

Università di Pisa

Antonio Virdis

Algoritmi e strutture dati

a.a. 2021/2022

Docente

Antonio Virdis – Ricercatore presso il Dipartimento di Ingegneria dell'Informazione
antonio.virdis@unipi.it

Didattica

Laurea Triennale in Ingegneria Informatica

- Algoritmi e strutture dati (1° anno)

Master Degree Computer Engineering

- Performance evaluation of computer systems and networks (1° anno)
- Advanced network architectures and wireless systems (2° anno)

Ricerca

Reti di calcolatori

- Edge computing
- Computer-networks Simulation
- Wireless networks

Materiale

Testi di riferimento:

- **Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein**
«INTRODUZIONE AGLI ALGORITMI E STRUTTURE DATI 3/ED»
- **Camil Demetrescu, Irene Finocchi, Giuseppe F. Italiano** «ALGORITMI E STRUTTURE DATI 2/ED»

Altro materiale:

Dispensa De Francesco – Martini (condivisa su Teams)

Slides fornite dal docente (su Teams)

in parte basate sulle slides della Prof. De Francesco

Contenuti

- Lezioni teoriche
- Lezioni pratiche (tramite VM su portatile) 
- Esercizi assegnati per casa

Pre Requisiti

- Fondamenti di Programmazione
- Utilizzo compilatore 
- Comandi base unix 

Esame Finale

- Prova in laboratorio
- Verificare abilitá di progettazione e programmazione

“Ma io so già programmare!”



Sia dato un array contenente delle frasi.

Scrivere un programma che prenda in input una stringa e restituisca tutte le frasi che la contengono.

Realizzare un motore di ricerca che abbia complessità $O(\log n)$ sulle operazioni di lettura.

Algoritmi e Strutture Dati

Efficienza di un programma...senza scrivere un programma

Nozione di Algoritmo

Origine del nome «algoritmo»



Muhammad ibn Musa al-Khwarizmi
matematico arabo dell'800 (libri scritti **813-830** circa)

notazione posizionale dei numeri e lo 0
(*Algoritmi de numero Indorum*) - algebra

Il nome algoritmo con il significato attuale
viene usato dal XIX secolo

Prime caratterizzazioni di algoritmo e computer

Alan Turing : Macchina di Turing (1936)

Alonso Church: lambda-calcolo (1936)

Concetto di algoritmo

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

- **insieme finito di istruzioni teso a risolvere un problema**
- **Input e Output**
- **ogni istruzione deve essere ben definita ed eseguibile in un tempo finito da un agente di calcolo**
- **E' possibile utilizzare una memoria per i risultati intermedi**

Primi algoritmi nella storia

Algoritmi aritmetici

Babilonesi (circa 2500 a.c.)

Egizi (circa 1500 a.c.)

Greci:

algoritmo di Euclide (300 a.c.)

setaccio di Eratostene (sieve, crivello) (I secolo d.c.)

Algoritmi e programmi

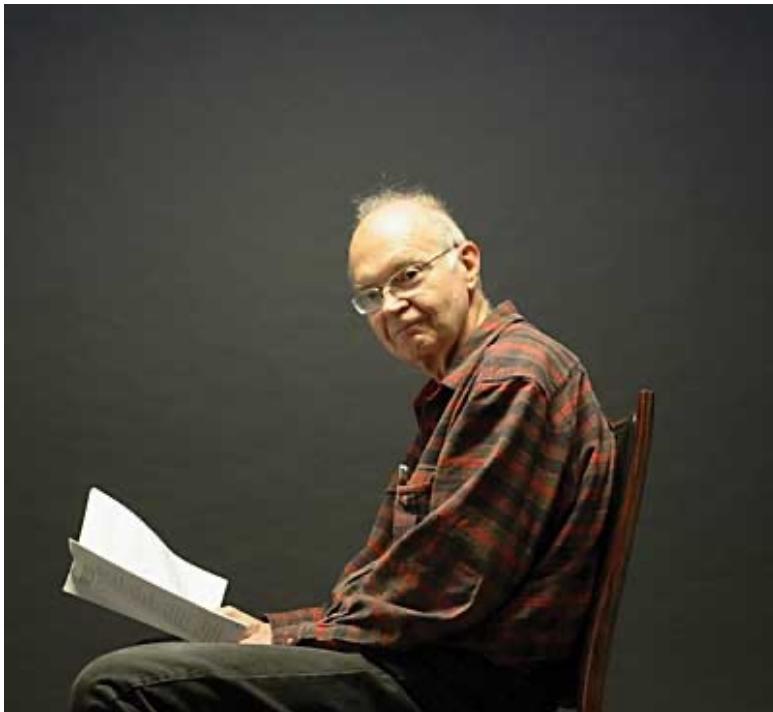
Un algoritmo può essere visto come **l'essenza computazionale** di un programma, nel senso che fornisce il procedimento per giungere alla soluzione di un dato problema di calcolo

~~Algoritmo diverso da programma~~

- **programma è la codifica (in un linguaggio di programmazione) di un algoritmo**
- **un algoritmo può essere visto come un programma distillato da dettagli riguardanti il linguaggio di programmazione, ambiente di sviluppo, sistema operativo**
- **Algoritmo è un concetto autonomo da quello di programma**

Perché studiare gli algoritmi

importanza teorica



“Se è vero che un problema non si capisce a fondo finché non lo si deve insegnare a qualcuno altro, a maggior ragione nulla è compreso in modo più approfondito di ciò che si deve insegnare ad una macchina, ovvero di ciò che va espresso tramite un algoritmo.”

Donald Knuth

Perché studiare gli algoritmi (2)

importanza pratica



"Esiste un detto: se vuoi diventare un buon programmatore, devi solo programmare tutti i giorni per due anni, e diventerai un programmatore eccellente.

Se vuoi diventare un programmatore di classe mondiale, hai due scelte:

- 1) programmare ogni giorno per 10 anni
- 2) oppure programmare ogni giorno per due anni e seguire un corso di algoritmi"

Charles E. Leiserson

Perché studiare gli algoritmi (3)

cosa sapere per lavorare a Google ?

This guide is intended to help you prepare for Google Technical Practicum interviews at Google. If you have any additional questions, please reach out to your recruiter.

Recruitment Process: Software Engineering Internships

Topics

- Maps/Dictionaries
- Linked Lists
- Trees
- Stacks & Queues
- Heaps
- Graphs
- Runtime Analysis
- Searching & Sorting
- Recursion & DP

Extra Prep Resources

Confidential + Proprietary

Algoritmo di Euclide

Trovare il Massimo Comun Divisore fra due numeri interi non negativi

$$\text{MCD}(30, 21) = 3$$

TH: il MCD fra due numeri è uguale al MCD fra il più piccolo e la differenza fra i due

```
1 int MCD(int x, int y) {  
2     while (x!=y)  
3     {  
4         if (x < y) y=y-x;  
5         else x=x-y;  
6     }  
7     return x;  
8 }
```

$$x = 30, y = 21$$

$$x = 9, y = 21$$

$$x = 9, y = 12$$

$$x = 9, y = 3$$

$$x = 6, y = 3$$

$$x = 3, y = 3$$

Algoritmo di Euclide con il resto della divisione

TH: il MCD fra due numeri x e y è uguale al MCD fra y e il resto della divisione fra x e y

```
1 int MCD(int x, int y)
2 {
3     while (y != 0)
4     {
5         int k=x;
6         x=y;
7         y=k%y;
8     }
9     return x;
}
```

$x = 30, y = 21$

$x = 21, y = 9$

$x = 9, y = 3$

$x = 3, y = 3$

$x = 3, y = 0$

$\rightarrow \text{MCD}(4022, 270)$



Numeri primi

Trovare tutti i numeri primi fino a un dato numero n

Algoritmo inefficiente: dividere ogni numero minore o uguale a n per tutti i suoi predecessori

brute force

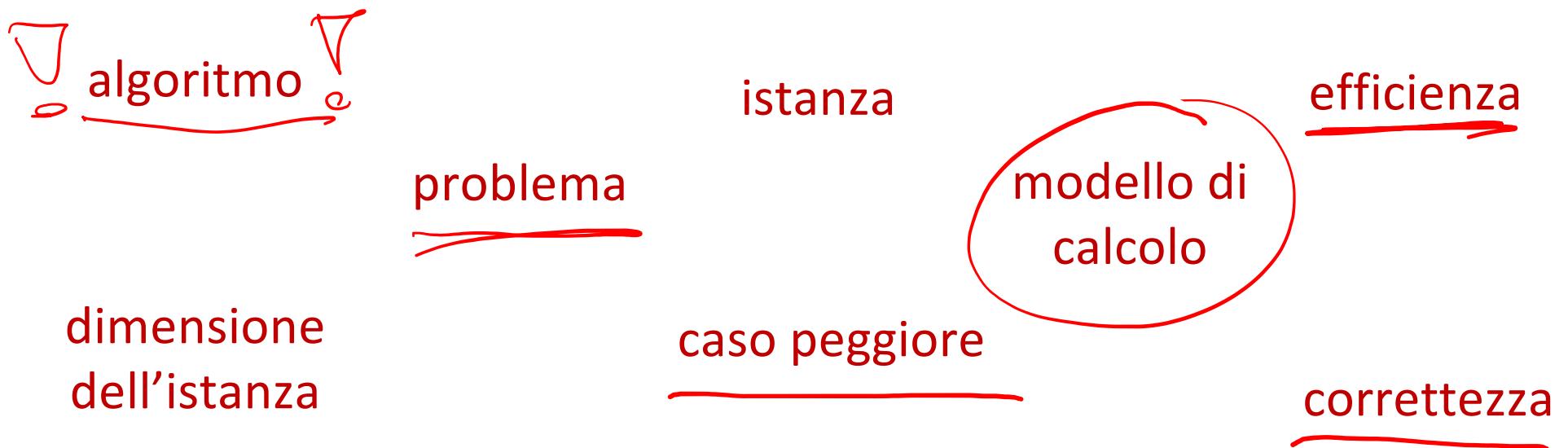
Algoritmo poco meno inefficiente: dividere ogni numero n per tutti i suoi predecessori fino a radice quadrata di n.

1



UNIVERSITÀ DI PISA

Alcuni concetti chiave



Concetti chiave applicati al nostro caso

problema: individuare i numeri primi minori di n

istanza: i numeri da 0 a n \rightarrow $0 - 79\ 324$ $0 - 22$

dimensione dell'istanza: il valore n

modello di calcolo: insieme di caselle contenente i numeri

algoritmo: strategia di calcolo dei numeri primi. La descrizione deve essere “comprendibile” e “compatta”. Deve descrivere la sequenza di operazioni sul modello di calcolo eseguite per una **generica** istanza

correttezza dell'algoritmo: la strategia di calcolo deve funzionare per una **generica** istanza, ovvero indipendentemente da quante caselle ci sono.

tornando ai concetti fondamentali

complessità temporale (dell'algoritmo): # di operazioni che esegue prima di individuare **TUTTI** i numeri primi dentro l'istanza. Dipende dalla dimensione dell'istanza e dall'istanza stessa.

complessità temporale nel caso peggiore: # **massimo** di operazioni che esegue su una istanza di una certa dimensione. E' una delimitazione superiore a quanto mi "costa" risolvere una **generica** istanza. Espressa come funzione della dimensione dell'istanza.

efficienza (dell'algoritmo): l'algoritmo deve fare poche operazioni, deve essere cioè **veloce**. Ma veloce rispetto a che? quando si può dire che un algoritmo è veloce?

Algoritmo molto più efficiente

Setaccio di Eratostene

1. Elencare tutti i numeri fino a n
2. Partendo dal primo numero primo, 2, cancellare dall'elenco tutti i multipli di 2
3. Ripetere il procedimento con i numeri seguenti non ancora cancellati

Setaccio di Eratostene

	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

Setaccio di Eratostene

Cancello i multipli di 2

	2	3	-4	5	-6	7	-8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

Setaccio di Eratostene

Cancello i multipli di 3 a partire da 9

	2	3	-4	5	-6	7	-8	-9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

Setaccio di Eratostene

Cancello i multipli di 5 a partire da 25

	2	3	-4	5	-6	7	-8	-9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

Setaccio di Eratostene

Cancello i multipli di 7 a partire da 49

	2	3	-4	5	-6	7	-8	-9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

Setaccio di Eratostene

```
void setaccio (int n)
{
    bool primi[n]; primi[0] = primi[1] = false;
    for (int i = 2; i < n; i++) primi[i] = true; //inizializza
    int i = 1;
    for (i++; i * i < n; i++)
    { //scorri i numeri successivi a partire da i*i *
        while (!primi[i]) i++; // cerca il prossimo primo
        for (int k=i * i; k< n; k+= i)
            primi[k]= false; //cancella multipli di i
    }
    for(int j = 2; j<n; j++)
        if (primi[j]) cout <<j<< endl; //stampa tutti primi
}
```

* i numeri non cancellati precedenti i^2 sono primi

Complessità degli algoritmi

complessità di un algoritmo

funzione (sempre positiva) che associa alla dimensione del problema il costo della sua risoluzione

Costo: tempo, spazio (memoria),

dimensione: dipende dai dati

Per confrontare due algoritmi si confrontano le relative funzioni di complessità

Complessitá vs Profiling

- **Profiling:** tecnica di analisi di un programma basata sulla misurazione di indici di prestazione del programma stesso (e.g., utilizzo memoria, #chiamate ci funzione, etc.)
- Questa misura dipende da fattori specifici dell'ambiente di esecuzione (calcolatore, sistema operativo, compilatore, linguaggio di programmazione) e dall'istanza considerata.

complessità degli algoritmi

E' necessario trovare un metodo di calcolo della complessità che misuri l'efficienza come proprietà dell'algoritmo, cioè astratta

- **dal computer su cui l'algoritmo è eseguito**
- **dal linguaggio in cui l'algoritmo è scritto**

Caso peggiore, migliore e medio

Misureremo il tempo di esecuzione di un algoritmo in funzione della dimensione n delle istanze

$$f(n)$$

Istanze diverse, a parità di dimensione, potrebbero però richiedere tempo diverso

Distinguiamo quindi ulteriormente tra analisi nel caso peggiore, migliore e medio

complessità dei programmi : esempio

$T_P(n)$ = Complessità del tempo di esecuzione del programma P al variare di n:

```
int max(int a[], int n)
{
    int m=1;
    for (int i=1; i < n; i++)
        if (m < a[i]) m = a[i];
    return m;
}
```

Se tutti i tempi costanti sono uguali a 1:

$$T_{\max}(n) = 4n$$

$$(n-1)4 + 1 + 1 + 1 + 1$$

$$4n - 4 + 4 \quad T_p(n) = 4n$$