

Risolvete il problema (se volete, postate la soluzione)

Problema: Somma massimale

Input: un vettore A contenente n interi (negativi/positivi)

Output: la **somma massimale**, cioè il valore massimo che si può ottenere sommando gli elementi di un qualunque **sottovettore contiguo** di A (sequenza di elementi consecutivi).



tinyurl.com/tresix

Potete usare una funzione $\text{sum}(A, i, j)$ che restituisce la somma degli elementi di A compresi fra i e j

Esempio: $\text{sum}(A, 4, 10) \rightarrow 21$

Algoritmi e Strutture Dati

20° Anno!

Alberto Montresor

Università di Trento

2025/05/30

This work is licensed under a Creative Commons
Attribution-ShareAlike 4.0 International License.



Cos'è un informatico?

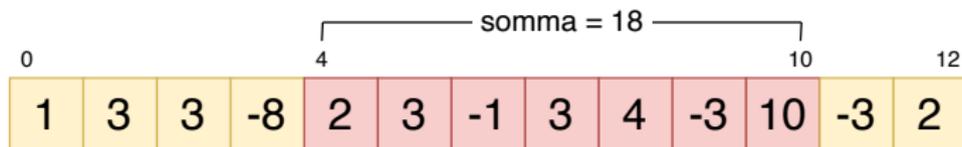


Colloquio di lavoro

Problema: Somma massimale

Input: un vettore A contenente n interi (negativi/positivi)

Output: la **somma massimale**, cioè il valore massimo che si può ottenere sommando gli elementi di un qualunque **sottovettore contiguo** di A (sequenza di elementi consecutivi).



tinyurl.com/tresix

Potete usare una funzione $\text{sum}(A, i, j)$ che restituisce la somma degli elementi di A compresi fra i e j

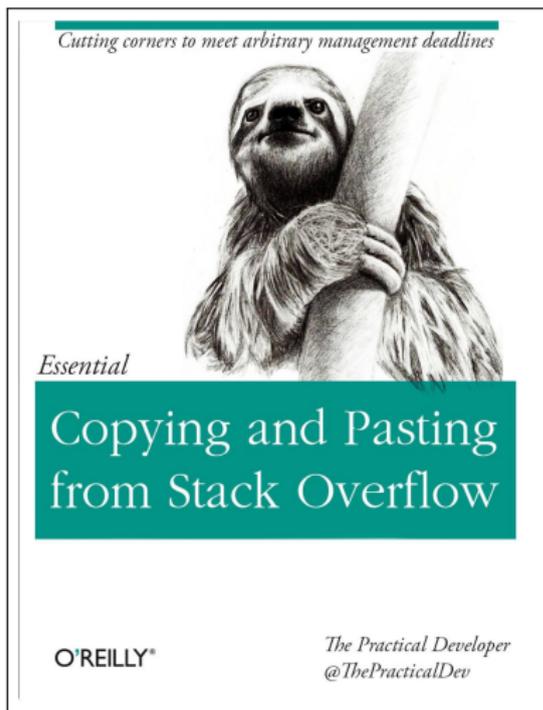
Esempio: $\text{sum}(A, 4, 10) \rightarrow 21$

La vostra risposta



Eh?
(NOOB)

La vostra risposta



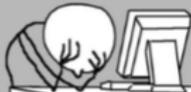
~~Cercavo qualcosa su StackOverflow~~
Chiedo a ChatGPT
(CODE MONKEY)

Days before OpenAI

Developer coding
- 2 hours



Developer debugging
- 6 hours

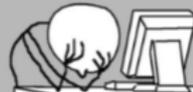


Days after OpenAI

ChatGPT generates Codes - 5 min



Developer debugging
- 24 hours



La vostra risposta



Posso sviluppare un algoritmo
efficiente per lei!
(COMPUTER SCIENTIST)

Versione 1 – Java – $O(n^3)$

Cicla su tutte le coppie (i, j) tali che $i \leq j$:

- chiama `sum()` per calcolare la somma dei valori compresi fra i e j ;
- aggiorna `maxSoFar` con il massimo fra la somma appena calcolata e il massimo trovato finora.

```
int maxsum1(int[] A, int n) {
    int maxSoFar = 0;           // Maximum found so far
    for (int i=0; i < n; i++) {
        for (int j=i; j < n; j++) {
            maxSoFar = max(maxSoFar, sum(A, i, j));
        }
    }
    return maxSoFar;
}
```

Versione 1 – Java – $O(n^3)$

Versione con il terzo ciclo esplicitato.

```
int maxsum1(int[] A, int n) {
    int maxSoFar = 0;           // Maximum found so far
    for (int i=0; i < n; i++) {
        for (int j=i; j < n; j++) {
            int sumSoFar = 0;   // Accumulator
            for (int k=i; k <= j; k++) {
                sumSoFar = sumSoFar + A[k];
            }
            maxSoFar = max(maxSoFar, sumSoFar);
        }
    }
    return maxSoFar;
}
```

Versione 2 – Java – $O(n^2)$

Ottimizzazione

Se ho calcolato la somma s dei valori in $A[i \dots j]$, la somma dei valori in $A[i \dots j + 1]$ è pari a $s + A[j + 1]$.

```
int maxsum2(int[] A, int n) {
    int maxSoFar = 0;                // Maximum found so far
    for (int i=0; i < n; i++) {
        int sumSoFar = 0;           // Accumulator
        for (int j=i; j < n; j++) {
            sumSoFar = sumSoFar + A[j];
            maxSoFar = max(maxSoFar, sumSoFar);
        }
    }
    return maxSoFar;
}
```

Versione 2 – Python – $O(n^2)$

Ottimizzazione

Se ho calcolato la somma s dei valori in $A[i \dots j]$, la somma dei valori in $A[i \dots j + 1]$ è pari a $s + A[j + 1]$.

```
def maxsum2(A):
    maxSoFar = 0                    # Maximum found so far
    for i in range(0, len(A)):
        sum = 0                    # Accumulator
        for j in range(i, len(A)):
            sum = sum + A[j]
            maxSoFar = max(maxSoFar, sum)
    return maxSoFar
```

Versione 2 – Python con librerie – $O(n^2)$

Uso di funzioni native

La funzione `accumulate()` del modulo `itertools` prende un vettore (lista) I come input e ritorna un vettore O tale che $O[k] = \sum_{i=0}^k I[i]$.

Può sostituire l'accumulatore nel codice precedente; normalmente è più veloce perchè l'implementazione sottostante è basata sul C

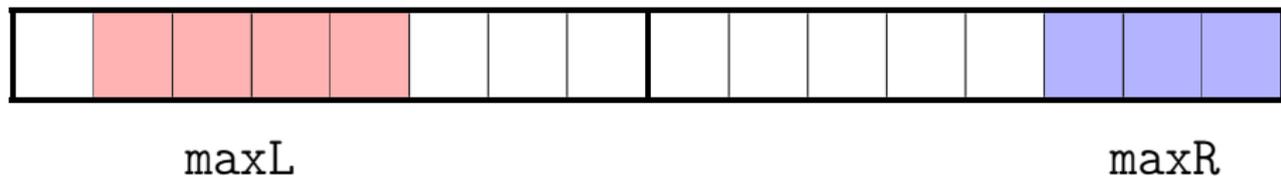
```
from itertools import accumulate
```

```
def maxsum2(A):  
    maxSoFar = 0 # Maximum found so far  
    for i in range(0, len(A)):  
        tot = max(accumulate(A[i:]))  
        maxSoFar = max(maxSoFar, tot)  
    return maxSoFar
```

Versione 3 – $O(n \log n)$

Divide-et-impera

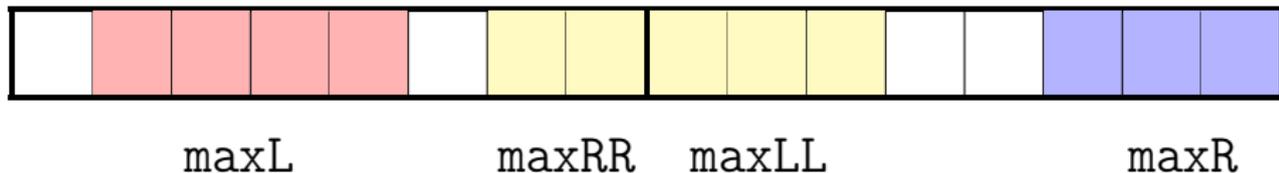
- Dividiamo il vettore in due parti più o meno uguali
- maxL è la somma massimale nella parte sinistra
- maxR è la somma massimale nella parte destra
- Ritorna il massimo dei due valori
- L'algoritmo proposto è corretto?



Versione 3 – $O(n \log n)$

Divide-et-impera

- Dividiamo il vettore in due parti più o meno uguali
- maxL è la somma massimale nella parte sinistra
- maxR è la somma massimale nella parte destra
- $\text{maxLL} + \text{maxRR}$ è il valore della sottolista massimale "a metà"
- Ritorna il massimo dei tre valori



Versione 3 – Python – $O(n \log n)$

```
def maxsum_rec(A,i,j):
    if (i==j):
        return max(0, A[i])
    m = (i+j)//2
    maxLL = 0 # Maximal subvector on the left ending in m
    sumSoFar = 0
    for k in range(m, i-1,-1):
        sumSoFar = sumSoFar + A[k]
        maxLL = max(maxLL, sumSoFar);
    maxRR = 0 # Maximal subvector on the right starting in m+1
    sumSoFar = 0
    for k in range(m+1,j+1):
        sumSoFar = sumSoFar + A[k]
        maxRR = max(maxRR, sumSoFar);
    maxL = maxsum_rec(A, i, m) # Maximal subvector on the left
    maxR = maxsum_rec(A, m+1, j) # Maximal subvector on the right
    return max(maxL, maxR, maxLL + maxRR)

def maxsum3(A):
    return maxsum_rec(A,0,len(A)-1)
```

Versione 4 – $O(n)$

Programmazione dinamica

Sia $maxHere_i$ il valore del sottovettore di somma massima che termina in posizione $A[i]$

$$maxHere_i = \begin{cases} \max(A[0], 0) & i = 0 \\ \max(maxHere_{i-1} + A[i], 0) & i > 0 \end{cases}$$

Il valore massimo contenuto in $maxHere_0 \dots maxHere_{n-1}$ rappresenta il valore del sottovettore di somma massima che termina in una qualunque posizione del vettore, quindi la nostra risposta.

Versione 4 – $O(n)$

```
int maxsum4(int A[], int n) {  
    int maxSoFar = 0;  
    int maxHere = 0;  
    for (int i=0; i < n; i++) {  
        maxHere = max(maxHere+A[i], 0);  
        maxSoFar = max(maxSoFar,maxHere);  
    }  
    return maxSoFar;  
}
```

A	1	3	4	-8	2	3	-1	3	4	-3	10	-3	2
maxHere	1	4	8	0	2	5	4	7	11	8	18	15	17
maxSoFar	1	4	8	8	8	8	8	8	11	11	18	18	18

La colonna associata ad ogni elemento del vettore A contiene il valore di `maxHere` e `maxSoFar` dopo l'esecuzione del ciclo per quell'elemento.

Versione 4 – $O(n)$ – Python

Stessa tecnica, ma in questo caso ritorniamo una coppia di indici.

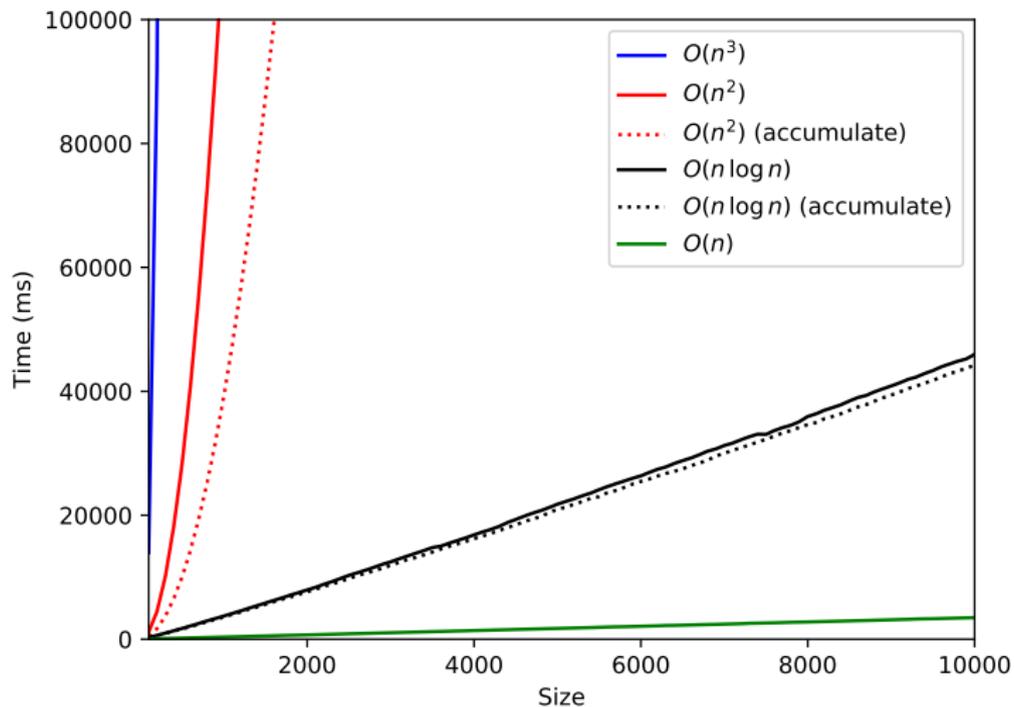
```
def maxsum4_slice(A):
    maxSoFar = 0      # Maximum found so far
    maxHere = 0      # Maximum slice ending at the current pos
    start = end = 0  # Start, end of the maximal slice found so far
    last = 0        # Beginning of the maximal slice ending here
    for i in range(0, len(A)):
        maxHere = maxHere + A[i]
        if maxHere <= 0:
            maxHere = 0
            last = i+1
        if maxHere > maxSoFar:
            maxSoFar = maxHere
            start, end = last, i
    return (start, end)
```

Versione 4 – $O(n)$

A	1	3	4	-8	2	3	-1	3	4	-3	10	-3	2
maxHere	1	4	8	0	2	5	4	7	11	8	18	15	17
maxSoFar	1	4	8	8	8	8	8	8	11	11	18	18	18
last	0	0	0	4	4	4	4	4	4	4	4	4	4
start	0	0	0	0	0	0	0	0	4	4	4	4	4
end	0	1	2	2	2	2	2	2	8	8	10	10	10

La colonna associata ad ogni elemento del vettore contiene il valore delle variabili dopo l'esecuzione del ciclo per quell'elemento.

Tempi di esecuzione



Un po' di storia

Sottovettore di somma massimale

- Dal punto di vista didattico, **best problem ever!**
- 1977: Ulf Grenander (Brown) introduce il problema, come versione semplificata di un problema più generale in immagini 2D (*maximum likelihood in image processing*)
- 1984: Algoritmo lineare proposto da Jay Kadane (Carnegie Mellon)

Jon Bentley. **Programming pearls: algorithm design techniques**. Commun. ACM 27(9):865-873. September, 1984. [[PDF](#)]

Jon Bentley. **Programming Pearls, 2nd ed.** Addison-Wesley, 2000.
[Una copia in BUC]

Esempio: Genome Sequence Analysis

"One line of investigation in genome sequence analysis is to locate the biologically meaningful segments, like conserved regions or GC-rich regions. A common approach is to assign a real number (also called score) to each residue, and then look for the maximum-sum segment."

Chao, Kun-Mao. **Genomic sequence analysis: A case study in constrained heaviest segments**. Computational Genomics: Current Methods, 2007, 49.

Due parole su di me

- Cosa insegno:
 - **Algoritmi e Strutture Dati**
LT INF (+LT ICE, LT MAT)
 - **Didattica dell'Informatica**
10 crediti nel Percorso 60CFU
- Highlights
 - Prorettore, Senatore accademico
+ 9 altre cariche
 - Co-fondatore Coderdojo Verona
 - Co-fondatore FabLab UniTrento



Organizzazione

Studenti Informatica

- Un corso **annuale** da 12 crediti (da 09/2024 a 05/2025), suddiviso in due moduli, ma con un unico esame orale finale

Studenti Matematica, Inf. Org., ICE

- Scelta fra versione 6 crediti (uno per semestre) o versione da 12 crediti.

Programma del corso

Modulo 1

- Introduzione
 - Analisi degli algoritmi
 - Notazione asintotica
 - Ricorrenze
 - Analisi ammortizzata
- Strutture dati
 - Strutture dati elementari
 - Alberi
 - Grafi
 - Insiemi e dizionari
- Tecniche di risoluzione
 - Divide-et-impera

Modulo 2

- Strutture dati avanzate
 - Code con priorità
 - Insiemi disgiunti
- Tecniche di risoluzione
 - Scelta struttura dati
 - Programmaz. dinamica
 - Algoritmi greedy
 - Ricerca locale
 - Backtrack
 - Algoritmi probabilistici
- Problemi intrattabili
 - Problemi NP-completi
 - Tecniche euristiche, approssimate

Scopo del corso

Conoscenze e competenze fondamentali

- **Conoscenze:** panoramica sui problemi fondamentali e le loro soluzioni algoritmiche
- **Competenze:** tecniche standard per risolvere un'ampia gamma di problemi complessi

Algoritmi

- Analizzate il loro codice
- Convincetevi che funzionano
- Implementateli!

Tecniche di soluzione

- Risolvete (tanti) problemi
- Analizzate le vostre soluzioni
- Implementatele!

Citazioni importanti – 1

"An algorithm must be seen to be believed, and the best way to learn what an algorithm is all about is to try it"



Donald Knuth
The Art of Computer Programming
Vol.1: "Fundamental Algorithms"
Section 1.1, page 4

<https://www.guitex.org/home/knuth-a-trento>

Citazioni importanti – 2

"Se volete fare gli scrittori, ci sono due esercizi fondamentali: leggere molto e scrivere molto. Non conosco stratagemmi per aggirare questa realtà, non conosco scorciatoie.

[...]

Quello che voglio dire è che per scrivere al meglio delle proprie capacità, è opportuno costruire la propria cassetta degli attrezzi e poi sviluppare i muscoli necessari a portarla con sé. Allora, invece di farsi scoraggiare davanti a un lavoro che si preannuncia complicato, può darsi che abbiate a disposizione l'utensile adatto con il quale mettervi immediatamente all'opera."

On writing, Stephen King

Riflessioni sul concetto di lezione universitaria

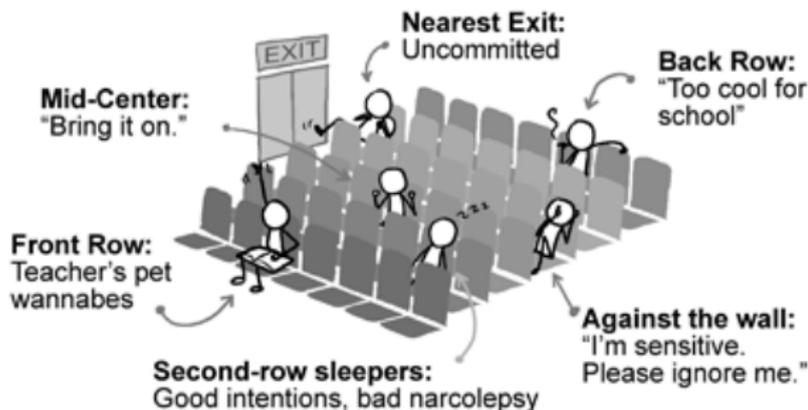


Laurentius da Voltolina – Bologna, seconda metà del XIV secolo

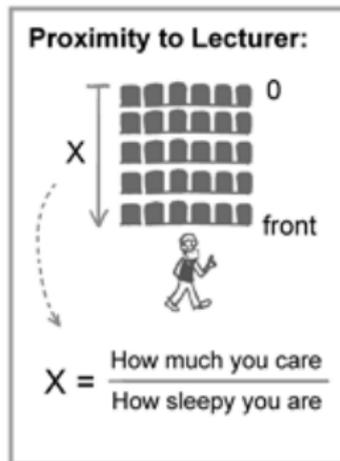
Riflessioni sul concetto di lezione universitaria

WHERE YOU SIT IN CLASS/SEMINAR

And what it says about you:



WWW.PHDCOMICS.COM



JORGE CHAM © 2008

Riflessioni sull'interazione a lezione

Chiedi, e ti vergognerai un attimo, non chiedere e ti vergognerai per sempre

Murakami Haruki
Kafka sulla spiaggia

If a listener nods his head when you're explaining your program, wake him up

Alan Perlis
Epigrams on Programming

Fate domande!

- Se sono poco chiaro, non esitate a chiedere ulteriori spiegazioni
- Se volete ulteriori approfondimenti, chiedete e vi sarà dato
- Non è detto che conosca tutte le risposte – ma so dove cercare!

Rispondete alle mie domande!

- Partecipare in 150+ è difficile, ma cercate di partecipare tutti

Come interagire

- "Iscrizione" corso:
<https://forms.gle/9USUD6LTQoBhbQo5A>
- Canale Telegram (annunci, parlo io): vedi home page del corso
- Gruppo telegram (discussione, parlate voi): vedi home page del corso
- Durante le lezioni (per porre domande)
<https://slido.com> Codice #ASD24
- Interazione personale con me
Via mail, non tramite telegram

Riflessioni sull'interazione a lezione



Sull'uso di portatili e cellulari durante la lezione

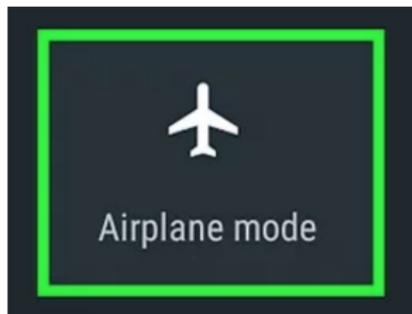


Sull'uso di portatili e cellulari durante la lezione

- C. May. **Students are Better Off without a Laptop in the Classroom.** Scientific American, 2017.
<https://www.scientificamerican.com/article/students-are-better-off-without-a-laptop-in-the-classroom/>
- C. Stothart, A. Mitchum, C. Yehnert. **The attentional cost of receiving a cell phone notification.** J Exp Psychol Hum Percept Perform. 41(4):893-7 (Aug. 2015). <https://www.ncbi.nlm.nih.gov/pubmed/26121498>
- A.F. Ward, K. Duke, A. Gneezy, and M.W. Bos. **Brain Drain: The Mere Presence of One's Own Smartphone Reduces Available Cognitive Capacity.** Journal of the Association for Consumer Research, 2(2):140-154 (April 2017)
<https://www.journals.uchicago.edu/doi/10.1086/691462>

Sull'uso di portatili e smartphone durante la lezione

Smartphone in modalità aereo



Smartphone nello zaino



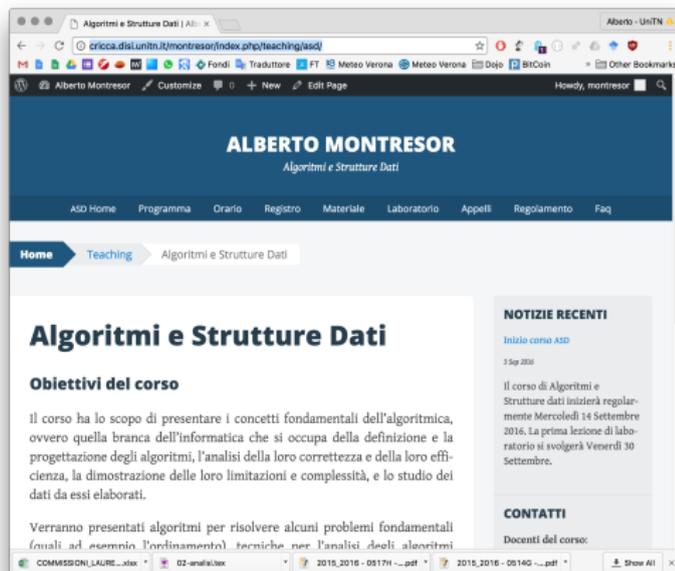
Spegnete il WiFi del portatile



Sito web del corso

<http://cricca.disi.unitn.it/montresor/asd/>

- Lucidi e appunti
- Video lezioni
- Software didattico
- Esercizi e compiti passati
- Progetti
- Approfondimenti



Sito web del corso



Docenti e assistenti

- Titolare: lezioni teoriche, esercitazioni, correzione scritti, orali
 - Prof. Alberto Montresor (alberto.montresor@unitn.it)
- Sessioni in laboratorio, correzioni progetti
 - Dott. Francesco Lotito (francesco.lotito@unitn.it)
 - Dott. Cristian Consonni (cristian.consonni@unitn.it)
- Tutorato:
 - Luca Mosetti
 - Carmen Casulli
 - Michele Ghilardi

Libro adottato

- Bertossi, Montresor
Algoritmi e Strutture di Dati.
Tecniche nuove, 3^a ed. (2014)
(€32.30)



Approfondimenti

- Cormen, Leiserson, Rivest, Stein. *Introduction to Algorithms.*
The MIT Press; 4th ed. (2022) (€135.71)
- Cormen, Leiserson, Rivest, Stein. *Introduction to Algorithms.*
The MIT Press; 3rd ed. (2009) (€81.98)
- Jon Kleinberg, Eva Tardos. *Algorithm Design.*
Addison Wesley, 1st Int. ed. (2013) (€82.00)

Lezioni e ricevimento

Lezioni

Martedì	10.30 – 12.30	Lezione/Esercitazione
Giovedì	13.30 – 15.30	Lezione/Esercitazione or Lab

Ricevimento

- Via zoom, annunciato settimanalmente
<https://tinyurl.com/TheHouseOfMontresor>
- Al termine di ogni lezione, in presenza
- Via mail, quando volete
- Su appuntamento

Esame diviso in due parti

- 50% - Parte scritta
 - Esame scritto (Uno per modulo/semestre) (Aula)
 - Progetti laboratorio (Uno per semestre) (Homework)
- 50% - Parte orale

Calcolo voto finale x 12 crediti

$$\frac{(\frac{\text{Voto Scritto 1} + \text{Bonus Lab1}}{2} + \frac{\text{Voto Scritto 2} + \text{Bonus Lab2}}{2}) + \text{Voto Orale}}{2}$$

Calcolo voto finale x 6 crediti

$$\frac{\text{Voto Scritto} + \text{Bonus Lab} + \text{Voto Orale}}{2}$$

Esame scritto

Open-book

- È possibile usare libri e appunti, non strumenti elettronici

Regole

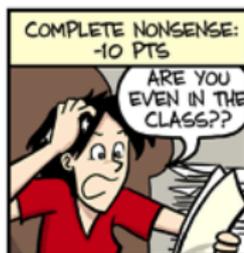
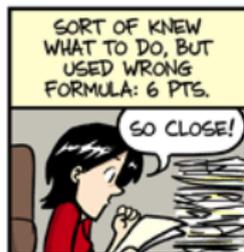
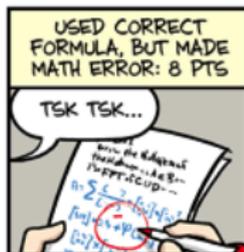
- **Salto appello:** in ogni anno solare (2024, 2025):
 - potete consegnare al massimo **3** scritti parte A
 - potete consegnare al massimo **3** scritti parte B
- **Ultimo voto:** se partecipate allo scritto del modulo X , l'eventuale voto già ottenuto del modulo X viene perso.

Compiti anni passati, con soluzioni

<http://cricca.disi.unitn.it/montresor/teaching/asd/materiale/esercizi/compiti/>

GRADING RUBRIC

PROBLEM 1 (TOTAL POINTS: 10)



JORGE CHAN © 2010



WWW.PHDCOMICS.COM

Progetti di laboratorio

- Svolgimento in un periodo di 8 giorni
- Corrette tramite software: Contest Management System (CMS), valutate in maniera competitiva
- Ricevete un bonus compreso fra 0 e 6 punti
 - Progetto 1: Dicembre 2024 [0-3] punti
 - Progetto 2: Maggio 2025 [0-3] punti

Esame orale

Per accedere all'orale, è necessario:

- Consegnare almeno un progetto funzionante
- Ottenere un voto scritto ≥ 17.5 , così definito:

$$12 \text{ crediti} \quad \frac{(\text{Voto Scritto 1} + \text{BonusLab 1}) + (\text{Voto Scritto 2} + \text{BonusLab2})}{2} \geq 17.5$$

$$6 \text{ crediti} \quad \text{Voto Scritto} + \text{Bonus Lab} \geq 17.5$$

- Dopo aver passato lo scritto, potete venire all'orale nello stesso appello d'esame o in un qualunque appello successivo
- Se rifiutate un voto all'orale, il voto dello scritto rimane valido
- Se l'appello è suddiviso in più giornate, non potete rifiutare e pretendere di tornare in una delle giornate successive; dovete passare all'appello successivo

Validità esami

I voti degli esami scritti non hanno scadenza

I voti dei progetti non hanno scadenza

Caveat emptor!

Se vi ripresentate fra 10 anni, non garantisco nulla....

Date scritti

Studenti 2024-25

Parte A	Parte B
01/2025	
02/2025	
06/2025	06/2025
07/2025	07/2025
09/2025	09/2025
	01/2025
	02/2025

Studenti precedenti

Parte A	Parte B
01/2025	01/2025
02/2025	02/2025
06/2025	06/2025
07/2025	07/2025
09/2025	09/2025

Cheating policies

Durante gli scritti

- È vietato comunicare in qualunque modo (oralmente, in forma scritta o elettronicamente), per qualsivoglia motivo.
- Chi viene sorpreso a parlare, viene invitato a lasciare l'aula e a ripresentarsi al prossimo appello
- Questo vale per entrambi gli "estremi" della comunicazione: sia chi parla sia chi ascolta
- Se avete bisogno di qualunque cosa, chiedete al docente

Dopo gli scritti

- Il compito potrà essere annullato anche in caso di manifesta copiatura scoperta nel corso della correzione degli scritti
- L'annullamento riguarderà sia il "copiatore" che il "copiato"

Art. 1

“Chiunque in esami o concorsi, prescritti o richiesti da autorità o pubbliche amministrazioni per il conferimento di lauree o di ogni altro grado o titolo scolastico o accademico, per l’abilitazione all’insegnamento ed all’esercizio di una professione, per il rilascio di diplomi o patenti, presenta, come proprii, dissertazioni, studi, pubblicazioni, progetti tecnici e, in genere, lavori che siano opera di altri, è punito con la reclusione da tre mesi ad un anno. La pena della reclusione non può essere inferiore a sei mesi qualora l’intento sia conseguito.”

Varie ed eventuali

Opportunità

- ACM-ICPC
- Google Summer of Code
- Google HashCode
- Hackathon(s)
- Speck&Tech
- Facoltiadi
- Coderdojo
- Palestra di algoritmi
- **Fablab**

Google Summer of Code

- Antonio Quartulli (2011)
- Federico Scrinzi (2012)
- Pietro Zambelli (2012)
- Edo Monticelli (2012)
- Savita Seetaraman (2014)
- Emilio Dorigatti (2015)
- Andrea Nardelli (2016)
- Lodovico Giarretta (2016)
- Giovanni De Toni (2017)
- Francesco Gazzetta (2018)
- Simone Degiacomi (2019)
- Beatrice Vergani (2020)
- Fabrizio Sandri (2022)
- Emanuele Grisenti (2022)
- Sebastiono Tocci (2023)

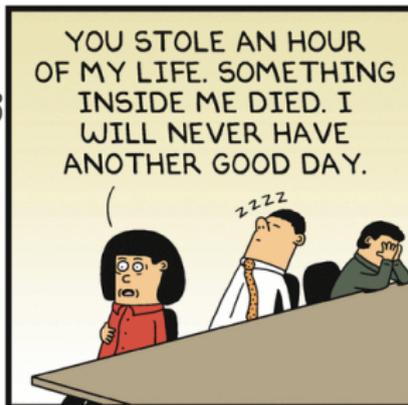
Valutazione della didattica

- Questionario valutazione didattica:
<https://goo.gl/forms/cpRZKmmJelfUomfG3>

Conclusioni



Dilbert.com DilbertCartoonist@gmail.com



©2011 Scott Adams, Inc./Dist. by Universal Uclick

