15 Teoria della calcolabilità

Si occupa delle questioni fondamentali circa la **potenza** e le **limitazioni** dei sistemi di calcolo. L'origine risale alla prima metà del ventesimo secolo, quando i logici matematici iniziarono ad esplorare i concetti di:

- Computazione
- Algoritmo
- Problema risolvibile per via algoritmica

e dimostrano l'esistenza di problemi che non ammettono un algoritmo di risoluzione.

Definizione 15.1 (Problemi computazionali)**.** *Problemi formulati **matematicamente** di cui cerchiamo una soluzione algoritmica. Si classificano in:*

- *Problemi **non decidibili**, non ammettono un algoritmo di risoluzione*
- *Problemi **decidibili**, che a loro volta possono essere:*
  - ***Trattabili**, ovvero di costo polinomiale*
  - ***Non trattabili**, ovvero di costo esponenziale*

Facciamo ora la distinzione tra:

- **Calcolabilità**: sfrutta le nozioni di *algoritmo* e di *problema non decidibile*. Ha lo scopo di classificare i problemi in risolvibili e non risolvibili.
- **Complessità**: sfrutta le nozioni di *algoritmo efficiente* e di *problema non trattabile*. Ha lo scopo di classificare i problemi in "facili" e "difficili".

15.1 Problemi indecidibili

15.2 Problemi decidibili ma intrattabili

Esempio 15.1 (Torre di Hanoi)**.**

```
TorriHanoi(n, p, t, s)
        if(n==1) print p->t;
        else{
          TorriHanoi(n-1, p, s, t);
          print p->t;
          TorriHanoi(n-1, s, t, p);
        }
```

Listing 34: Torre di Hanoi

Scriviamo la relazione di ricorrenza:

$$\begin{cases} 1 & n = 1 \\ 2M(n-1)+1 & n > 1 \end{cases}$$

Non essendo risolvibile con il Master's Theorem, proviamo con il metodo di sostituzione: Dalla sosti-

n	1	2	3	4	...	i

tuzione sembra che $M(n) = 2^n - 1$. Dimostriamolo per induzione su n:

- **Caso base**: per $n = 1$, $M(1) = 1$ e $2^1 - 1 = 2 - 1 = 1$
- **Ipotesi induttiva**: $M(i) = 2^i - 1$, $\forall i < n$
- **Passo induttivo**: per $n > 1$, $M(n) = 2M(n-1) + 1 = 2(2^n - 1) + 1 = 2^{n+1} - 1$

Abbiamo quindi dimostrato che Ora vogliamo capire qual'è il numero di mosse **necessarie**

La conclusione è che l'algoritmo ricorsivo dimostrato in precedenza è **ottimo** in quanto $2^n - 1$ mosse sono **necessarie** e **sufficienti**.
Se i monaci spostassero 1 disco al secondo, per spostare i 64 dischi ci vorrebbero comunque $2^{64} - 1$ secondi, ovvero circa $585 \cdot 10^9$ anni.
È quindi lecito dire che un problema con soluzione esponenziale è spesso a livello pratico assimilabile ad un problema indecidibile.

Esempio 15.2 (Generazione delle sequenze binarie)**.** Dato $A = \{a_0, a_1, \dots, a_{n-1}\}$ insieme di n oggetti. Il numero di sottoinsiemi di A è 2^n in quanto lo possiamo descivere con sequenze binarie. Ad esempio dato $A' \subseteq A$ tale che $A' = \{a_0, a_3, a_5, a_6\}$:

$$\Big\{ 2$$