Corso di laurea in Informatica
Introduzione agli Algoritmi
Lezioni in modalità mista o a distanza
A.A. 2022/23

Benvenuti!

SAPIENZA
UNIVERSITÀ DI ROMA

Corso di laurea in Informatica
Introduzione agli Algoritmi
Lezioni in modalità mista o a distanza
A.A. 2022/23

Syllabus e Introduzione

Tiziana Calamoneri

SAPIENZA
UNIVERSITÀ DI ROMA

Slides realizzate sulla base di quelle preparate da T. Calamoneri e G. Bongiovanni per il corso di Informatica Generale tenuto a distanza nell'A.A. 2019/20

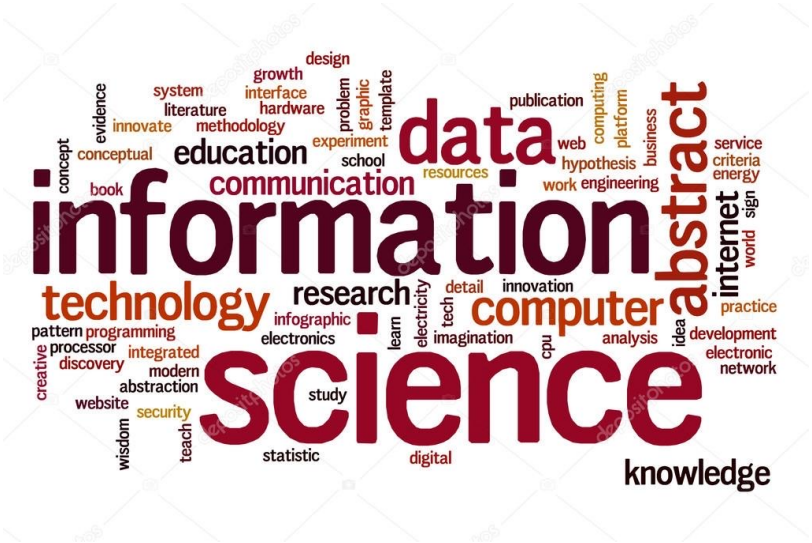
Di cosa tratta questo corso

Algoritmi (1)

Cominciamo dal principio:

La definizione di informatica proposta dall'ACM (**Association for Computing Machinery**) è la seguente:

"***L'informatica è la scienza degli algoritmi che descrivono e trasformano l'informazione: la loro teoria, analisi, progetto, efficienza, realizzazione e applicazione.***"



" Quello di ***algoritmo,*** dunque, è un concetto fondamentale, centrale per l'informatica.

Algoritmi (2)

Un algoritmo è "***una sequenza di comandi elementari ed univoci che terminano in un tempo finito ed operano su strutture dati***".

Un comando è **elementare** quando non può essere scomposto in comandi più semplici; è **univoco** quando può essere interpretato in un solo modo.

Algoritmi (3)

Se un algoritmo è ben specificato, chi (o ciò che) lo esegue non ha bisogno di pensare, deve solo eseguire con precisione i passi elencati nell'algoritmo, nella sequenza in cui appaiono.

E infatti un calcolatore non pensa, esegue pedissequamente tutte le operazioni elencate negli algoritmi pensati (ossia progettati) da un essere umano.

Se si verifica un errore e il risultato è sbagliato, l'errore non è del calcolatore ma di chi ha progettato l'algoritmo!

Strutture dati (1)

Un algoritmo è "***una sequenza di comandi elementari ed univoci che terminano in un tempo finito ed operano su strutture dati***".

Per risolvere i problemi abbiamo, ovviamente, bisogno di gestire i relativi dati. A tal fine dovremo definire le opportune ***strutture dati***: strumenti necessari per organizzare e memorizzare i dati veri e propri, semplificandone l'accesso e la modifica.

Strutture dati (2)

Non esiste una struttura dati che sia adeguata per ogni problema, per cui è necessario conoscere proprietà, vantaggi e svantaggi delle principali strutture dati in modo da poter scegliere di volta in volta quale sia quella più adatta (o più facilmente adattabile) al problema.

Il progetto o la scelta della struttura dati da adottare nella soluzione di un problema è un aspetto fondamentale per la risoluzione del problema stesso, al pari del progetto dell'algoritmo.
Perciò, gli algoritmi e le strutture dati fondamentali vengono sempre **studiati e illustrati assieme**.

Efficienza (1)

Un algoritmo è "***una sequenza di comandi elementari ed univoci che terminano in un tempo finito ed operano su strutture dati***".

Affinché un algoritmo sia utilizzabile, deve concludersi e produrre il suo output entro un tempo "ragionevole".

Un aspetto fondamentale che va affrontato nello studio degli algoritmi è la loro **efficienza**, cioè la ***quantificazione delle loro esigenze in termini di tempo e di spazio***, ossia tempo di esecuzione e quantità di memoria richiesta.

Efficienza (2)

Questo perché:

- I calcolatori sono molto veloci, ma non infinitamente veloci;
- La memoria è economica e abbondante, ma non è né gratuita né illimitata.

Nel corso illustreremo il concetto di

costo computazionale

degli algoritmi, in termini di numero di operazioni elementari e quantità di spazio di memoria necessari in funzione della dimensione dell'input.

Efficienza (3)

Esempio:

Problema: ordinare $n=10^6$ numeri interi;

- Il calcolatore ***V*** (veloce) effettuare 10^9 operazioni/ sec
- Il calcolatore ***L*** (lento) effettua 10^7 operazioni/ sec

- L'algoritmo ***IS*** (Insertion Sort) richiede $2\,n^2$ operazioni;
- L'algoritmo ***MS*** (Merge Sort) richiede $50\ n \log n$ operazioni.

Efficienza (4)

Esempio (segue)

La maggiore velocità di V riesce a bilanciare la minore efficienza dell'algoritmo IS?
Confrontiamo il tempo di esecuzione di IS sul calcolatore V con quello di MS sul calcolatore L.

Tempo di V(IS) = $\frac{2(10^6)^2 istruzioni}{10^9 istruzioni\ al\ secondo} = 2000\ secondi = \mathbf{33\ minuti}$

Tempo di L(MS) = $\frac{50*10^6\ log\ 10^6\ istruzioni}{10^7 istruzioni\ al\ secondo} = 100\ secondi = \mathbf{1,5\ minuti}$

Cioè, **la risposta è no**.

Efficienza (5)

Esempio (segue):

Aumentiamo la dimensione dell'input, portandola da ***10^6*** a ***10^7***:

Tempo di V(IS) = $\frac{2(10^7)^2 istruzioni}{10^9 istruzioni\ al\ secondo} = \mathbf{2\ giorni}$

Tempo di L(MS) = $\frac{50*10^7\ log\ 10^7\ istruzioni}{10^7 istruzioni\ al\ secondo} = \mathbf{20\ minuti}$

Cioè, indipendentemente dall'aumento di velocità dei calcolatori prodotto dagli avanzamenti tecnologici, **l'efficienza degli algoritmi è un fattore di importanza cruciale**.

Problem solving (1)

Il ***problem solving*** è un'attività che ha lo scopo di raggiungere una soluzione a partire da una situazione iniziale.

E' quindi un'attività creativa, di natura essenzialmente progettuale, ed in questo risiede la sua difficoltà.

Problem solving (2)

Approccio al problem solving:

- ***analisi del problema**:* lettura approfondita della situazione iniziale, comprensione ed identificazione del problema;
- ***esplorazione degli approcci possibili**:* identificazione delle metodologie di soluzione tra i metodi noti;
- ***selezione di un approccio**:* scelta dell'approccio migliore;
- ***definizione dell'algoritmo risolutivo**:* identificazione dei dati e progettazione della sequenza di passi elementari da applicare su di essi;
- ***riflessione critica**:* a problema risolto, ripensamento delle fasi della soluzione proposta per identificare eventuali criticità e possibili migliorie.

Problem solving (3)

Quali problemi? 🤔

In questo corso restringiamo la nostra attenzione ai ***problemi computazionali***, problemi cioè che richiedono di descrivere in modo automatico una specifica relazione tra un insieme di valori in ***input*** e il corrispondente insieme di valori in ***output***.

Un algoritmo è ***corretto*** se, ***per ogni istanza di un problema computazionale, termina producendo l'output corretto***.

In tal caso diremo che ***l'algoritmo risolve il problema***.

Problem solving (4)

Esempio di problema computazionale:

Definizione del problema:

Ordinare *n* numeri dal più piccolo al più grande.

Input (anche detto ***istanza del problema***):

Sequenza di *n* numeri $a_1, a_2, \ldots, a_n$;

Output:

Permutazione $a'_1, a'_2, \ldots, a'_n$ della sequenza di input t.c.

$$a'_1 \leq a'_2, \ldots, \leq a'_n.$$

Random access machine (1)

Per poter valutare l'efficienza di un algoritmo è necessario **analizzarlo**, cioè quantificare le risorse che esso richiede per la sua esecuzione, **senza che tale analisi sia influenzata da una specifica tecnologia** che, inevitabilmente, col tempo diviene superata.

Si usa la ***Random Access Machine*** (***modello RAM***) che è **indipendente dalle caratteristiche tecniche di uno specifico calcolatore reale**.

La RAM è quindi una ***macchina astratta***, la cui validità e potenza concettuale risiede nel fatto che non diventa obsoleta con il progredire della tecnologia.

Random access machine (2)

Nel modello RAM:

- esiste un **singolo processore**, che esegue le operazioni sequenzialmente;
- esistono delle **operazioni elementari**, l'esecuzione di ciascuna delle quali richiede per definizione un ***tempo costante***. (Es.: operazioni aritmetiche, letture, scritture, salto condizionato, ecc.);
- esiste un ***limite*** alla **dimensione di ogni valore memorizzato** ed al **numero complessivo di valori utilizzati** (il max valore rappresentabile in memoria non può superare *2 elevato al numero di bit della parola (32 o 64)*).

Criterio della misura di costo uniforme (1)

Se è soddisfatta l'ipotesi che ogni dato in input sia minore di un valore

$$k = 2^{\text{numero di bit della parola di memoria}}$$

ciascuna operazione elementare sui dati del problema verrà eseguita in un tempo costante.

In tal caso si parla di ***misura di costo uniforme***.

Criterio della misura di costo uniforme (2)

Tale criterio **non è sempre realistico** perché, se un dato del problema è **più grande di k**, esso deve comunque essere memorizzato, ed in tal caso si useranno più parole di memoria.

Di conseguenza, anche le operazioni elementari su di esso dovranno essere reiterate per tutte le parole di memoria che lo contengono, e quindi richiederanno un tempo che non è più costante: Calcolo scientifico e misura di costo logaritmico (non sarà affrontato qui).

Criterio della misura di costo uniforme (3)

Esempio:

```
def PotenzaDi2(n)
x = 1
for i in range(n):
    x = x*2
return x
```

(ciclo, reiterato *n* volte, che calcola il valore 2^n)

Il tempo di esecuzione totale è ***proporzionale ad n***:

- si tratta di un **ciclo eseguito *n* volte**;
- ad ogni iterazione del ciclo si compiono **due operazioni**, ciascuna delle quali ha costo unitario:
  1. l'incremento del contatore
  2. il calcolo del nuovo valore di *x*.

Pseudocodice

Affinché tutti siamo in grado di comprendere un algoritmo, è necessario utilizzare una descrizione:

- il più formale possibile
- indipendente dal linguaggio che si intende usare

↓

Pseudocodice

(di cui riparleremo…)

Dettagli per studiare

Pagina web del corso

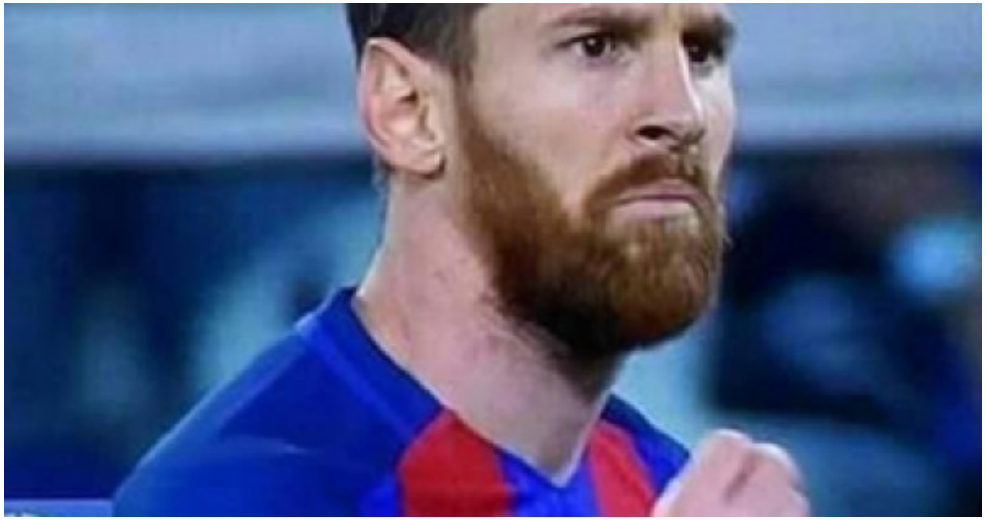
Alla pagina:

https://twiki.di.uniroma1.it/twiki/view/Intro_algo/AD/WebHome

troverete:

- Il programma del corso
- Il diario delle lezioni
- Le modalità d'esame
- Un elenco di libri di testo utili
- Delle dispense e degli esercizi svolti

Un suggerimento



Non basta essere solo “spettatori”:

Alla fine di (quasi) ogni lezione troverete degli esercizi che vengono proposti.

Provate a risolverli e, se non ci riuscite:

- approfondite ulteriormente l’argomento trattato
- chiedete spiegazioni