

Algoritmi e Strutture Dati

Stagione 16!

Alberto Montresor

Università di Trento

2021/02/28

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



Il Covid fa schifo!



Cos'è un informatico?



Problema: Sottovettore di somma massimale

- **Input:** un vettore di interi $A[1 \dots n]$
- **Output:** il sottovettore $A[i \dots j]$ di somma massimale, ovvero il sottovettore la cui somma degli elementi $\sum_{k=i}^j A[k]$ è più grande o uguale alla somma degli elementi di qualunque altro sottovettore.

La vostra risposta



Eh?
(NOOB)

La vostra risposta

Cutting corners to meet arbitrary management deadlines



Essential

Copying and Pasting from Stack Overflow

O'REILLY®

The Practical Developer
@ThePracticalDev

Alberto Montresor (UniTN)

ASD - Introduzione

2021/02/28

5 / 49

Cerco qualcosa su StackOverflow
(CODE MONKEY)

La vostra risposta



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

Colloquio di lavoro

Problema: Sottovettore di somma massimale

- **Input:** un vettore di interi $A[1 \dots n]$
- **Output:** il sottovettore $A[i \dots j]$ di somma massimale, ovvero il sottovettore la cui somma degli elementi $\sum_{k=i}^j A[k]$ è più grande o uguale alla somma degli elementi di qualunque altro sottovettore.
- Il problema è descritto bene?

Colloquio di lavoro

Problema: Sottovettore di somma massimale

- **Input:** un vettore di interi $A[1 \dots n]$
 - **Output:** il sottovettore $A[i \dots j]$ di somma massimale, ovvero il sottovettore la cui somma degli elementi $\sum_{k=i}^j A[k]$ è più grande o uguale alla somma degli elementi di qualunque altro sottovettore.
-
- Il problema è descritto bene?

1	3	4	-8	2	3	-1	3	4	-3	10	-3	2
---	---	---	----	---	---	----	---	---	----	----	----	---

Colloquio di lavoro

Problema: Sottovettore di somma massimale

- **Input:** un vettore di interi $A[1 \dots n]$
- **Output:** il sottovettore $A[i \dots j]$ di somma massimale, ovvero il sottovettore la cui somma degli elementi $\sum_{k=i}^j A[k]$ è più grande o uguale alla somma degli elementi di qualunque altro sottovettore.
- Il problema è descritto bene?

1	3	4	-8	2	3	-1	3	4	-3	10	-3	2
---	---	---	----	---	---	----	---	---	----	----	----	---

Colloquio di lavoro

Problema: Sottovettore di somma massimale

- **Input:** un vettore di interi $A[1 \dots n]$
- **Output:** il sottovettore $A[i \dots j]$ di somma massimale, ovvero il sottovettore la cui somma degli elementi $\sum_{k=i}^j A[k]$ è più grande o uguale alla somma degli elementi di qualunque altro sottovettore.

- Il problema è descritto bene?
- Riuscite a risolverlo?
- Riuscite a risolverlo in maniera efficiente?

Utility function: Somma sottovettore

Usiamo una funzione che dato un vettore A e due indici i, j contenuti in A , restituisca la somma di tutti gli elementi compresi fra i e j (inclusi).

Java

```
int sum(int[] A, int i, int j) {  
    int sumSoFar = 0;  
    for (int k = i; k <= j; k++) {  
        sumSoFar += A[k];  
    }  
    return sumSoFar;  
}
```

Python

```
sum(A[i:j+1])
```

C++

```
accumulate(A+i, A+(j+1), 0)
```

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 sum = 0;                    // Accumulator  
            for (int k=i; k <= j; k++) {  
                sum = sum + A[k];  
            }  
            maxSoFar = max(maxSoFar, sum);  
        }  
    }  
    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 sum = 0;                        // Accumulator
        for (int j=i; j < n; j++) {
            sum = sum + A[j];
            maxSoFar = max(maxSoFar, sum);
        }
    }
    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 perché l'implementazione sottostante è basata sul C

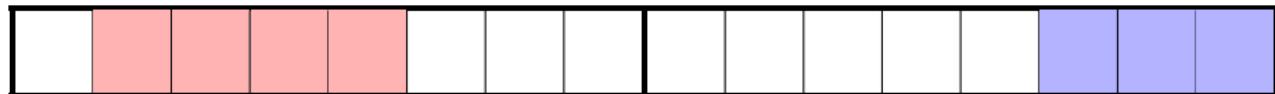
```
from itertools import accumulate

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

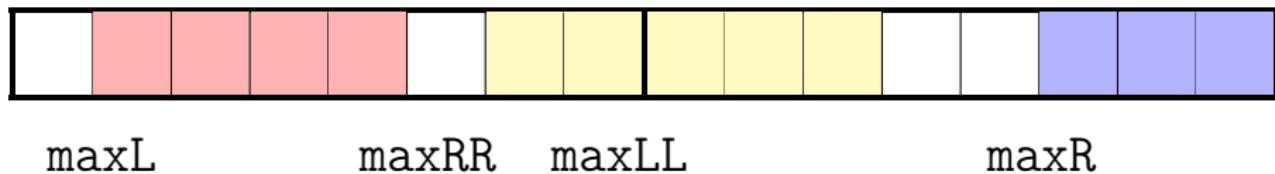
- Dividiamo il vettore a metà (sinistra, destra), in due parti più o meno uguali
- maxL è la somma massimale nella parte sinistra
- maxR è la somma massimale nella parte destra
- Ritorna il massimo dei due valori
- Riuscite a dimostrare che è corretto?



Versione 3 – $O(n \log n)$

Divide-et-impera

- Dividiamo il vettore a metà (sinistra, destra), 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
    sum = 0
    for k in range(m, i-1,-1):
        sum = sum + A[k]
        maxLL = max(maxLL, sum);
    maxRR = 0           # Maximal subvector on the right starting in m+1
    sum = 0
    for k in range(m+1,j+1):
        sum = sum + A[k]
        maxRR = max(maxRR, sum);
    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 $A[i]$

$$\maxHere[i] = \begin{cases} 0 & i < 0 \\ \max(\maxHere[i - 1] + A[i], 0) & i \geq 0 \end{cases}$$

Il valore massimo contenuto in \maxHere 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.

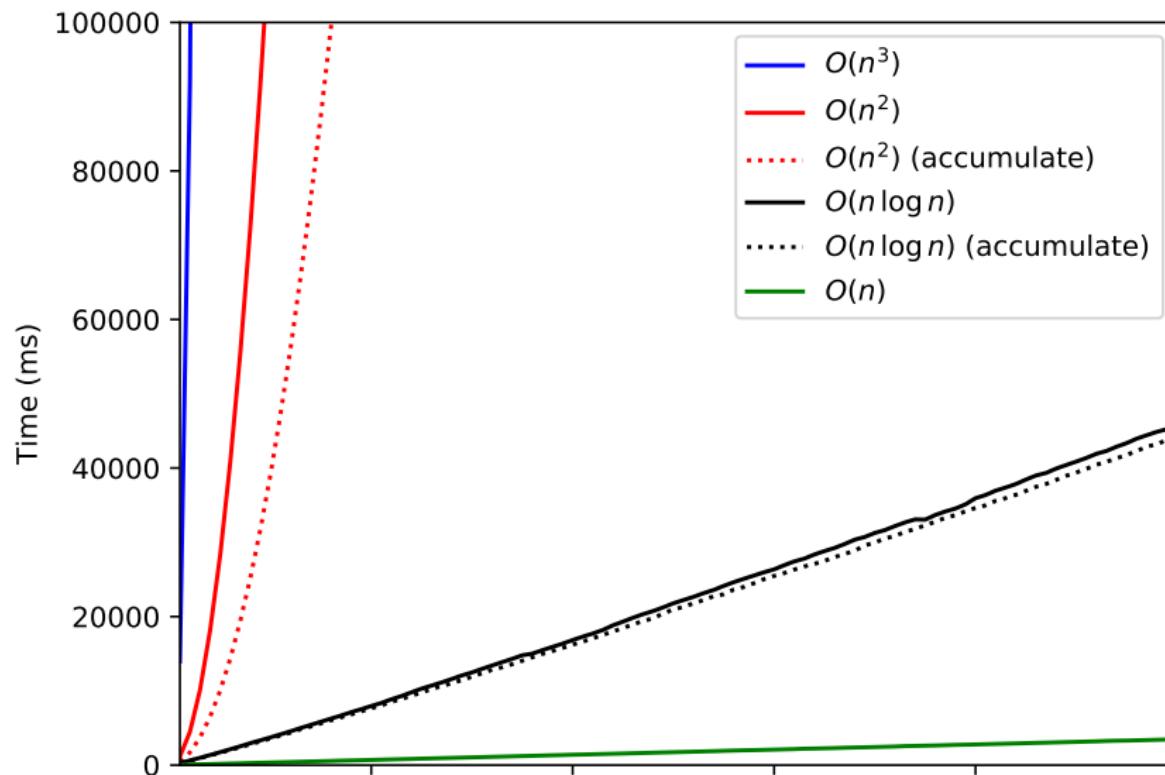
```
def maxsum4_slice(A):
    maxSoFar = 0          # Maximum found so far
    maxHere = 0           # Maximum slice ending at the current pos
    start = end = 0        # Start, end of the maximal slice found so far
    last = 0               # Beginning of the maximal slice ending here
    for i in 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]

Un po' di storia

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.

Studenti Informatica

- Un corso **annuale** da 12 crediti (da 09/2019 a 05/2020), suddiviso in due moduli ma con un unico esame orale finale

Studenti Matematica

- Due corsi da 6 crediti (uno per semestre), si può fare il primo o entrambi. Nel caso si faccia entrambi, un unico esame orale finale.

Studenti Inf. Org.

- Suggerito solo il corso del primo semestre, con esame orale, a meno che non vogliate fare la Magistrale di Informatica.

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
 - Programmazione dinamica
 - Algoritmi greedy
 - Ricerca locale
 - Backtrack
 - Algoritmi probabilistici
- Problemi intrattabili
 - Algoritmi approssimati
 - Problemi NP-completi

Scopo del corso

Conoscenze e competenze fondamentali

- **Conoscenze**: panoramica aggiornata 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 sè. 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

<http://cricca.disi.unitn.it/montresor/teaching/asd/la-cassetta-degli-attributi/>

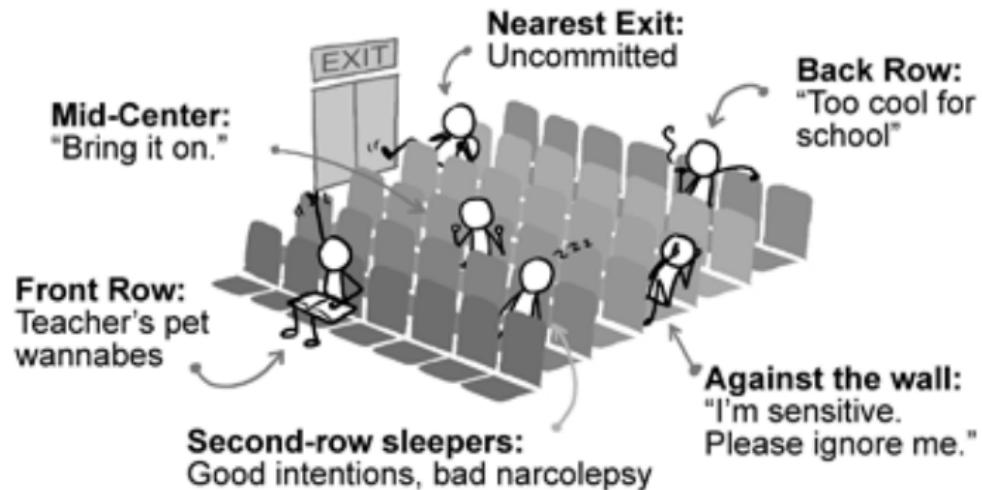
Riflessioni sul concetto di lezione universitaria



Riflessioni sul concetto di lezione universitaria

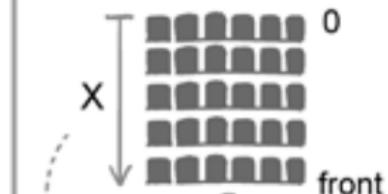
WHERE YOU SIT IN CLASS/SEMINAR

And what it says about you:



WWW.PHDCOMICS.COM

Proximity to Lecturer:



$$X = \frac{\text{How much you care}}{\text{How sleepy you are}}$$

JORGE CHAM © 2008

Riflessioni sull'interazione a lezione

Domanda e sembrerà sciocco per un minuto, non domandare e resterà sciocco per sempre

Proverbio cinese

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

Alan Perlis

Epigrams on Programming

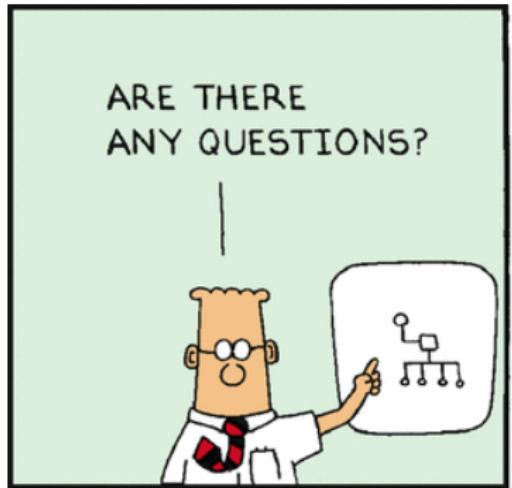
Fate domande!

- 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!

- Parlare in 150 è difficile, ma cercate di partecipare tutti

Riflessioni sull'interazione a lezione



Sito web del corso

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

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

The screenshot shows a web browser window with the following details:

- Title Bar:** Algoritmi e Strutture Dati | Alberto - UniTN
- Address Bar:** cricca.disi.unitn.it/montresor/index.php/teaching/asd/
- Toolbar:** Includes links for Fondi, Traduttore, FT, Meteo Verona, Dojo, BitCoin, and Other Bookmarks.
- User Information:** Alberto Montresor, Customize, Edit Page.
- Header:** ALBERTO MONTRESOR, Algoritmi e Strutture Dati
- Navigation Bar:** ASD Home, Programma, Orario, Registro, Materiale, Laboratorio, Appelli, Regolamento, Faq.
- Breadcrumbs:** Home > Teaching > Algoritmi e Strutture Dati
- Main Content:**

Algoritmi e Strutture Dati

Obiettivi del corso

Il corso ha lo scopo di presentare i concetti fondamentali dell'algoritmica, ovvero quella branca dell'informatica che si occupa della definizione e la progettazione degli algoritmi, l'analisi della loro correttezza e della loro efficienza, la dimostrazione delle loro limitazioni e complessità, e lo studio dei dati da essi elaborati.

Verranno presentati algoritmi per risolvere alcuni problemi fondamentali
- NOTIZIE RECENTI:** Inizio corso ASD (3 Sep 2016)
- CONTATTI:** Dati contatti del corso
- Page Footer:** Alberto Montresor (UniTN), ASD - Introduzione, 2021/02/28, 33 / 49

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. Antonio Buccharone (antonio.buccharone@fbk.eu)
 - Dott. Cristian Consonni (cristian.consonni@unitn.it)
 - Dott.ssa Marta Fornasier (marta.fornasier@studenti.unitn.it)

Testi

Libro adottato

- Bertossi, Montresor
Algoritmi e Strutture di Dati.
Tecniche nuove, 3^a ed. (2014)
(€29.45)



Approfondimenti

- Cormen, Leiserson, Rivest, Stein. *Introduction to Algorithms.*
The MIT Press; 3rd ed. (2009) (€55.72)
- Jon Kleinberg, Eva Tardos. *Algorithm Design.*
Addison Wesley, 1st Int. ed. (2013) (€74.96)

Lezioni e ricevimento

Lezioni

Martedì	15.30 – 17.30	Lezione/Esercitazione or Lab
Giovedì	10.30 – 12.30	Lezione/Esercitazione

Ricevimento

- Ricevimenti di gruppo, più avanti
- Al termine di ogni lezione, online
- Via mail, quando volete
- Su appuntamento

Esame

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

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

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:
 - 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.

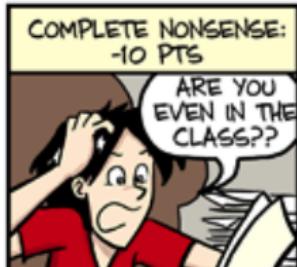
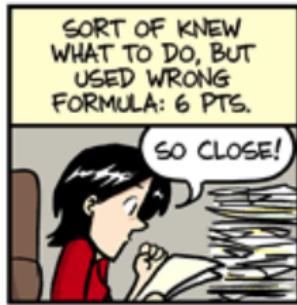
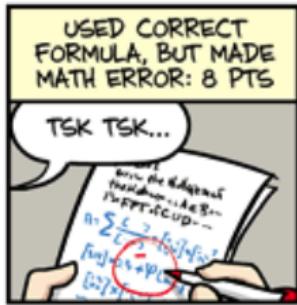
Compiti anni passati, con soluzioni

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

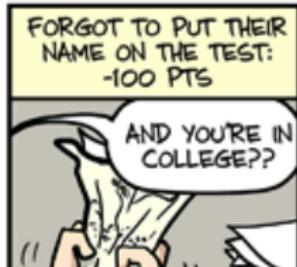
Esame scritto

GRADING RUBRIC

PROBLEM 1 (TOTAL POINTS: 10)



ORGE CHAM © 2010



Progetti di laboratorio

- Corrette tramite software: Contest Management System (CMS), valutate in maniera competitiva
- Ricevete un bonus compreso fra 0 e 6 punti
 - Progetto 1: Dicembre 2020 [0-3] punti
 - Progetto 2: Maggio 2021, [0-3] punti

Esame orale

Per accedere all'orale, è necessario:

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

$$\begin{array}{ll} \text{12 crediti} & \frac{(\text{Voto Scritto 1} + \text{BonusLab 1}) + (\text{Voto Scritto 2} + \text{BonusLab2})}{2} \geq 18 \\ \text{6 crediti} & \text{Voto Scritto} + \text{Bonus Lab} \geq 17.5 \end{array}$$

- 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 parziali

Studenti 2019-20

Parte 1	Parte 2
01/2021	
02/2021	
	06/2021
06/2021	06/2021
07/2021	07/2021
09/2021	09/2021
	01/2022
	02/2022

Studenti precedenti

Parte 1	Parte 2
01/2021	01/2021
02/2021	02/2021
06/2021	06/2021
07/2021	07/2021
09/2021	09/2021

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”

LEGGE 19/04/1925, n. 475 – GU 29/04/1925 , n. 99

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
- Olimpiadi dell'Informatica

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)

Contatti

- Canale Telegram (annunci lezione):
<https://t.me/ASD20Unitn>
- "Iscrizione" corso:
<https://forms.gle/1iax2wc8L6VmWaYy8>
- Questionario valutazione didattica:
<https://goo.gl/forms/cpRZKmmJelfUomfG3>

Conclusioni

