

Introduzione agli algoritmi

Algoritmo: Procedimento che descrive una sequenza di passi ben definiti finalizzato a risolvere un dato problema (computazionale).

Il termine Algoritmo deriva da Algorismus, traslitterazione latina del nome di un matematico persiano del IX secolo, Muhammad al-Khwarizmi, che descrisse delle procedure per i calcoli matematici.

Ci sta una netta differenza tra algoritmo e programma, nonostante siano concetti collegati:

- Un algoritmo può essere visto come l'essenza computazionale di un programma, cioè il **procedimento per raggiungere alla soluzione**.
- Il **programma** è la **codifica di un algoritmo**.
- **Algoritmo è un concetto autonomo da programma**.

Durante l'anno studieremo come analizzare e progettare **buoni** algoritmi, cioè:

- **Corretti:** Producono il risultato esatto.
- **Efficienti:** Usano poche risorse di calcolo (Tempo e Memoria)

Se un algoritmo ha entrambe le caratteristiche allora è un **algoritmo veloce**.

Esistono però delle caratteristiche importanti quanto l'efficienza durante la progettazione di un software:

- Correttezza;
- Mantenibilità;
- Modularità;
- User-friendly;
- Sicurezza;

Ma perché allora è così importante l'efficienza? Perché :

- Legato alla User-friendliness;
- Veloce è comodo;
- Ci sono situazioni in cui è necessario avere un algoritmo veloce, altrimenti non funziona il programma.
- Possiamo usare l'efficienza per "pagare" lacune.

Ogni algoritmo è composto da due parti:

- Identificazione della appropriata **tecnica di progetto algoritmico**.
 - Chiave del **nucleo matematico del problema stesso**.
-

Esercizio fatto in classe

Zio Paperone ha n monete, fra queste monete ci sta una moneta falsa che pesa leggermente di più, abbiamo una bilancia a due piani, dobbiamo cercare quella falsa.

Innanzitutto introduciamo dei concetti importanti per la materia con degli esempi riguardante il problema:

- **PROBLEMA**: Individuare una moneta falsa tra n monete.
- **ISTANZE**: n specifiche monete e una moneta falsa, cioè un **input specifico del problema**.
- **DIMENSIONE DELL'ISTANZA**: il **valore di n** .
- **MODELLO DI CALCOLO**: bilancia a due piatti, **specifica quali operazioni si possono eseguire**.
- **ALGORITMO**: Strategia di pesata, **descrizione generale della sequenza di operazioni sul modello di calcolo**.
- **CORRETTEZZA DELL'ALGORITMO**: La strategia di pesatura deve essere corretta per ogni istanza.
- **COMPLESSITA TEMPORALE**: Numero di pesate prima di individuare la moneta falsa, **dipende dall'istanza e dalla dimensione dell'istanza e quantifica la quantità di tempo impiegata da un algoritmo a essere eseguito**.
- **COMPLESSITA TEMPORALE NEL CASO PEGGIORE**: Massimo di pesate che esegue su una istanza di una certa dimensione, **situazioni peggiori su una istanza particolare**.
- **EFFICIENTE**: l'algoritmo deve fare poche pesate, così è veloce, ma rispetto a cosa?

Dobbiamo avere una **procedura che risolva il problema indipendentemente dalla dimensione dell'istanza**.

Esistono diverse soluzioni:

Algoritmo 1: Uso la prima moneta e la confronto con le altre, In questo modo eventualmente troverò una moneta che pesa di più rispetto a quella di riferimento.

Corretto: **SI**

Il numero di pesate: dipende
nel caso peggiore: $n-1$.

E' efficiente? non si sa **dipende**

La domanda giusta è **Posso fare meglio?**

Pseudocodice:

```
Alg1 (X=[x1 , x2 , ..., xn ]) 1. for i=2 to n do  
2. if peso(x_1 ) > peso(x_i ) then return x_1 3. if peso(x_1 ) < peso(x_i ) then return  
x_i`
```

Per descrivere un algoritmo in senso qualitativo utilizziamo uno **pseudocodice che non tiene conto dei dettagli legati a specifici linguaggi**. I costrutti sono sempre gli stessi;

Sequenziamento, condizionale, ciclo.

Algoritmo 2: Utilizziamo una strategia diversa, pesiamo le monete a coppie distinte due a due.

Corretto: SI,

Quante pesate: dipende,

Nel **caso peggiore?** $\lfloor n/2 \rfloor$,

E' efficiente: Boo;

Meglio dell'algoritmo 1, posso fare meglio?

Algoritmo 3: Peso le monete dividendole ogni volta in due gruppi delle stesse dimensione.

Corretto: SI;

Numero pesate caso peggiore? $\lfloor \log_2(n) \rfloor$ (da argomentare);

Efficiente? bo;

Meglio dell'algoritmo 2.

ANALISI COMPLESSITA' algoritmo 3:

$P(n)$ = numero pesate che algoritmo esegue nel caso peggiore su istanza di dimensione n .

$$P(n) = P(\lfloor n/2 \rfloor) + 1$$

$$P(1) = 0$$

Questa è un **Equazione di ricorrenza**: La tecnica migliore per risolverla è lo *srotolamento*

E' una tecnica che impareremo a utilizzare nelle prossime lezioni.


Algoritmo 4: Dividiamo in tre gruppi invece di due.

Corretto: SI;

Nel caso peggiore $\lfloor \log_3(n) \rfloor$ (Da argomentare);

Meglio dell'algoritmo 3.

Potremmo dividere per 4, ma non aiuta, anzi peggiora la situazione.

 **Teorema:** Un qualsiasi algoritmo che correttamente individua la moneta falsa tra n monete deve effettuare nel caso peggiore almeno $\lceil \log_3(n) \rceil$.

Una dimostrazione di questo teorema usa la tecnica dell'**albero di precisione** di un problema.

 **Corollario:** L'algoritmo 4 è un algoritmo **ottimo** per il problema.