigue sono realmente separate perché c'è almeno un Problema contenuto in una ma non nell'altra.

### La ricerca dicotomica

Quando cerchiamo un nome in un elenco telefonico utilizziamo un Algoritmo che si potrebbe riassumere così:

- a) Apriamo l'elenco a metà e se il nome che leggiamo è inferiore (nel tradizionale ordinamento alfabetico) a quello che stiamo cercando buttiamo via la parte sinistra dell'elenco se è maggiore buttiamo via la parte destra
- b) Rieseguiamo quanto al punto a) nella parte residua dell'elenco sino a quando troviamo il nome che stiamo cercando

Tale Algoritmo è molto veloce e richiede un numero di Passi pari a  $\log(n)$  dove  $n$  è il numero degli abbonati [ci si ferma al primo intero  $n$  tale che  $2^n \geq$  numero degli abbonati]. Si dice allora che opera in un Tempo  $O(\log(n))$  e si trova nella Classe di Complessità "L" che potremmo considerare come "Complessità Logaritmica"

### Ordinamento

Perché la ricerca dicotomica funzioni è necessario che l'elenco su cui cerchiamo sia Ordinato, si pone quindi il Problema di trovare opportuni Algoritmi di Ordinamento. C'è tutta una storia, il primo di tali Algoritmi si chiama Bubble Sort e opera in un Tempo  $O(n^2)$  dove  $n$  è il numero degli elementi da riordinare, poi ne è stato trovato un altro denominato Heap Sort (4) che opera in un Tempo  $O(n \log(n))$  e risulta quindi molto più efficiente del precedente. Poi, caso più unico che raro, è stato possibile provare che Heap Sort è il migliore degli Algoritmi possibili per l'Ordinamento e così la ricerca di ulteriori Algoritmi che potessero migliorare i Tempi di esecuzione è cessata.

Comunque il Tempo di esecuzione di entrambi gli Algoritmi è polinomiale [è esprimibile tramite un polinomio in  $n$  di grado  $k$  con  $k$  prefissato, si indica in generale con  $O(n^k)$ ] e quindi questo Problema è comunque codificato nella Classe "P" che potremmo considerare come "Complessità Polinomiale"

E' interessante notare come i 2 Algoritmi precedenti pur trovandosi in Classi distinte rispetto al Tempo siano entrambi polinomiale rispetto allo Spazio e questo fa intravedere come la Complessità rispetto al Tempo sia inclusa nella Complessità rispetto allo Spazio [un Algoritmo che risolve in uno Spazio polinomiale risolve in un Tempo polinomiale, mentre del viceversa non c'è certezza e anzi si pensa sia falso]. La Classe dei Problemi risolti in Spazio polinomiale è la PSPACE, è una Classe molto ampia ed include anche Problemi risolvibili in Tempo Esponenziale [ma verificabili in Tempo polinomiale]

## **Lo Zaino**

Questo Problema ha diverse formulazioni [ed un interesse pratico notevole in molti campi] la più semplice è forse la seguente: dati  $n$  oggetti di diverso Peso ed uno Zaino in grado di sopportare al più Tot chili quali oggetti scegliere per raggiungere, ma non superare, la capacità dello Zaino?

Di fatto non conosciamo al momento nessun Algoritmo che sia sostanzialmente migliore del provare tutte le combinazioni possibili degli oggetti. E questo colloca tale problema in  $O(2^n)$  perché tale è la Potenza dei Sottoinsiemi di un Insieme rendendolo estremamente complesso dal punto di vista Computazionale. Eppure in questo tipo di Problema la verifica di una soluzione (o come spesso si dice di un “certificato”) è polinomiale [se vi si dice di provare gli  $m$  oggetti  $A_1, A_2, A_3, \dots, A_m$  perché questi raggiungono la capacità dello Zaino senza superarla la verifica si fa in tempo  $O(m)$  perché basta sommare i pesi degli  $m$  oggetti].

La Classe di appartenenza è la “NP” ed è anche denominata “la Classe dei Problemi intrattabili”. Per capire il perché di questa definizione supponiamo di avere scritto un programma in grado di eseguire un milione di verifiche al secondo, ovvero sarà in grado di trovare in un solo secondo la soluzione qualora gli oggetti siano 20 [ $2^{20} \sim 1.000.000$ ]. Quanto Tempo impiegherà lo stesso programma per trovare la soluzione qualora gli oggetti siano 80? La incredibile risposta è che ci vorranno 33 miliardi di anni e questo vuol dire 2 volte l’età dell’universo. Più intrattabili di così ..

Ciò che scoccia maggiormente e che tanti Problemi con grande interesse pratico [crittografia tradizione, percorsi minimi, ecc.] sono in questa Classe. Al momento la miglior cosa che si possa fare [visto che non sempre disponiamo di tanto tempo per avere la risposta] è utilizzare Algoritmi probabilistici, ovvero Algoritmi che “probabilmente” trovano la miglior soluzione in un Tempo polinomiale.

## **Gli Scacchi**

C’è di peggio, per gli Scacchi i Tempi non solo sono esponenziali nella ricerca della soluzione, ma lo sono anche nella verifica. (5) Se avete mai provato a trovare la sequenza delle mosse di un Problema del tipo “Il Bianco muove è matta in 2 mosse” sapete di cosa sto parlando. Tali problemi sono nella Classe “EXPTIME”.

## **La Aritmetica non moltiplicativa**

Solo per curiosità: la verifica (6) di un’affermazione sulla Aritmetica dei numeri reali che coinvolga la sola operazione di addizione e nella Classe “EXPSPACE” [necessita di Spazi esponenziali]

Si sa che  $L \subseteq P \subseteq NP \subseteq PSpace \subseteq ExpTime \subseteq ExpSpace$ . Ciò che non si sa e se l’inclusione è “stretta” oppure no. Le uniche cose che si sanno, al momento, tra quelle riportate e che  $L \subset PSpace$ ,  $P \subset ExpTime$  e  $PSpace \subset ExpSpace$ .

In particolare ciò che maggiormente vorremmo sapere è se  $P$  uguale  $NP$  [esiste un Algoritmo che ancora non conosciamo che consente di risolvere tutti i Problemi di  $NP$  in un Tempo polinomiale] oppure  $P$  diverso da  $NP$ . [almeno un Problema della Classe  $NP$  mai si potrà risolvere in Tempo polinomiale].

## **P diverso da NP**

Questa è la ipotesi più accreditata dagli esperti. E’ notizia di questi giorni che il ricercatore della HP Vinay Deolalikar ha annunciato di aver trovato tale dimostrazione, il suo lavoro è al vaglio della comunità scientifica internazionale, ma su questo pesa fortemente il commento di Terence Tao, uno degli scienziati più rinomati, che ha sostanzialmente detto: “La strada intrapresa da Deolalikar difficilmente potrà dimostrare questo risultato”.

## **P uguale NP**

Nel 1971 sembrava fatta. Cook e Levin, separatamente, hanno provato che tutti i problemi NP sono riconducibili ad un unico problema denominato SAT (7). Risolto con un Algoritmo polinomiale questo risolti tutti. L'anno dopo Karp ha provato che altri 21 problemi (8) godevano di questa caratteristica: basta risolverne uno con un Algoritmo polinomiale perché siano automaticamente risolti tutti i Problemi della Classe NP. Ad oggi questi Problemi "Duri" sono oltre 3.000 e basta risolverne uno con un Algoritmo polinomiale perché la Classe NP collassi nella Classe P.

Penso di aver trovato tale prova per il Problema denominato "3Sat" che è uno dei 21 della lista di Karp. Tale dimostrazione è al vaglio di una piccola parte della comunità scientifica nazionale [beh, si fa quel che si può].

## **P verso NP non decidibile**

Non stupirebbe. Fin dal 1931, quando Kurt Gödel ha provato il famoso "Teorema di Incompletezza" che sancisce l'esistenza di questioni indecidibili in Sistemi con Potenza pari o superiore all'Aritmetica, ci aspettiamo che da un momento all'altro salti fuori uno di questi Problemi. Inoltre si spiegherebbe perché, nonostante la soluzione sia cercata da tempo, al momento abbiamo solo ipotesi e nessuna prova conclamata.

Gödel comunque ha a che fare con questa questione molto più di quanto sembri. Nel 1950 scrisse al suo amico von Neumann una lettera privatissima in cui chiedeva notizie sullo stato di salute e gli augurava una pronta guarigione. Ma si sa come sono fatti i geni, così tra i saluti e gli abbracci trovo il modo di scrivere "Non sarei stupito se si riuscisse a provare che P coincide con NP" [libera sintesi di oltre 20 righe]. Su Gödel è stato detto tanto, non a caso è considerato il miglior logico del nostro tempo; ma ciò che meglio esprime l'ammirazione verso questo grande matematico è la seguente frase [di cui sconosco l'autore] "Per trovare un logico a lui comparabile bisogna risalire ad Aristotele". Ecco allora che la affermazione su P e NP deve essere trattata con il dovuto rispetto.

Catania 25 Settembre 2010

Luigi Salemi

## Referenze sul Web

- 1) [http://it.wikipedia.org/wiki/Teoria\\_della\\_complessit%C3%A0\\_computazionale](http://it.wikipedia.org/wiki/Teoria_della_complessit%C3%A0_computazionale)
- 2) [http://it.wikipedia.org/wiki/Macchina\\_di\\_Turing](http://it.wikipedia.org/wiki/Macchina_di_Turing)
- 3) [http://it.wikipedia.org/wiki/Classi\\_di\\_complessit%C3%A0\\_P\\_e\\_NP](http://it.wikipedia.org/wiki/Classi_di_complessit%C3%A0_P_e_NP)
- 4) [http://it.wikipedia.org/wiki/Heap\\_sort](http://it.wikipedia.org/wiki/Heap_sort)
- 5) <http://en.wikipedia.org/wiki/EXPTIME>
- 6) <http://en.wikipedia.org/wiki/EXPSPACE>
- 7) [http://it.wikipedia.org/wiki/Soddisfacibilit%C3%A0\\_booleana](http://it.wikipedia.org/wiki/Soddisfacibilit%C3%A0_booleana)
- 8) [http://it.wikipedia.org/wiki/Richard\\_Karp](http://it.wikipedia.org/wiki/Richard_Karp)