U14-2040 I piano per ricevimenti su appuntamento
o da remoto

claudia.zaudron@unimib.it

indicare nome cognome matricola

controllare bene sui siti prima

17/04 circa primo parziale

Giugno il secondo (il parziale é la stessa data del completo)

Luglio recupero (stessa data secondo appello)

Esame

saper scrivere algo. valutarlo in tempo e spazio

Saper scrivere un algoritmo corretto più veloce e che occupi meno memoria possibile

Programma

Algo

- ordinamenti (intuitivi)
  - ricorsione
  - divide et impera
- ord. rapidi

Str. Dati

- liste dinamiche
- pile e code
- alberi binari
- grafi -> BFS, DFS

Cos'é un algoritmo ?

↓

Sequenza di istruzioni precise che risolvono un problema computazionale agendo su degli input e fornendo degli output.
Le istruzioni sono elementari ed eseguibili singolarmente.

Problema computazionale

cosa ci serve definire ?

1. Descrivo parametri per input e output
2. Descrivo relazione tra input e output
3. Istanza, specifici valori di input (esempi)

Esempio ordinamento di un vettore di interi

1. input

proposte:

- vettore non ordinato ✗ potrei avere un vettore già ordinato
- seq. di numeri ✗ potrei dover ordinare altro non solo numeri
- array ✗ potrei usare altre strutture
- str. dati ✗ troppo generico

ho bisogno di essere più preciso nella richiesta

- vettore di interi di lunghezza n ✓

output

- vettore di interi di lunghezza n
2. con gli stessi element: dell'input e ordinato

3. istanza: input = 1, 4, 6, 3
output = 1, 3, 4, 6

Come scelgo quale algoritmo usare?

Un problema può essere risolto con vari algoritmi (in teoria infiniti)

requisiti:

① Deve essere **corretto**

> la relazione tra input e output deve essere rispettata per ogni istanza

Scrivere algo corretti è molto complicato, si cerca di ridurre i casi in cui non funziona.

② Valutare in termini di **tempo** e **spazio**

> Non dire il tempo ma quante operazioni ci vogliono

Nell'esempio dell'ordinamento anche se ho la stessa quantità di numeri ho tempi diversi con lo stesso algoritmo.
A parità di elementi ho due limiti:

fissata generica dimensione (n)

peggiore — t_p — dipende dall'algoritmo

t_{medio} — il tempo di esecuzione è in funzione di (n)

migliore — già ordinato — t_m

(quello in cui devo fare meno op.)

esempio:

$T_p(n) = 10n + 7$

$T_{medio}(n) = 5n$ è quello più frequente

$T_m(n) = n + 3$ è quello che mi aspetto

ricerca in array
migliore: primo peggiore: non c'è

medio: tempo a metà

NON È SEMPRE COSÍ
molto spesso si avvicina al peggiore

10^6 op. al secondo

		n = 20	n = 100	n = 1000	
1)	$100.000\,n$	2"	10"	100"	migliore
2)	$10\,n^3$	1"	20"	2,7 ore	
3)	2^n	~ 0,9"	$4 \cdot 10^{16}$ anni	secoli e secoli	

↑ tempo esponenziale.
Nella stra grande maggioranza dei casi è inutilizzabile

$100.000\,n^2 + 700n + 400$

$10n^3$ guardo solo questi (come confronto fra infiniti)

l'obiettivo è scrivere algo con tempi polinomiali

Valutare tempi di esecuzione

```
int RS (int V[], int K)
```

↑ ricerca sequenziale ↖ valore che devo trovare

Non metterei return e break nei cicli

Non forzare le variabili per uscire dai cicli

```
C1 · 1      pos = 1          // per noi array partono da 1
C2 · t_w+1  while ( V[pos] ≠ K) AND (pos ≤ length(V))
C3 · t_w        pos++
C4 · 1      if pos > length(V)
C5 T_if             return -1
C6 F_if     else    return pos
```

quante volte è testato

Array n elementi

una istruzione richiede un tempo C_i per l'exe.

$$t_{RS}(n) = C_1 + C_2 \cdot (t_w + 1) + C_3(t_w) + C_4 + C_5 T_{if} + C_6 F_{if}$$

verranno sempre eseguiti a prescindere dall'input

Così non riesco a valutare, devo fare il caso peggiore e migliore.

Il caso migliore è quello in cui vengono eseguite il minor numero possibile di istruzioni.

Il peggiore il maggior numero di istruzioni.

Qual è il più piccolo tw che posso avere?

$t_w = 0$ qual'è input che porta a sto caso?

V[1] = K e viene automaticamente settato Fif

Fif

caso migliore $t_w = 0 \rightarrow V[1] = K$ trovo K in prima pos. dell'array

$t_m(n) = c_1 + c_2 + c_4 + c_6 \approx 4$ c_3 e c_5 non vengono mai fatta

è una funzione di n ma non compare perché se trovo nella prima pos non importa la lunghezza

caso peggiore $t_w = n \rightarrow$ K non fa parte di V

il max di volte che controllo il ciclo

$$t_p(n) = c_1 + c_2(n+1) + c_3 n + c_4 + c_5 \approx$$

$$(c_2 + c_3)n + (c_1 + c_2 + c_4 + c_5) \approx (n)$$

tempo medio

$t_{medio}(n)$ $t_w = \frac{n}{2}$

Se l'array è ordinato?

```
int RS (int V[], int K)
  pos - 1     // per noi array partono da 1
  while ( V[pos] < K) AND (pos <= length(V)){
      pos++
  }
  if pos > length(V)
        return -1
  else
        return pos
```

cambia tutto cambia il caso migliore, ha lo stesso tempo ma è più probabile che si verifichi

caso peggiore, non trovo il numero e tutti gli elementi sono minori di K. È molto meno probabile che accada.

Cambia dunque il tempo medio