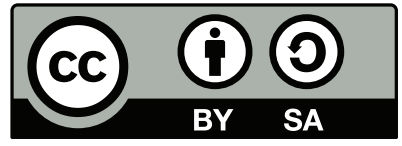
Pensiero computazionale

Introduzione

Alberto Montresor

Università di Trento

2018/09/09



Sommario

Sommario

Due parole su di me (Versione professionale)

Alberto Montresor

- Cosa insegno?
  - Algoritmi e strutture dati nel CdL in Informatica
  - Scientific programming nel CdLM in Quantitative and Computational Biology
  - Distributed Algorithms nel CdLM in Computer Science
- Quali sono i miei argomenti di ricerca?
  - Sistemi distribuiti decentralizzati e di grandi dimensioni
  - Analisi di grandi quantità di dati non strutturati (big data)

Due parole su di me (Versione personale)

Alberto Montresor

- Marito di un'insegnante di Scuola Secondaria di I grado (lettere)
- Padre di due figli, 10 e 12 anni
- Fondatore Coderdojo Verona

Obiettivi del corso

Dal punto di vista scientifico

- Definizione e motivazione
- Fondamenti epistemologici del pensiero computazionale
- Come insegnare il pensiero computazionale
- Come valutare il pensiero computazionale

Dal punto di vista pratico

- I fondamenti di Scratch
- Introduzione ad App Inventor

Valutazione finale

- Realizzazione di un modulo didattico

Pensiero computazionale – Una prima definizione

Wing, Jeannette (2014). "Computational Thinking Benefits Society". 40th Anniversary Blog of Social Issues in Computing. [Link]

Definizione [Wing'14]

- "Thinking as a computer scientist"

Pensiero computazionale – Una prima definizione

Wing, Jeannette (2014). "Computational Thinking Benefits Society". 40th Anniversary Blog of Social Issues in Computing. [Link]

Definizione [Wing'14]

- "Thinking as a computer scientist"
- Computational thinking is the thought processes involved in formulating a problem and expressing its solution(s) in such a way that a computer – human or machine – can effectively carry out.

Pensiero computazionale – Una prima definizione

Wing, Jeannette (2014). "Computational Thinking Benefits Society". 40th Anniversary Blog of Social Issues in Computing. [Link]

Definizione [Wing'14]

- "Thinking as a computer scientist"
- Computational thinking is the thought processes involved in formulating a problem and expressing its solution(s) in such a way that a computer – human or machine – can effectively carry out.

Nota etimologica:
Il termine computer entra nella lingua Inglese negli anni '40...

Pensiero computazionale – Una prima definizione

Wing, Jeannette (2014). "Computational Thinking Benefits Society". 40th Anniversary Blog of Social Issues in Computing. [Link]

Definizione [Wing'14]

- "Thinking as a computer scientist"
- Computational thinking is the thought processes involved in formulating a problem and expressing its solution(s) in such a way that a computer – human or machine – can effectively carry out.

Nota etimologica:
Il termine computer entra nella lingua Inglese negli anni '40... del 1600!

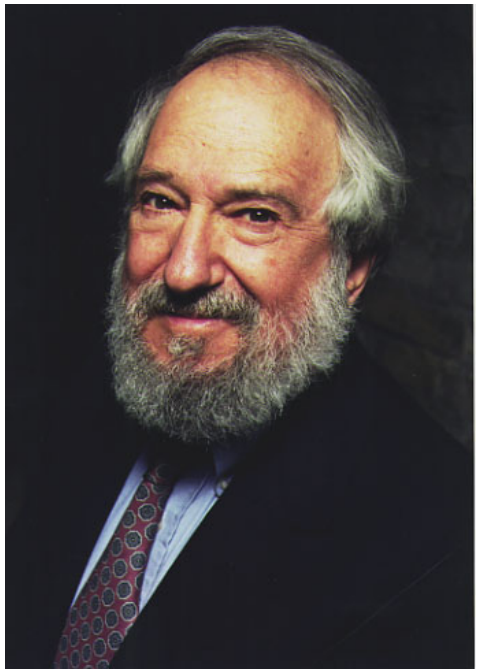
Sommario

Seymour Papert – Anni '80

Il termine "computational thinking" viene introdotto per la prima volta da Seymour Papert

Seymour Papert (1928-2016)

- Ha lavorato con Piaget
- Ha sviluppato la teoria del costruzionismo, come evoluzione del costruttivismo
- Inventore del linguaggio LOGO



- Papert, Seymour. *Mindstorms: Children, computers, and powerful ideas*. Basic Books, Inc. (1980) [Link]

Costruzionismo

Costruttivismo

- afferma che l'individuo che apprende costruisce modelli mentali per comprendere il mondo intorno a lui
- vede l'apprendimento come una ricostruzione piuttosto che come trasmissione di conoscenze

Costruzionismo

- sostiene che l'apprendimento avviene in modo più efficiente se questi modelli mentali vengono riflessi nella produzione di oggetti "tangibli" e "significativi"

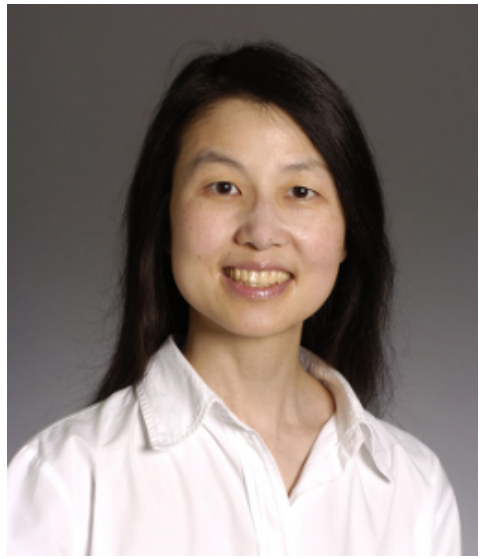
Mitchel Resnick – Anni '90

- LEGO Papert Professor of Learning Research
- Ha fondato, insieme a Natalie Rusk, il primo Computer ClubHouse (1993)
- Con il suo gruppo, ha contribuito allo sviluppo dei Lego Mindstorms (1998) e ha sviluppato il linguaggio Scratch (2005)

- Yasmin B. Kafai, Mitchel Resnick, eds. *Constructionism in Practice: Designing, Thinking, and Learning in a Digital World.* Routledge. (1996)

Jeannette Wing – Anni '00

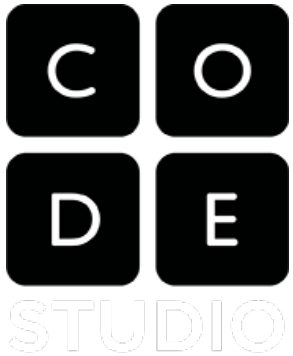
- Corporate Vice President of Microsoft Research
- Ha popolarizzato il concetto di Computational Thinking
- Ha fornito una prima definizione operativa del termine



- Wing, Jeanette M. "Computational thinking". *Communications of the ACM*, 49(3):33, 2006 [Link]
- Wing, Jeannette. "Computational Thinking Benefits Society". *40th Anniversary Blog of Social Issues in Computing*. (2014) [Link]

Code.org – 2013

Code.org è un'organizzazione no-profit il cui scopo è incoraggiare le persone, ed in particolare gli studenti di scuole superiori negli Stati Uniti, a imparare l'informatica.



- Organizzatori dell'Ora del codice e della Codeweek negli Stati Uniti
- Supportati da aziende come Microsoft, Apple, Facebook, Twitter
  - Video con Bill Gates, Mark Zuckenberg, Will.I.Am, etc.
- Endorsement da parte del presidente Obama (2015)
- Lobbying per lo sviluppo di nuovi curriculi per tutti gli ordini scolastici
- Integrazione razziale, riduzione del gap di genere

Un grosso impatto sui media

The New York Times

U.S.

Reading, Writing, Arithmetic, and Lately, Coding

By **MATT RICHTEL** MAY 10, 2014

Riforme dei curriculi scolastici

- Inghilterra (2014-2015)
- Australia, Francia, Polonia, Finlandia (2016-2017)
- Stati Uniti (2017-2018)

Computer Science for All

In the coming years, we should build on that progress, by . . . offering every student the hands-on computer science and math classes that make them job-ready on day one.

President Obama
State of the Union Address, 2016

UK National Curriculum in Computing (2014-2015)

Riforma del curriculum scolastico

- Sostituzione della materia "Information and Communications Technology (ICT)" con "Computing", che prevede un'introduzione alla programmazione
- Prevede corsi di aggiornamento degli insegnanti sulle nuove tecnologie

Valutazione preliminare (2 anni)

- Grande successo fra gli studenti
- Solo 15% dei docenti a loro agio con il nuovo curriculum
- Oltre il 60% dei docenti si auto-valutano non sufficientemente preparati per questa sfida

UK National Curriculum in Computing (2014-2015)

Goals

All pupils:

- can understand and apply the fundamental principles and concepts of computer science, including abstraction, logic, algorithms and data representation
- can analyse problems in computational terms, and have repeated practical experience of writing computer programs in order to solve such problems
- can evaluate and apply information technology, including new or unfamiliar technologies, analytically to solve problems
- are responsible, competent, confident and creative users of information and communication technology

Pensiero computazionale in Italia

PNSD

- Azione #17: Portare il pensiero logico-computazionale a tutta la scuola primaria
- Azione #18: Aggiornare il curricolo di Tecnologia alla scuola secondaria di primo grado
- Azione #15: Sviluppo di competenze digitali applicate

Riforma Gelmini

Risultati di apprendimento comuni a tutti i percorsi liceali: "Essere in grado di utilizzare criticamente strumenti informatici e telematici nelle attività di studio e di approfondimento; comprendere la valenza metodologica dell'informatica nella formalizzazione e modellizzazione dei processi complessi e nell'individuazione di procedimenti risolutivi."

Programma il futuro – 2014

Programma il futuro è un'iniziativa ministeriale che ha l'obiettivo di fornire alle scuole una serie di strumenti semplici, divertenti e facilmente accessibili per formare gli studenti ai concetti di base dell'informatica.

- Organizzatori dell'Ora del codice in Italia
- Il sito è l'emanazione dell'Azione #17 del PNSD

Gli strumenti disponibili sono [...] progettati e realizzati in modo da renderli utilizzabili in classe da parte di insegnanti di qualunque materia. Non è necessaria alcuna particolare abilità tecnica né alcuna preparazione scientifica

Sommario

Introduzione

Il tipo di "pensiero" richiesto per poter lavorare e partecipare con successo in una società è collegato ai materiali disponibili e ai processi di produzione utilizzati per risolvere i problemi.

Società industriale

Conoscere le proprietà fisiche dei materiali e degli strumenti, allo scopo di utilizzarli e combinarli per risolvere problemi "fisici".

Società dell'informazione

Conoscere le proprietà logiche dei dati e degli strumenti che operano su di essi, per localizzare e utilizzare le informazioni necessarie per risolvere problemi.

Oltre l'Information and Communication Technology

Computational Thinking (CT)

Il pensiero computazionale è il processo mentale coinvolto nella formulazione di un problema e nell'espressione delle sue soluzioni, in modo che possano esser effettivamente portate a termine da un esecutore – umano o artificiale.

- Formulare un problema: creare una rappresentazione astratta del problema
- Esprimere una soluzione: creare una rappresentazione linguistica della soluzione, allo scopo di comunicare la soluzione ad altri
- Effettivo: nell'ambito dell'informatica, approcci risolutivi che portino correttamente e velocemente alla soluzione richiesta

Le due "A" del pensiero computazionale

Astrazione

Il processo di astrazione richiede di:

- Catturare le proprietà essenziali in comune ad un insieme di oggetti
- Nascondere le distinzioni irrilevanti
- Operare simultaneamente a multipli livelli di astrazione
- Definire le relazioni tra i livelli di astrazione

Abstractions are our "mental" tools [Wing'06]

Alcuni esempi – Tipi di dato

Tipo di dato astratto

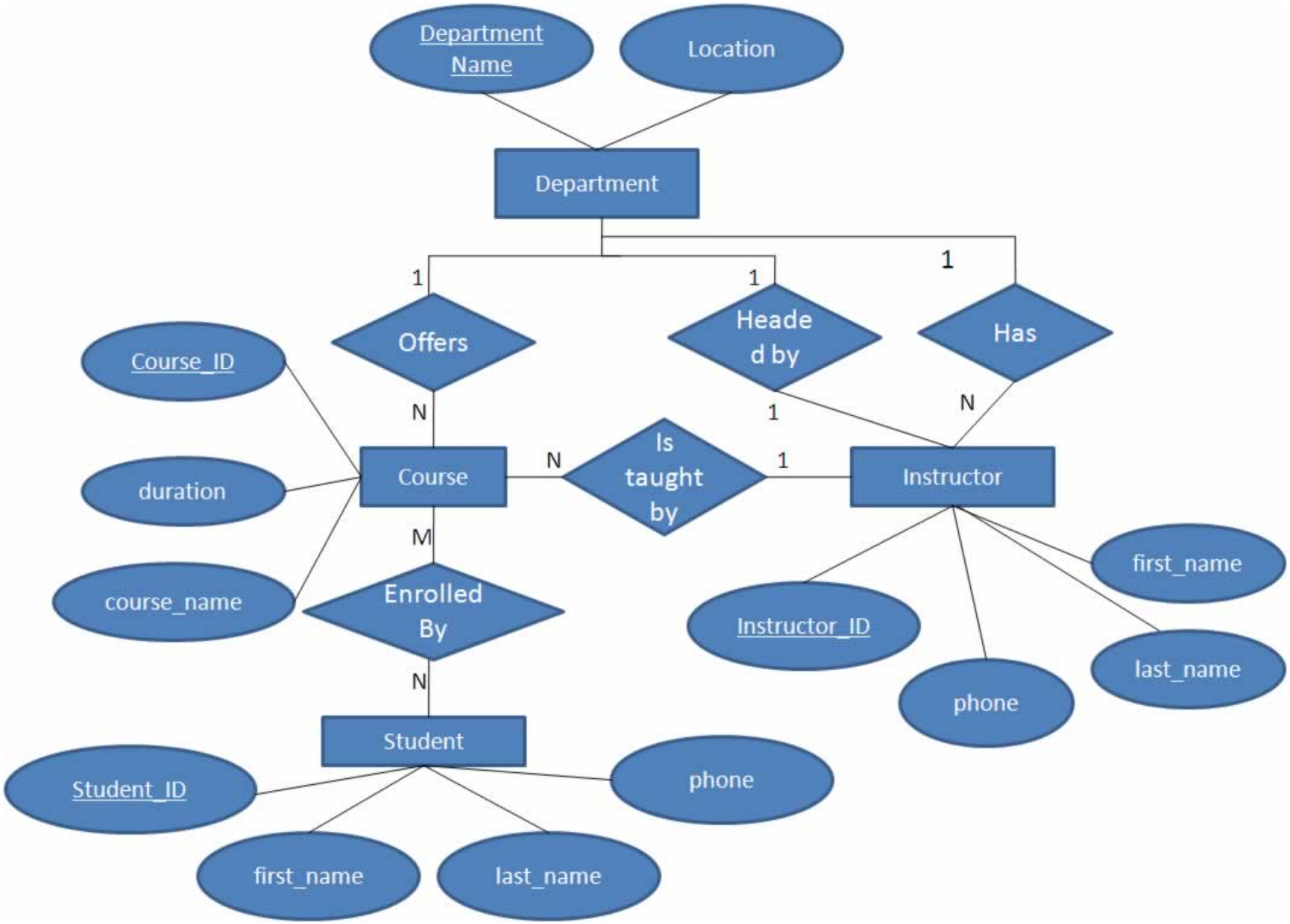
Un tipo di dato astratto è definito da un insieme di valori e le operazioni per manipolarli.

Example (Dizionario)

Un dizionario è un tipo di dati astratti composto da una collezione di coppie (chiave, valore), dove ogni possibile chiave compare una volta sola nella collezione.

- lookup(key) $\rightarrow$ value
- put(key, value)
- delete(key, value)

Alcuni esempi – Diagramma ER



Le due "A" del Pensiero Computazionale

Automazione

Il processo di automazione richiede di:

- Tradurre i concetti astratti in notazioni precise, che possano essere interpretate da un esecutore
- Definire un insieme di passi astratti che vengono eseguiti sui concetti astratti

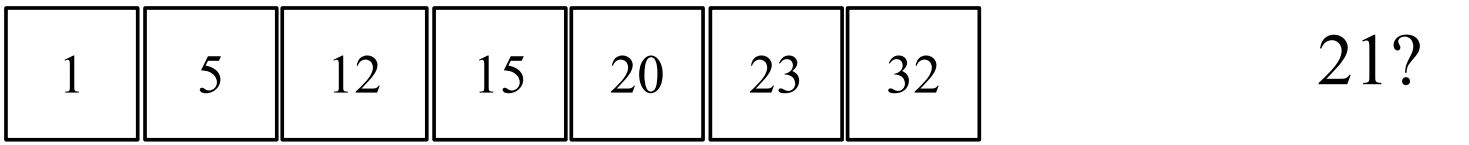
The power of our "mental" tools is amplified by our "metal" tools [Wing'06]

Esempio

Algoritmo

Un algoritmo è un'astrazione di un processo che prende un input, esegue un insieme di passi, e produce un output che soddisfa le caratteristiche richieste.

Example (Ricerca dicotomica)

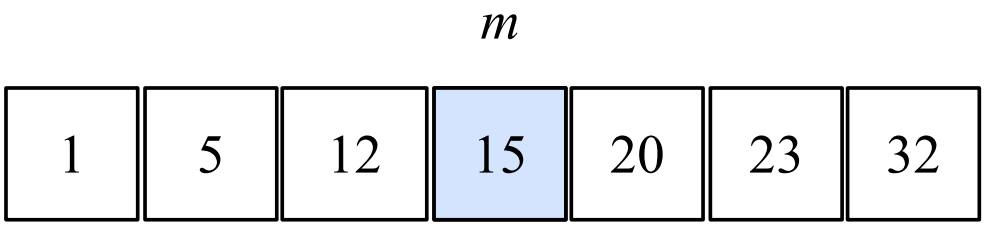


Esempio

Algoritmo

Un algoritmo è un'astrazione di un processo che prende un input, esegue un insieme di passi, e produce un output che soddisfa le caratteristiche richieste.

Example (Ricerca dicotomica)

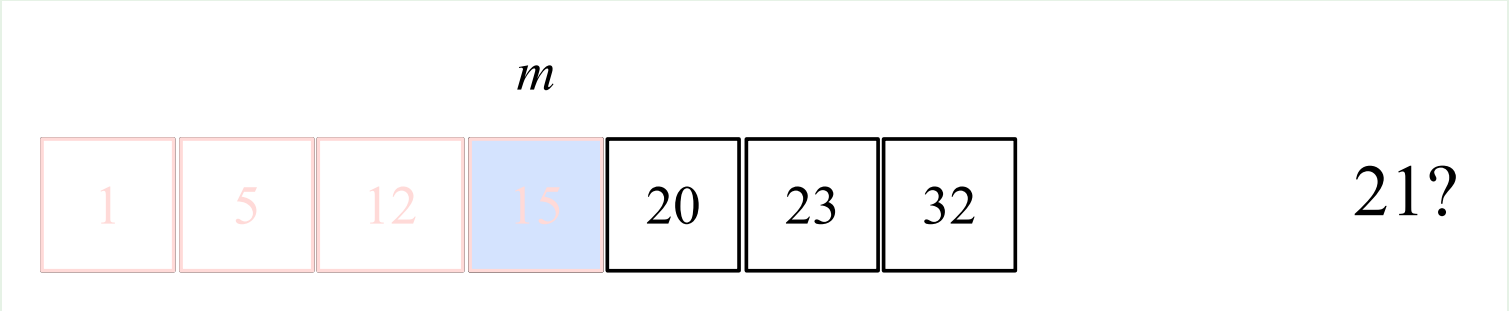


Esempio

Algoritmo

Un algoritmo è un'astrazione di un processo che prende un input, esegue un insieme di passi, e produce un output che soddisfa le caratteristiche richieste.

Example (Ricerca dicotomica)

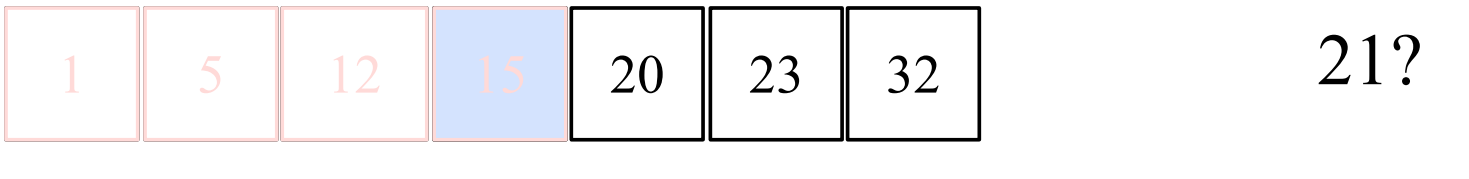


Esempio

Algoritmo

Un algoritmo è un'astrazione di un processo che prende un input, esegue un insieme di passi, e produce un output che soddisfa le caratteristiche richieste.

Example (Ricerca dicotomica)

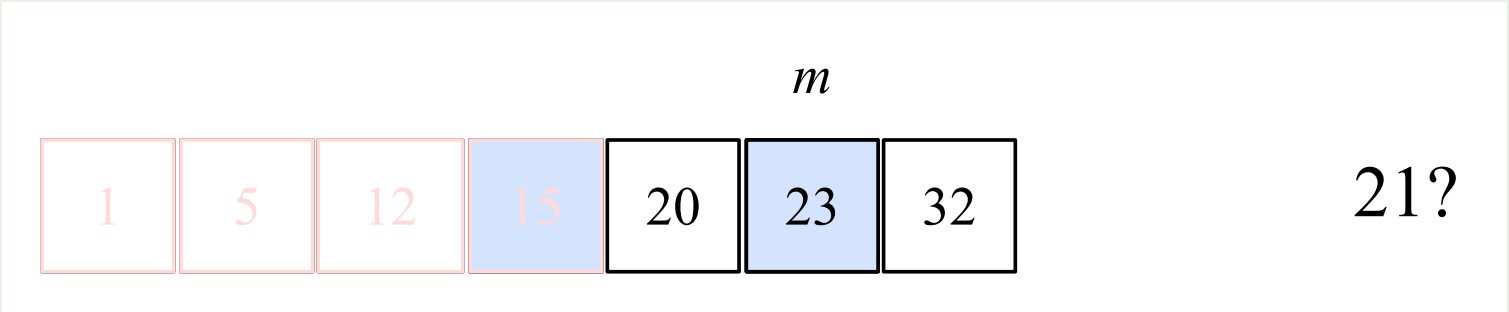


Esempio

Algoritmo

Un algoritmo è un'astrazione di un processo che prende un input, esegue un insieme di passi, e produce un output che soddisfa le caratteristiche richieste.

Example (Ricerca dicotomica)

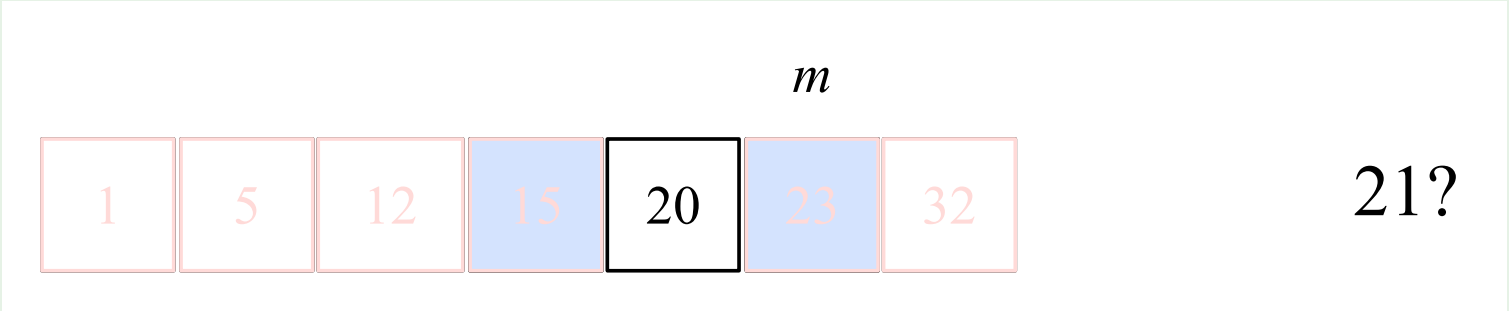


Esempio

Algoritmo

Un algoritmo è un'astrazione di un processo che prende un input, esegue un insieme di passi, e produce un output che soddisfa le caratteristiche richieste.

Example (Ricerca dicotomica)

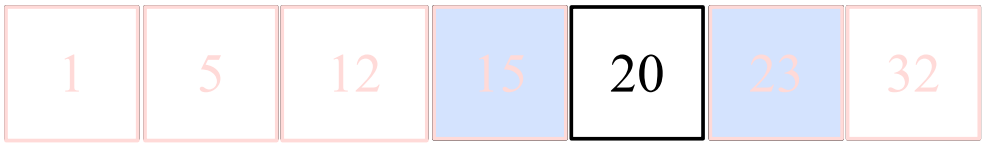


Esempio

Algoritmo

Un algoritmo è un'astrazione di un processo che prende un input, esegue un insieme di passi, e produce un output che soddisfa le caratteristiche richieste.

Example (Ricerca dicotomica)



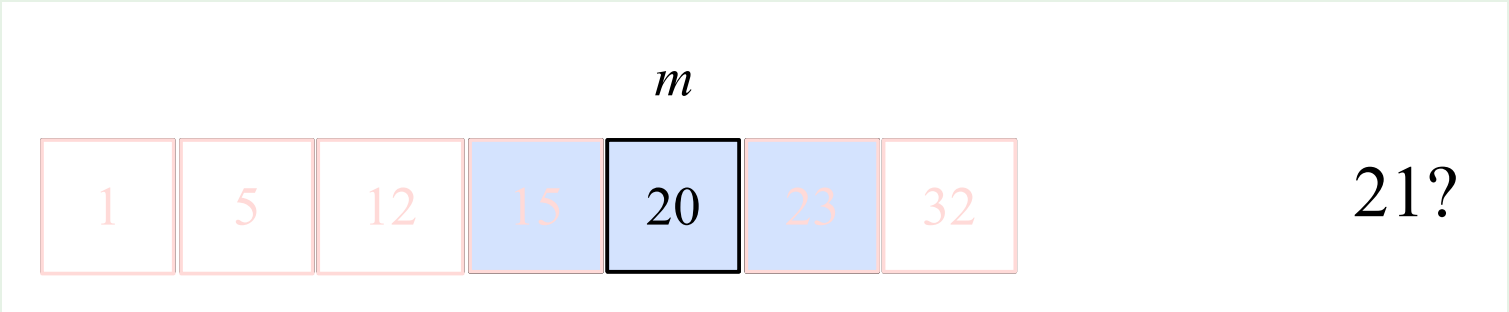
21?

Esempio

Algoritmo

Un algoritmo è un'astrazione di un processo che prende un input, esegue un insieme di passi, e produce un output che soddisfa le caratteristiche richieste.

Example (Ricerca dicotomica)



Dove sta la novità?

Il pensiero computazionale complementa, estende e combina approcci già esistenti, quali il pensiero matematico e il pensiero ingegneristico.

- L'astrazione è alla base del pensiero matematico; l'informatica basa le sue definizioni formali sulla matematica.
  - Il tipo di dati dizionario non è altro che una funzione parziale
  - La ricerca dicotomica ricorda il metodo della bisezione
  - Il diagramma ER sembra una mappa concettuale
- L'automazione e la ripetizione sono alla base del pensiero ingegneristico; l'informatica è una branca dell'ingegneria.
  - Algoritmo di Borůvka (1926), un metodo di costruzione di una rete elettrica efficiente per la Moravia
- In fin dei conti, stiamo parlando di problem solving...

Oltre il problem solving

Diversamente da quelle matematiche e logiche, le astrazioni create dal pensiero computazionale sono tangibili, in quanto processabili

Citazione

"Computing is the automation of abstractions"
Aho e Ullman, Foundations of Computer Science, 1992 [Link]

Il pensiero computazionale ("pensare come un informatico") implica:

- osservare il mondo ragionando per modelli astratti
- creare astrazioni che sono processabili e tangibili
- avere la possibilità di testare le proprie astrazioni
- avere l'audacia e la capacità di ragionare in grande ("scalare")

Rendere processabile un'astrazione

Java

```
Map<String, String> capoluoghi = new HashMap<>();
capoluoghi.put("Toscana", "Firenze");
capoluoghi.put("Lombardia", "Milano");
capoluoghi.put("Sardegna", "Cagliari");
```

Rendere processabile un algoritmo

int binarySearch(**int**[ ] A, **int** n, **int** v)

int $i = 1$
int $j = n$
int $m = \lfloor (i + j)/2 \rfloor$
while $i < j$ **and** $A[m] \neq v$ **do**
    if $A[m] < v$ **then**
        $i = m + 1$
    else
        $j = m - 1$
    $m = \lfloor (i + j)/2 \rfloor$
if $i > j$ **or** $A[m] \neq v$ **then**
    return 0
else
    return m

Altre definizioni – Stephenson, Barr

- Analizzare e organizzare i dati del problema in base a criteri logici;
- Rappresentare i dati del problema tramite opportune astrazioni;
- Formulare il problema in un formato che ci permetta di usare un "sistema di calcolo"
- Automatizzare la risoluzione del problema definendo una soluzione algoritmica, consistente in una sequenza accuratamente descritta di passi, selezionati da un catalogo ben definito di operazioni di base;
- Identificare, analizzare, implementare e verificare le diverse soluzioni avendo come obiettivo la ricerca della soluzione migliore secondo opportuni criteri;
- Generalizzare il processo di risoluzione del problema per poterlo trasferire ad un ampio spettro di altri problemi.

Chris Stephenson and Valerie Barr. Defining Computational Thinking for K–12. CSTA Voice, 7(2):1, 2011. Link

Strumenti intellettuali propri del CT

- Confidenza nel trattare la complessità.
- Ostinazione nel lavorare con problemi difficili.
- Tolleranza all'ambiguità (da riconciliare con il necessario rigore che assicuri la correttezza della soluzione).
- Abilità nel trattare con problemi definiti in modo incompleto.
- Abilità nel trattare con aspetti sia umani che tecnologici.
- Capacità di comunicare e lavorare con gli altri per il raggiungimento di una meta comune o di una soluzione condivisa.

Ritorno su questo punto...

The New York Times

U.S.

Reading, Writing, Arithmetic, and Lately, Coding

By **MATT RICHTEL** MAY 10, 2014

Il pensiero computazionale come linguaggio

"*A number of workshop participants advanced the idea that computational thinking could be better understood as a fundamental intellectual skill comparable to reading, writing, speaking, and arithmetic. Functionally, these fundamental skills are all means of describing and explaining complex problems and situations to others, and computational thinking serves the same purpose. In other words, computational thinking is comparable to other basic cognitive abilities that the average person in modern society is expected to possess.*"

National Academy of Sciences, Report of a Workshop on The Scope and Nature of Computational Thinking, 2010

Il pensiero computazionale come linguaggio

"*Computational thinking is more than programming, but only in the same way that language literacy is more than writing. They are both very important. Yes, it's more, but don't minimize programming just because it's more." He went on to say that programming is a particularly important form of expression, and that "programming, like writing, is a means of expression and an entry point for developing new ways of thinking.*"

National Academy of Sciences, Report of a Workshop on The Scope and Nature of Computational Thinking, 2010

Sommario

Denning – Due domande fondamentali

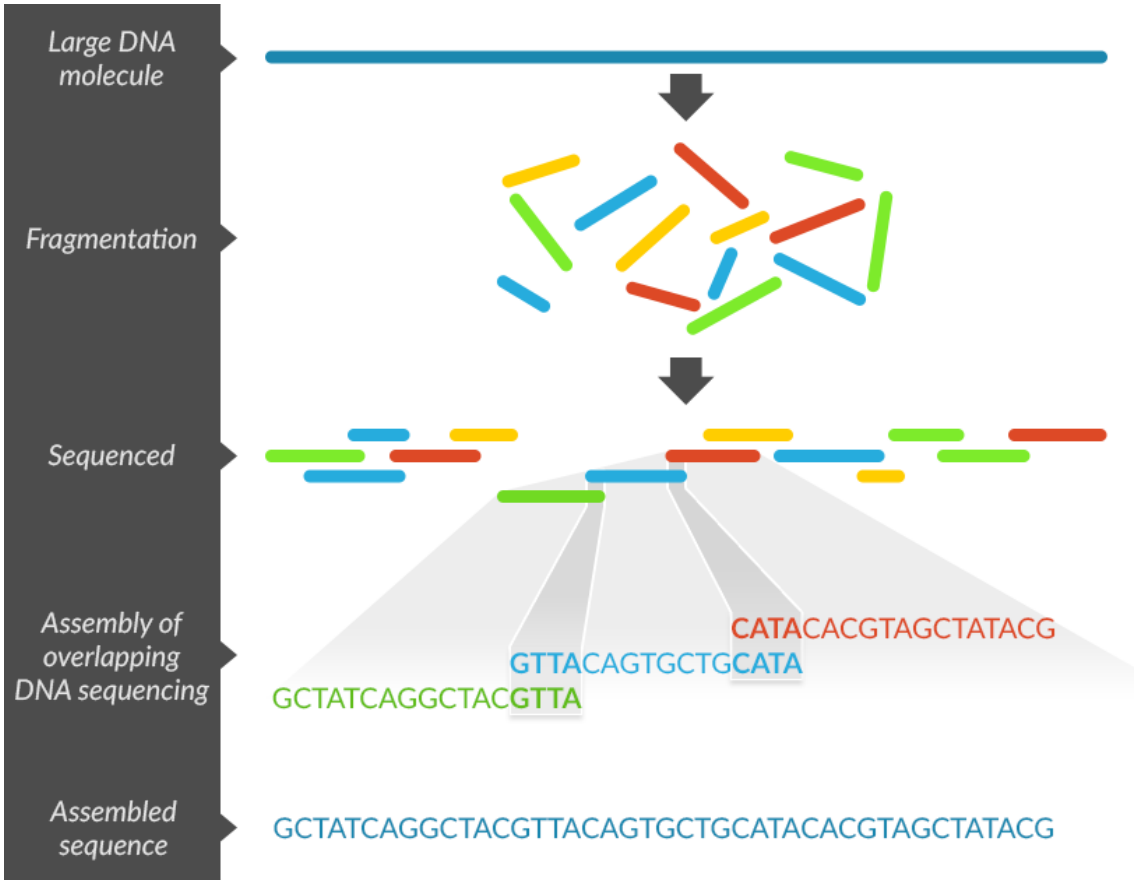
- Il CT è caratteristico della sola Informatica?
  - Negli anni '40, von Neumann scrisse che i computer non saranno semplicemente uno strumento per fare scienza, ma un metodo per fare scienza.
  - Ken Wilson – Nobel in Fisica – ha spinto per la creazione di dipartimenti di scienze computazionali nelle Università
  - "Computational science" vs Computer Science
- Il CT è una descrizione adeguata dell'Informatica?
  - Secondo Denning, il pensiero computazionale è solo una delle prassi dell'informatica
  - Altre pratiche: programming, system thinking, performance prediction, and approach to design.

Denning, P. "The profession of IT - Beyond computational thinking". *Communications of the ACM*, 52(6):28-30, 2009. [Link]

Biologia computazionale

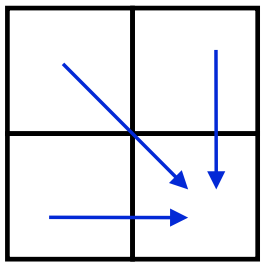
Computational genomics

- Algoritmo "Shotgun" per la sequenzazione del genoma
- Le sequenze di DNA sono stringhe in un linguaggio formale



- **ACGGCT**
- **CTCTGT**

↘ deriva da *i*-1,*j*-1
↓ deriva da *i*-1,*j*
→ deriva da *i*,*j*-1



i \ j		0	1	2	3	4	5	6
			C	**T**	**C**	**T**	**G**	**T**
0		0	0	0	0	0	0	0
1	**A**	0	↓ 0	↓ 0	↓ 0	↓ 0	↓ 0	↓ 0
2	**C**	0	↘1	→1	↘ 1	→1	→1	→1
3	**G**	0	↓ 1	↓ 1	↓ 1	↓ 1	↘2	→2
4	**G**	0	↓ 1	↓ 1	↓ 1	↓ 1	↘ 2	↓ 2
5	**C**	0	↘ 1	↓ 1	↘2	→2	↓ 2	↓ 2
6	**T**	0	↓ 1	↘2	↓ 2	↘ 3	→3	↘ 3

$$D[i,j] = \begin{cases} 0 & \text{se } i = 0 \vee j = 0 \\ D[i-1, j-1] + 1 & \text{se } i > 0 \wedge j > 0 \wedge p_i = t_j \\ \max\{D[i-1,j], D[i,j-1]\} & \text{se } i > 0 \wedge j > 0 \wedge p_i \neq t_j \end{cases}$$

Biologia computazionale

Computational biomodeling

- Reti di Petri

Computational neuroscience

- Simulazione

Computational evolutionary biology

- Ricostruzione dell'albero filogenetico

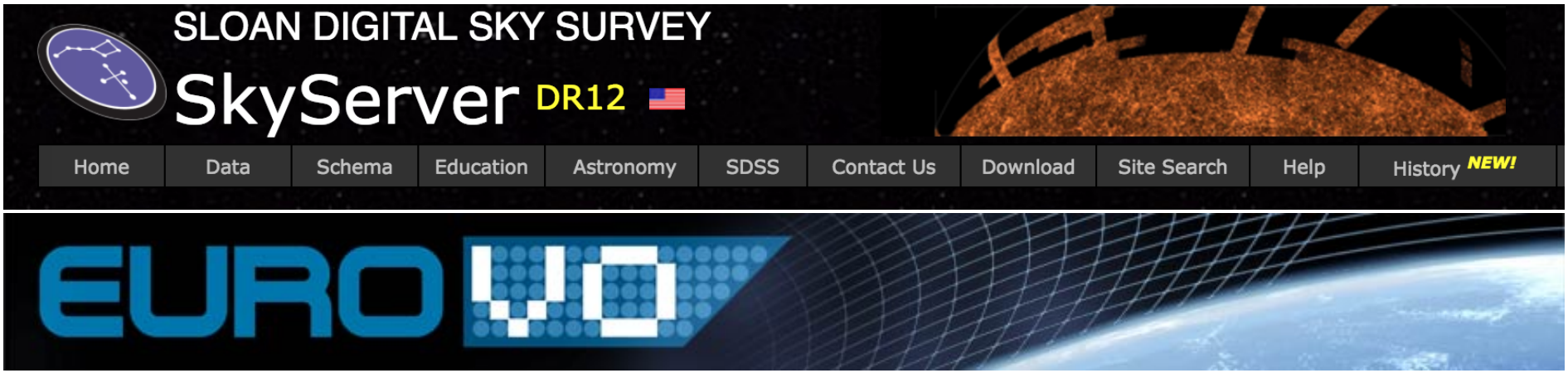
Computational proteinomics

- Relazioni tra proteine modellate come grafi

Astroinformatica

L'astroinformatica sviluppa metodi e strumenti derivate dalle scienze computazionali e dalla statistica per la ricerca e la didattica nell'astronomia basata su grandi quantità di dati

- Data mining, machine learning
- Iniziative di Virtual Observatory



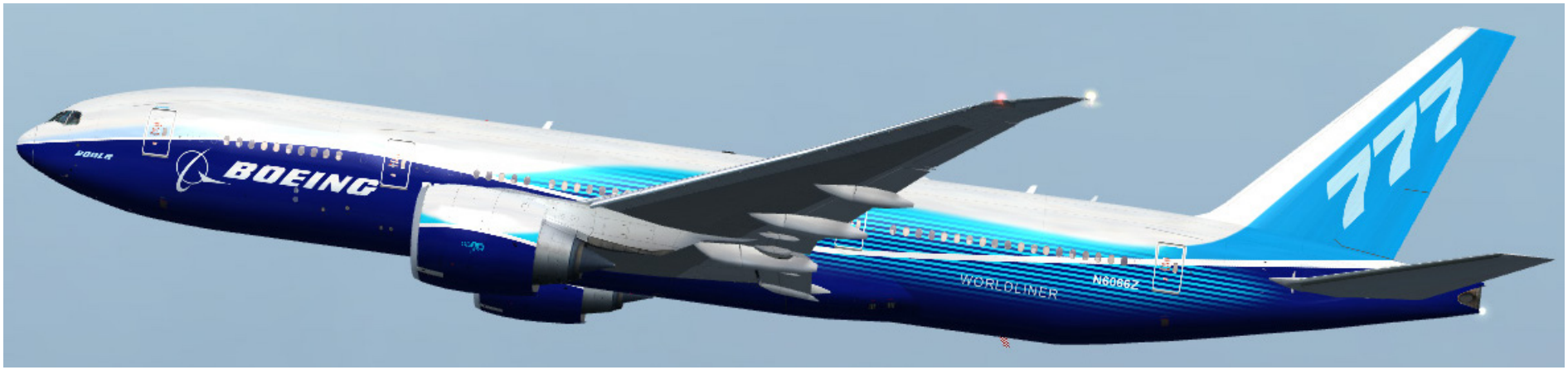
Astroinformatica

Daniele Gasparri, classe 1983, è laureato in astronomia all'università di Bologna. Appassionato di astronomia sin da bambino, nel 2007 è diventato il primo astronomo dilettante al mondo a scoprire il transito di un pianeta extrasolare.

Ingegneria computazionale

L'ingegneria computazionale si occupa dello sviluppo e dell'applicazione di modelli computazionali e di simulazione per risolvere problemi fisici complessi che occorrono nella progettazione e nell'analisi ingegneristica.

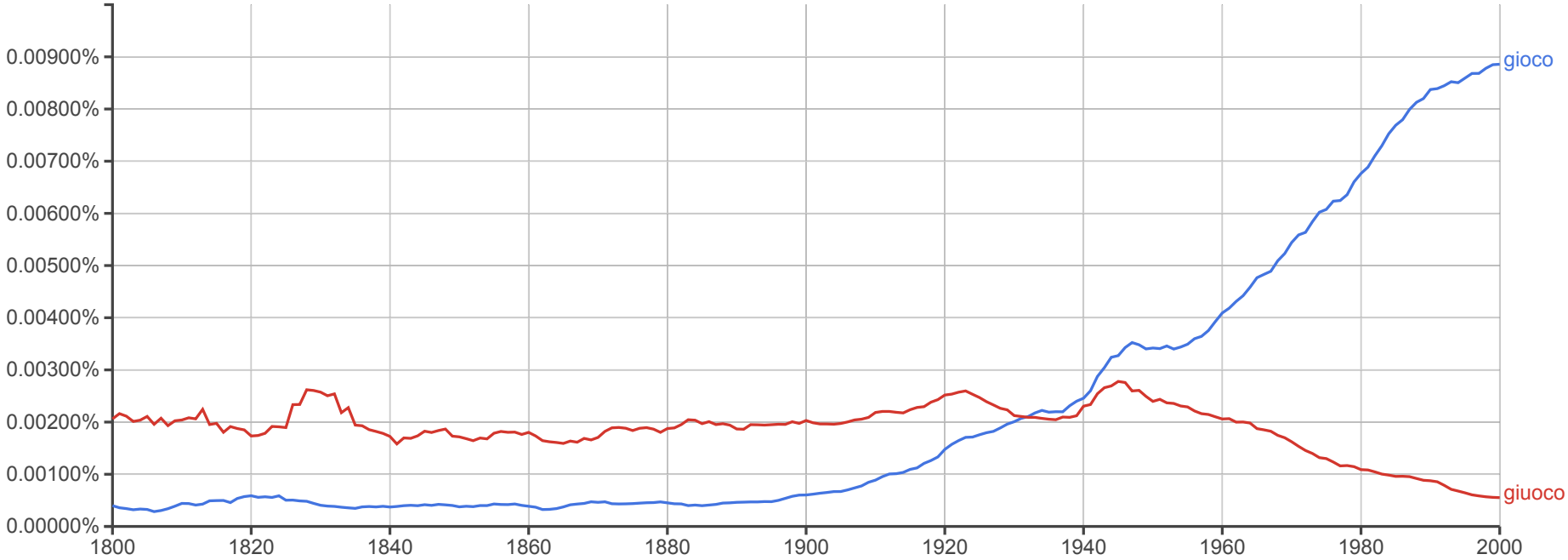
- High performance computing
- Modellazione e simulazione
- Algoritmi per la soluzione di problemi e nel continuo
- Analisi e visualizzazione di dati



Linguistica computazionale

Lo studio scientifico di un linguaggio da una prospettiva computazionale, allo scopo di fornire modelli computazionali per vari tipi di fenomeni linguistici.

- Natural language processing
- Google Translate, Apple Siri



Digressione: Nella vita quotidiana

- Cercare un nome in una lista ordinata alfabeticamente
  - Lineare: inizia dal principio
  - Logoritmico: inizia dal mezzo (binary search)
  - Log Log: inizia da una stima della posizione del nome (interpolation search)
- In coda al supermarket
  - Analisi delle prestazioni delle singole code, shortest job first
- Portare i bambini a calcio, nuoto, catechismo, scout
  - Problema del commesso viaggiatore
- Gestire la cucina di un ristorante
  - Gestione del parallismo
- Ristrutturare casa senza un direttore lavori
  - Ordinamento topologico su grafi

Sommario

Framework per il pensiero computazionale

Il laboratorio del MIT Media Lab ha sviluppato un framework operativo per l'insegnamento del pensiero computazionale

- Definizione
  - Concetti computazionali (computational concepts)
  - Prassi computazionali (computational practices)
  - Riflessioni computazionali (computational perspectives)
- Valutazione
  - Interviste basate su artefatti (progetti)
  - Scenari di progettazione
  - Documentazione di progetto
- Curriculum per il pensiero computazionale
  - Bilancio fra rigidità di percorso e libertà di esplorazione

Concetti computazionali

- Sequenza: identificare una serie di passi per compiere un'operazione
- Cicli: eseguire la stessa sequenza più volte
- Parallelismo: eseguire più cose contemporaneamente
- Eventi: una cosa causa l'altra
- Condizioni: prendere decisioni in base a condizioni
- Operatori: supporto per le operazioni matematica
- Dati: memorizzare, leggere, modificare dati
- Moduli: definire nuovi elementi del linguaggio
- Ricorsione: risolvere un problema in maniera ricorsiva (!)

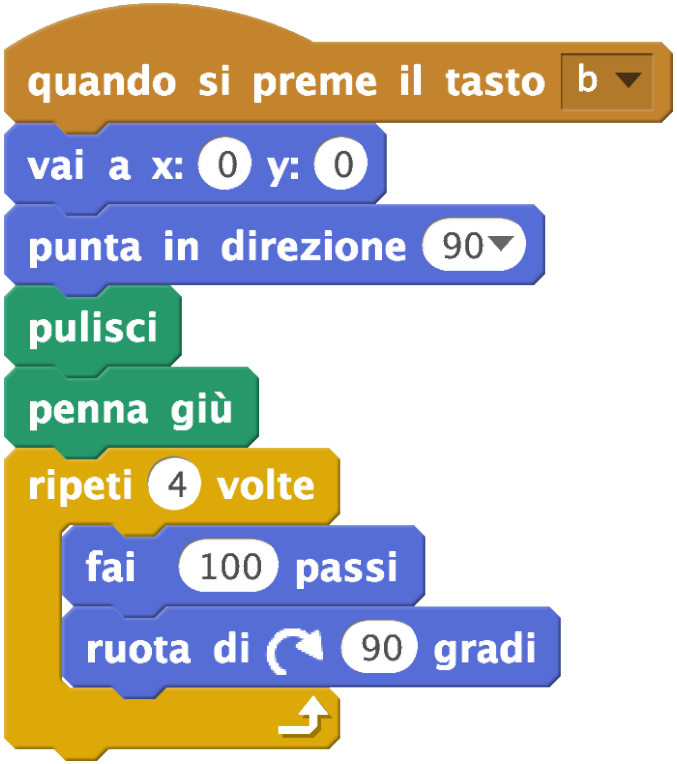
Sequenza (+ Eventi)


```
import turtle

wn = turtle.Screen()
tess = turtle.Turtle()
tess.color("blue")
tess.pensize(3)

tess.forward(100)
tess.left(90)
tess.forward(100)
tess.left(90)
tess.forward(100)
tess.left(90)
tess.forward(100)
tess.left(90)
```

Cicli



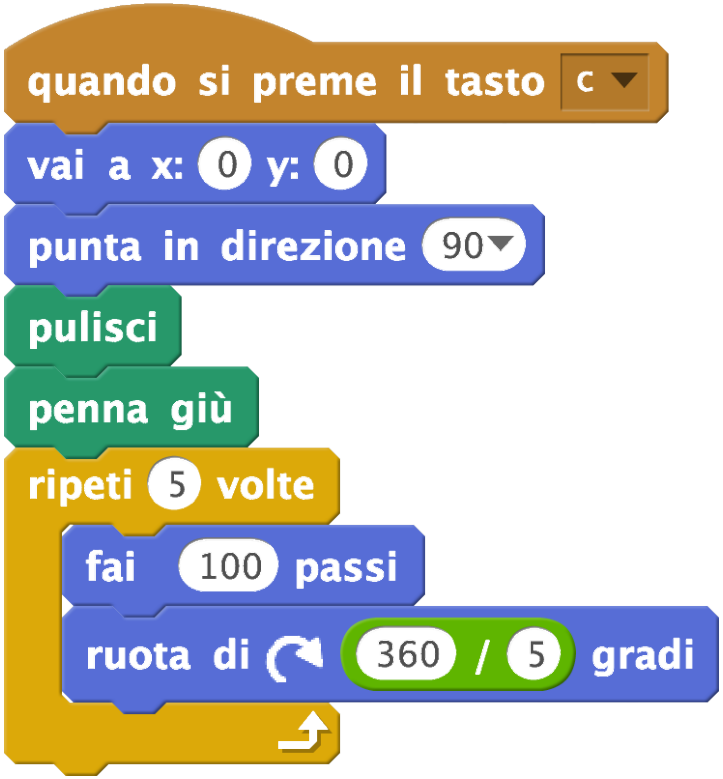

```
import turtle

wn = turtle.Screen()
tess = turtle.Turtle()
tess.color("blue")
tess.pensize(3)

for i in range(4):
  tess.forward(100)
  tess.left(90)

wn.exitonclick()
```

Operatori




```
import turtle

wn = turtle.Screen()
tess = turtle.Turtle()
tess.color("blue")
tess.pensize(3)

for i in range(5):
  tess.forward(100)
  tess.left(360/5)

wn.exitonclick()
```

Variabili

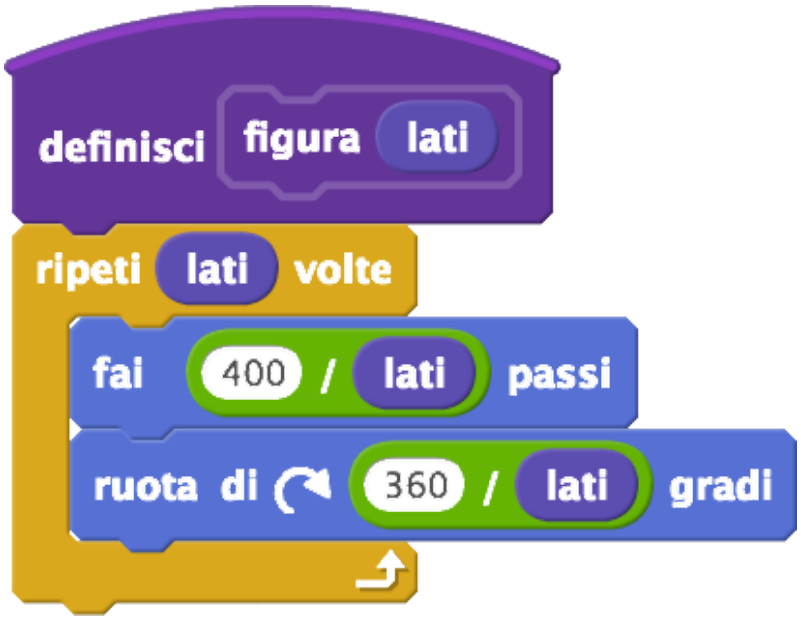

```
import turtle

wn = turtle.Screen()
tess = turtle.Turtle()
tess.color("blue")
tess.pensize(3)

lati = 6
for i in range(lati):
  tess.forward(400/lati)
  tess.left(360/lati)

wn.exitonclick()
```

Moduli




```
def figura(tess,lati):
  for i in range(lati):
    tess.forward(400/lati)
    tess.left(360/lati)
```

Condizioni


```
import turtle

risposta = input("Lati?")
if lati < 3:
  print("Almeno 3 lati")
  quit()
else:
  lati = risposta

wn = turtle.Screen()
tess = turtle.Turtle()
tess.color("blue")
tess.pensize(3)
figura(tess,lati)
wn.exitonclick()
```

Condizioni


```
import turtle

lati = 0
while lati < 2:
  risposta = input("Lati?")
  if risposta < 3:
    print("Almeno 3 lati")
  else:
    lati = risposta

wn = turtle.Screen()
tess = turtle.Turtle()
tess.color("blue")
tess.pensize(3)
figura(tess,lati)
wn.exitonclick()
```

Eventi, parallelismo

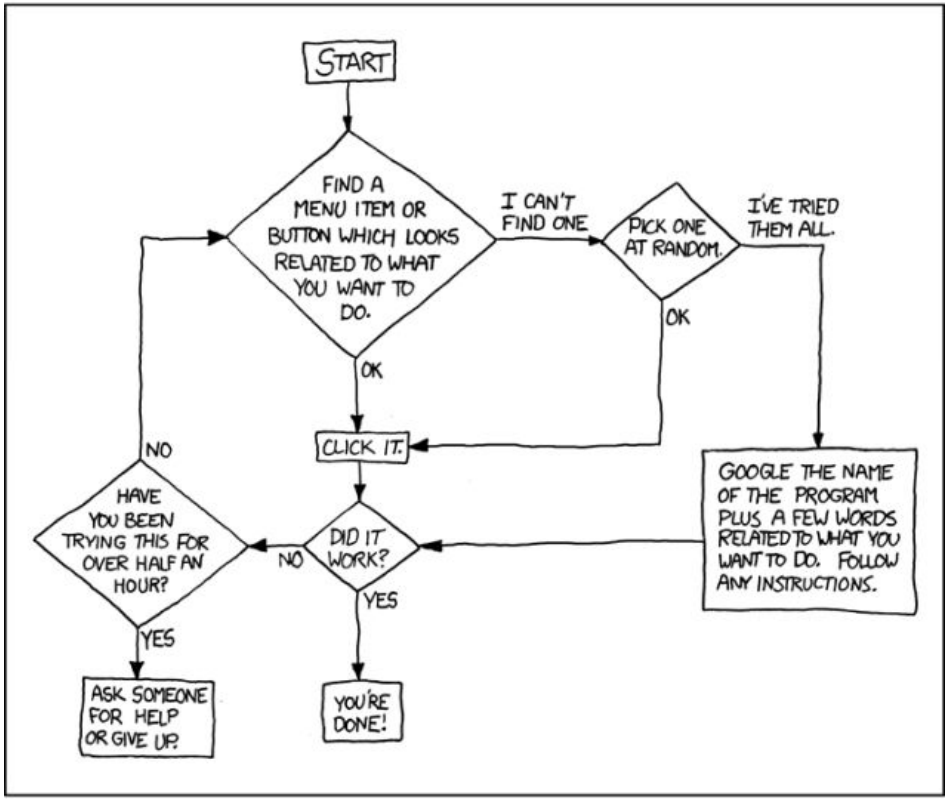
Prassi computazionali

- Sperimentare ed iterare: sviluppare un poco, poi provare, poi sviluppare ancora un po'
- Testare e fare debug: assicurarsi che le cose funzionino, identificare gli errori
  - Papert: gestione dell'errore
- Riutilizzare e remixare: creare qualcosa partendo da progetti esistenti o idee
  - Verso il mondo open source
- Astrarre e modularizzare: esplorare connessioni fra il tutto e le parti
  - Gestire la complessità

Debug

DEAR VARIOUS PARENTS, GRANDPARENTS, CO-WORKERS, AND OTHER "NOT COMPUTER PEOPLE."

WE DON'T MAGICALLY KNOW HOW TO DO EVERYTHING IN EVERY PROGRAM. WHEN WE HELP YOU, WE'RE USUALLY JUST DOING THIS:



PLEASE PRINT THIS FLOWCHART OUT AND TAPE IT NEAR YOUR SCREEN. CONGRATULATIONS; YOU'RE NOW THE LOCAL COMPUTER EXPERT!

Riflessioni computazionali

- Espressività
  - Comprendere che la computazione è un mezzo di espressione
  - "Posso creare"
- Connessione
  - Riconoscere il poter di creare con e per gli altri
  - "Posso fare di più se ho accesso agli altri"
- Porsi domande
  - Sentirsi in grado ("empowered") di porsi domande sul mondo
  - "Posso (utilizzare la computazione per) pormi domande per capire come funziona il mondo (in senso computazionale)"

Valutazione

"Interviste" basate su artefatti (progetti)

- Protocollo di valutazione
- Griglie di valutazione

Scenari di progettazione

- Computational Thinking Design Scenarios (Studio)
- `https://scratch.mit.edu/studios/573426/`

Documentazione di progetto

- Scrittura di un diario
- Commenti nel progetto

Sommario

Contenuti

Laboratorio di Scratch

- Concetti computazionali
- Prassi computazionali
- Riflessioni computazionali

Seymour papert e il costruzionismo

- L'attualità del pensiero di Papert
- `code.org` non è una risposta adeguata al bisogno di pensiero computazionale...
- ... ma forse al momento è l'unica risposta possibile
- `code.org` come punto di partenza, non di arrivo

Contenuti

Computer Science Unplugged

- L'approccio all'informatica senza computer
- Può essere un punto di partenza per comprendere quali sono i principi fondanti dell'Informatica

Valutazione

- Scenari di progettazione
- Valutazione funzionale
- Interviste, scrittura di un diario

Questioni aperte: Quale curriculum?

- Quali sono i modi effettivi per imparare (insegnare) il pensiero computazionale?
- Quali sono i concetti analoghi a:
  - Numeri (Materne)
  - Aritmetica (Primaria)
  - Algebra (Secondaria I grado)
  - Analisi (Secondaria II grado)
- Come integrare questo curriculum con "computer fluency"?
  - Ma poi, serve?
- Come integrare questo curriculum nei piani didattici?
  - E' necessario separarla dalle altre materie?
  - "La coperta è troppo corta" (cit. Pavani)

Questioni aperte: Servirà?

- Critiche al coding
  - Tutti questi informatici, serviranno?
  - Realizzare sistemi software non è un gioco
- Critiche all'approccio visuale