Metodologie e tecnologie didattiche per l'informatica

Luca Barra

Anno accademico 2023/2024

Indice

Capitolo 1

Introduzione

Il corso verterà sul ruolo dell'informatica nella scuola. Si dovranno progettare delle attività didattiche con determinati argomenti, modalità, svolgimenti, valutazioni, etc..

Note:-

Poichè l'istruzione nella scuola è un tema in costante divenire e soggetto a cambiamenti, anche su base politica, questi appunti potrebbero discostarsi dallo stato attuale.

1.1 Che cos'è l'informatica?

L'*informatica* presenta molti problemi, a partire dalla sua definizione. Essa non è comunemente percepita come una disciplina/scienza. In vari campi, come la matematica, si tende a dire "io non so nulla", mentre per l'informatica qualunque persona che sa usare un software propende per "io l'informatica la capisco bene".

1.1.1 Problema 1: il termine informatica

Su molti siti di e-commerce (unieuro, Carrefour, etc.) i televisori, gli elettrodomestici, etc. compaiono sotto il termine "*informatica*". Questo fenomeno non si presenta nei confronti delle altre scienze: se si va in un negozio non si trovano le categorie "*chimica*" o "*fisica*".

Dunque, possiamo dire che l'informatica è una scienza? Se si fanno delle ricerche sul web non sempre si trova informatica nelle discipline scientifiche.

Tuttavia, a livello accademico (nelle università), l'informatica è presente nella categoria "*scienze della natura*".

Definizione 1.1.1: L'informatica

L'informatica è una scienza che studia i principi e i metodi per l'elaborazione automatica dell'informazione.

- *Elaborazione:* le informazioni vengono elaborate anche in contesti in cui l'informatica non c'entra;
- *Automatica:* elaborazione delle informazioni da parte di un *interprete meccanico*;
- *Informazione:* cos'è? come la si può rappresentare in modo da farla elaborare dall'interprete meccanico?

Dobbiamo smetterla di pensare che l'informatica riguardi i computer

Micheal R. Fellows, Ian Parberry (1993)

L'informatica non riguarda i computer, esattamente come l'*astronomia* non riguarda i *telescopi*, la *biologia* i *microscopi*, o la *chimica* i *vetrini* e le *provette*.

Il *computer* è solo uno strumento, non il fine. La scienza non riguarda gli strumenti, ma come li usiamo e ciò che scopriamo quando lo facciamo.

1.1.2 Problema 2: il termine digitale

C'è distinzione tra *informatico* e *digitale*. Ma che cosa sono le "*competenze digitali*"?

Definizione 1.1.2: Digitale

Digitale è un termine che si riferisce a tutte le *tecnologie* basate su computer.

Il termine digitale indica la rappresentazione di un *dato* mediante un simbolo numerico, mentre informatico si riferisce alla capacità di *elaborazione automatica* dei dati resa possibile dai metodi e dalle teorie dell'informatica. Le competenze digitali riguardano l'utilizzo del dispositivo che elabora i dati. Ci possono essere ottimi informatici che hanno competenze digitali mediocri. Bisogna anche tener conto del fatto che le tecnologie attuali sono molto più fruibili dal pubblico rispetto al XX secolo. Per cui i cosidetti "*nativi digitali*" sono in grado di usare dispositivi e software semplici, ma non applicativi complessi.

Le competenze :

- *Digitali* sono *competenze operative*;
- *Informatiche* sono *competenze scientifiche*.

Note:-

Servono e vanno insegnate *entrambe*, ma hanno scopi diversi.

1.2 Perchè insegnare informatica?

Domanda 1.1

In primo luogo: perchè si insegnano le *scienze naturali* nella scuola primaria?

Risposta: Alla base la motivazione che viene data è: noi viviamo in un mondo e dobbiamo avere una chiave di interpretazione per esso.

Domanda 1.2

Perchè si studia *chimica* nelle scuole secondarie di secondo grado?

Risposta: Bisogna sapere che cosa compone la materia che ci circonda.

Domanda 1.3

E la *fisica*?

Risposta: Si studia per avere una maggior comprensione dei princìpi fisici che regolano il mondo.

Domanda 1.4

E allora l'*informatica*?

Risposta: Come le scienze naturali danno una chiave di lettura del mondo fisico, l'informatica offre una chiave di lettura del mondo digitale.

1.2.1 L'algoritmo

L'algoritmo, nell'immaginario comune, è considerato come un essere vivente, in particolar modo come un *mostro* con potere di decisione assoluto su ogni altra creatura. Il problema è che l'informatica è una scienza giovane, per cui è più soggetta ai pregiudizi.

1.2.2 Il manifesto di Vienna

Il *manifesto di Vienna per l'umanesimo digitale* è una presa di coscienza riguardo alla diffusione massiva delle macchine e del digitale. Ci sono molti motivi per cui la rivoluzione digitale potrebbe risultare un fallimento o non democratica. Per risolvere questi problemi sono necessari programmatori, insegnanti di informatica bravi che riescano a interfacciarsi con il mondo in modo da creare un dialogo.

Note:-

L'insegnamento dell'informatica deve partire dalla scuola primaria e deve avere carattere interdisciplinare.

1.3 L'informatica è una scienza?

Ci sono tre possibili paradigmi su cui basare l'informatica.

1.3.1 I paradigmi

	Paradigma razionalista	**Paradigma tecnocratico**	**Paradigma scientifico**
Metodo	l'informatica fa parte della matematica ragionamento deduttivo	l'informatica fa parte dell'ingegneria test di affidabilità	l'informatica fa parte delle scienze teoria (formulazione di ipotesi) → deduzione → validazione empirica
Ontologia	testo del programma = espressione matematica programma = oggetto matematico	testo del programma = aggregato di meri dati il programma è un'entità immateriale, perciò è inutile presupporne l'esistenza	testo del programma assimilabile a una rete di neuroni programma assimilabile a un processo mentale
Epistemologia	specifiche formali dei programmi complete la correttezza è ben definita conoscenza a-priori	specifiche formali dei programmi impraticabili (o impossibili) ha senso parlare solo di affidabilità conoscenza a-posteriori (misure di affidabilità)	specifiche formali dei programmi incomplete proprietà certe / proprietà stimate conoscenza a-priori (parziale) e a-posteriori (probabilistica)

Definizione 1.3.1: Paradigma razionalista (o matematico)

Per il *paradigma razionalista* si formalizza un linguaggio che garantisce certe proprietà (il tutto prima dell'esecuzione).

Definizione 1.3.2: Paradigma tecnocratico (o ingegneristico)

Per il *paradigma tecnocratico* si devono fare molti test di affidabilità con diversi input (unit test, a run time).

Definizione 1.3.3: Paradigma scientifico

Per il *paradigma scientifico* si deve validare empiricamente la correttezza di un programma.

Ognuno di questi paradigmi può essere usato per offrire una visione differennte dell'informatica:

- alcune attività sono principalmente scientifiche: *algoritmi sperimentali*;
- alcune attività sono principalmente ingegneristiche: *design*, *sviluppo*, etc;
- alcune attività sono principalmente matematiche: *calcolo di complessità*, *metodi formali*.

Perchè è importante pensarci? È importante pensare al prioprio modo di vedere l'informatica perchè ciò andrà a in cui si andrà a insegnare. Come si è visto la risposta è che l'informatica contiene in sè un po' di ogni paradigma, ma ci sono parti che possono essere ritenute, personalmente, più importanti.

Definizione 1.3.4: Anima matematica

Rimanda al razionalismo filosofico. La *conoscenza pura* è più affidabile dei propri sensi (*conoscenza a posteriori*). L'unica metodologia accettabile è il ragionamento deduttivo. Il testo di un programma è un'espressione matematica.

Definizione 1.3.5: Anima ingegneristica

Rimanda all'empirismo inglese per cui l'esperienza è la base della conoscenza. È impossibile stabilire una conoscenza a priori sui programmi, l'informatica teorica è pura speculazione. Il metodo per valutare la qualità di un programma è il *testing* (abbinato alla statistica). Il testo di un programma è un aggregato di dati.

Definizione 1.3.6: Anima scientifica

Rimanda al metodo scientifico sperimentale. Si effettuano dei test basandosi su delle ipotesi. Il testo di un programma è assimilato a un *processo biologico* e la sua istanziazione (esecuzione) è assimilata a un *processo mentale*.

Esempio 1.3.1 (Un algoritmo di ordinamento spiegato in una scuola secondaria di secondo grado)
Un algoritmo di ordinamento può essere presentato in modi diversi a seconda dell'approccio scelto.
Domande matematiche:

- L'algoritmo riesce a ordinare qualsiasi sequenza di dati, a prescindere dalla situazione iniziale, con eventuali ripetizioni?
- Gli algoritmi sono tutti uguali o ce ne sono di più efficienti?
- C'è un limite oltre cui non esiste un algoritmo pù veloce?

Domande scientifiche:

- Come si possono misurare sperimentalmente i tempi di calcolo e con quale attendibilità?
- I tempi confermano i risultati teorici?
- Quali aspetti sono generali e quali contingenti?
- Quick Sort è sempre meglio di Insertion Sort?

Domande ingegneristiche:

- È possibile migliorare le prestazioni di un algoritmo?
- Come si organizza il processo di sviluppo?
- Quali sono le condizioni ottimali?

1.4 Quali sono i criteri per definire una scienza?

- Organizzata per capire, sfruttare e affrontare fenomeni pervasivi;
- Include sia processi naturali che artificiali;
- Ha un corpo ben strutturato;
- C'è un impegno per la scoperta e validazione dei principi;
- I risultati ottenuti sono riproducibili;
- Si possono falsificare ipotesi e modelli;
- Si riescono a fare ipotesi affifdabili, alcune sorprendenti.

Capitolo 2

L'informatica a scuola

2.1 L'organizzazione del sistema scolastico italiano

I gradi di istruzione in Italia sono:

1. Scuola dell'infanzia;
2. Scuola di primo grado (*ex elementari*);
3. Scuola secondaria di primo grado (*ex medie*);
4. Scuola secondaria di secondo grado (*ex superiori*);
5. Formazione superiore (università, master, dottorato).

2.1.1 Come si accede all'insegnamento?

Per diventare docenti è necessario avere:

- un titolo di accesso all'insegnamento (Laurea, Diploma, etc.);
- l'*abilitazione all'insegnamento*.

Chi possiede solo il primo requisito può essere inserito nelle graduatorie di istituto di III fascia per incarichi di supplenza a *tempo determinato*[1].

Quando si consegue l'abilitazione si può accedere alle graduatorie di istituto di II fascia e si può partecipare al *concorso*.

Note:-

Dalle graduatorie dei concorsi, annualmente, si attinge per l'immissione in ruolo a tempo indeterminato.

Definizione 2.1.1: Titoli di accesso all'insegnamento per la scuola dell'infanzia e primaria

Per accedere all'insegnamento per la scuola dell'infanzia e per la scuola primaria i titoli di accesso sono: Laurea in scienze della formazione primaria, Diploma di Istituto Magistrale o Diploma di Liceo Socio-Pedagogico conseguiti entro l'anno scolastico (2001-2002)[a].

[a] Tutti questi titoli sono anche abilitanti

[1] In realtà i criteri vengono aggiustati in base all'istituto, abbassando i requisiti

Definizione 2.1.2: Titoli di accesso all'insegnamento per la scuola secondaria di I e II grado

Per accedere all'insegnamento per la scuola secondaria di I e II grado è necessario avere un determinato tipo di Laurea appartenente a una *classe di concorso*. In più bisogna acquisire 60 CFU nei settori: antro/psico/pedagogici/metodologico[a].

[a]Di cui 16 CFU di disciplina

2.2 Cosa si può insegnare con la laurea magistrale in informatica?

La *Laurea magistrale in informatica* dà accesso "diretto[2]" alle classi di concorso:

- A-41, Scienze e tecnologie informatiche;
- A-47, Scienze matematiche applicate.

Mentre, prendendo dei crediti extra in altri settori, si può accedere a:

- A-26, Matematica;
- A-28, Matematica e scienze;
- A-40, Scienze e tecnologie elettriche ed elettroniche.

Note:-

Nelle scuole, per via di carenza di professori, molte cattedre di informatica sono affidate a docenti di matematica. Questo fa si che ci sia una preparazione molto eterogenea a seconda della scuola.

Definizione 2.2.1: Le indicazioni nazionali

Per ogni scuola e per ogni indirizzo sono presenti le *indicazioni nazionali* in cui si elencano le competenze presupposte in ingresso, gli obiettivi di apprendimento e le competenze attese in uscita. Gli insegnanti devono progettare le loro attività didattiche a partire da queste indicazioni.

[2]Se si ha l'abilitazione

Capitolo 3

Le teorie dell'apprendimento

L'impostazione che si dà a un'attività didattica si basa su:

- una *teoria dell'apprendimento*;
- l'idea, personale, che si ha di che cosa sia la *conoscenza*.

3.1 I paradigmi di apprendimento

Definizione 3.1.1: Teorie dell'apprendimento

Le *teorie dell'apprendimento* descrivono come le persone imparano. Esse sono prodotte da psicologia, pedagogia e filosofia.

Definizione 3.1.2: I paradigmi dell'apprendimento

I *paradigmi dell'apprendimento* sono classificazioni delle teorie in base ai loro tratti comuni.

Principalmente si individuano due macro-categorie:

- istruttivismo;
- costruttivismo.

3.2 Comportamentismo

Definizione 3.2.1: Comportamentismo

Si vuole modellare un *comportamento desiderabile*. L'apprendimento veniva valutato in base ai cambiamenti nel comportamento degli alunni. Gli studenti sono una *tabula rasa* che ricevono passivamente le informazioni per ripetizione. Si modella il comportamento tramite *rinforzi*[a].

[a] Punizioni corporali

Note:-

Questo è un approccio istruttivista il cui focus è sulla trasmissione della conoscenza, sulla strutturazione e presentazione dei contenuti, e non sullo studente, che viene visto come un recipiente da riempire.

Focus:

- Filosoficamente si ritiene vera l'esistenza di una *realtà oggettiva* che può essere imparata;
- La conoscenza è una *rappresentazione mentale* della realtà;
- C'è il presupposto che il mondo e la conoscenza esista anche se nessun essere umano li percepisce;
- Il linguaggio è il mezzo di trasporto della conoscenza.

3.3 Cognitivismo

Definizione 3.3.1: Cognitivismo

Il cognitivismo supera l'idea di osservare solo i comportamenti esterni. Si elaborano alcune teorie:

- il *carico cognitivo* è il carico di lavoro mentale necessario per eseguire un compito;
- gli *schemi* e i *modelli mentali*.

Note:-

Lo scopo dell'educazione è quello di far ricordare e applicare la conoscenza.

Focus:

- Gli studenti sono degli *elaboratori di informazione*;
- Gli insegnanti devono facilitare l'elaborazione.

Note:-

Questo approccio è ambivalente: può essere sia istruttivista che costruttivista.

3.4 Costruttivismo

Definizione 3.4.1: Costruttivismo

Il costruttivismo[a] si basa sullo *scetticismo*:

- la conoscenza deriva dall'esperienza[b];
- non c'è modo di sapere la "vera" verità.

[a] Ideato da Jean Piaget
[b] Soggettiva

Note:-

- Il concetto di verità è illusorio;
- Non si può confrontare la rappresentazione con l'oggetto rappresentato;
- Si deve ricorrere alla "viabilità".

Definizione 3.4.2: Viabilità

Una conoscenza viene definità *viabile* se ha funzionato bene nelle esperienze precedenti.

Corollario 3.4.1 Criteri di viabilità

- Azioni fisiche: è viabile tutto ciò che porta allo scopo scelto;

- Piano concettuale: non ci deve essere contradditorietà o non coerenza logica.

Esempio 3.4.1 (Alcuni metodi di apprendimento)

- Assimilazione: si incorpora un concetto in uno schema già acquisito;
- Accomodamento: si modifica la struttura cognitva in relazione a cose nuove.

Definizione 3.4.3: Apprendimento attivo

L'*apprendimento attivo* è un ampio insieme di metodologie didattiche che coinvolgono gli studenti come parte *attiva* dell'apprendimento insieme all'insegnante.

Definizione 3.4.4: Costruttivismo cognitivo

Nel *costruttivismo cognitivo* lo scopo dell'educazione è quello di permettere agli studenti di creare nuova conoscenza. L'apprendimento è il processo di costruzione del signidìficato. L'insegnante ha solo lo scopo di facilitare la scoperta offrendo le risorse necessarie.

Corollario 3.4.2

Lo sviluppo cognitivo è consentito dall'ambiente culturale e l'apprendimento si deve svolgere con l'aiuto altrui. La *ZPS*[a] descrive le possibili zone di sviluppo di un bambino:

- zona 1 (o zona di sviluppo attuale): lo studente può apprendere da solo;
- zona 2/ZPD (o zona di sviluppo prossimale): lo studente può apprendere solo se supportato;
- zona 3 (o zona di sviluppo potenziale): lo studente non può ancora apprendere nè da solo nè supportato.

ZPS

[a] Zona di sviluppo prossimale

Note:-

Un insegnante può agire solo sulla zona 2.

Definizione 3.4.5: Socio-costruttivismo

Nel *socio-costruttivismo* lo scopo dell'educazione è quello di permettere agli studenti di creare nuova conoscenza insieme. Si dà enfasi sulle relazioni umane.

3.4.1 Didattica costruttivista

Le moderne teorie sono molto spesso basate sul costruttivismo:

- *active learning*: le pratiche in cui gli studenti svolgono attivamente qualcosa e riflettono;
- *productive failure*: si chiede agli studenti di svolgere problemi mal strutturati o difficili. In seguito si forniscono le conoscenze necessarie;
- *problem-based learning* (PBL): problema realistico;
- *inquiry-based learning*: gli studenti formulano domande, raccolgono dati, li analizzano, provano a spiegarli e creano conoscenza teorica.

Note:-

Il productive failure funziona una volta sola, quindi va usato come "jolly".

Caratteristica della didattica costruttivista:

⇒ l'apprendimento non avviene attraverso fasi standard;

⇒ ogni studente deve avere la possibilità di stabilire il proprio percorso;

⇒ l'insegnante deve indirizzare;

⇒ le parole e le azioni del docente sono strumenti per apprendere;

⇒ si dà priorità all'esperienza diretta piuttosto che alla lezione tradizionale.

Note:-

Ovviamente l'esperienza diretta va gestita dall'insegnante.

Compiti del docente:

⇒ accertare le pre-concezioni degli alunni;

⇒ far emergere concezioni sbagliate;

⇒ ristabilire le idee mediante ipotesi e tentativi;

⇒ far elaborare una nuova interpretazione coerente a quella socialmente condivisa.

3.5 Costruzionismo

Definizione 3.5.1: Costruzionismo

Il *costruzionismo* si basa sull'idea costruttivista di strutture di conoscenza. A ciò viene aggiunta l'idea che ciò accade nei contesti in cui si è coinvolti nella costruzione di un'entità (artefatto).

Note:-

Il costruzionismo è stato ideato da Seymour Papert, l'inventore di LOGO.

I ragazzi si avventurano nell'esplorazione di come loro stessi pensano. L'esperienza può essere esaltante: pensare sul pensare trasforma i ragazzi in epistemologi, un'esperienza rara anche per gli adulti.

Seymour Papert

Capitolo 4

Progettazione della didattica

4.1 Le fasi, gli snodi e gli indicatori

Ogni attività didattica può essere costituita da una o più fasi. Ogni fase a sua volta deve essere svolta secondo uno schema:

1. *consegna*: si introduce la descrizione del compito da svolgere. Deve essere progettato per far emergere dubbi e domande agli studenti;
2. *svolgimento*: gli studenti devono svolgere l'attività da soli, in coppia o in gruppo. Il compito dell'insegnante è quello di controllare lo svolgimento della consegna e favorire la discussione;
3. *discussione*: ogni studente, coppia o gruppo presenta alla classe la propria soluzione e la discute. Il docente deve far emergere le idee dei singoli. Chiunque presenti una critica a una soluzione deve adeguatamente motivarla;
4. *conclusione*: la classe deve essere messa allo stesso livello in base a quanto detto nel punto precedente. L'insegnante si occupa di riassumere i risultati ottenuti.

Note:-

La divisione di un'attività in fasi è utile per controllare che ogni studente abbia appreso le competenze necessarie.

Definizione 4.1.1: Snodi

Gli *snodi* sono i passaggi cognitivi per rispondere correttamente alle domande, svolgere un compito o risolvere un problema.

Esempio 4.1.1 (Snodi)

- Comprendere come effettuare una determinata azione;
- Eseguire un compito;
- Trovare un modo per risolvere un problema.

Note:-

Uno snodo non è una fase o una sottofase.

Definizione 4.1.2: Indicatori

Gli *indicatori* servono per "indicare" se uno snodo è stato superato o meno.

Esempio 4.1.2 (Indicatori)

- Frasi;
- Domande sulla fase appena svolta;
- Commenti sull'attività.

4.2 Obiettivi formativi

Definizione 4.2.1: Obiettivi formativi

Gli *obiettiv formativi* sono ciò che deve rimanere come apprendimento. Alcune attività possono essere funzionali all'apprendimento in sè pur non avendo contenuti da apprendere.

Note:-

Gli obiettivi dell'insegnamento possono essere diversi dagli obiettivi dell'attività.

Obiettivi formativi:

- *conoscenze* (K): cosa ci si aspetta che l'alunna conosca dopo l'attività;
- *abilità* (A): cosa ci si aspetta che l'alunno abbia imparato a fare;
- *competenze* (C): come ci si aspetta che l'alunno sappia applicare conoscenze e abilità in un ambiente nuovo, per risolvere un problema.

Esempio 4.2.1 (Obiettivi)

Lo studente è in grado di <verbo di azione> + <oggetto> + [<specifiche>].

⇒ Verbo: indica la prestazione attesa. Osservabile e misurabile;

⇒ Oggetto: la conoscenza che deve essere acquisita;

⇒ Specifiche: specificano condizioni aggiuntive, tempo, precisione con cui la prestazione deve essere eseguita. Sono opzionali.

Note:-

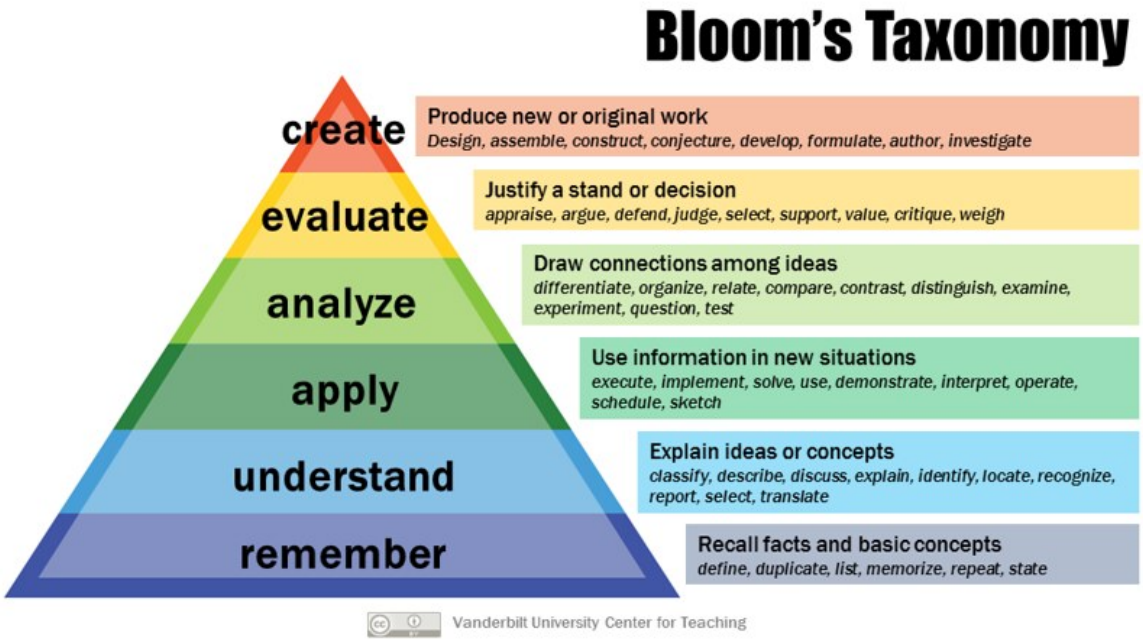
I verbi non sono casuali, ma sono individuati dalla tassonomia di Bloom

Definizione 4.2.2: Tassonomia di Bloom

La *tassonomia di Bloom* è un modello di classificazione degli obiettivi educativi che si concentra sui processi cognitivi necessari per completare un compito educativo.

Note:-

La tassonomia di Bloom si articola su più livelli in cui, per proseguire, è necessario aver padroneggiato il piano precedente.



Note:-

Questa tassonomia va integrata con le indicazioni nazionali, citate in un capitolo precedente.

Esempio 4.2.2 (I codici segreti con le vernici)

Domanda: come fa un computer a sussurrare il numero di una carta di credito a un altro computer proteggendolo da tutti gli altri computer connessi alla rete?
Per spiegare ciò (lo scambio di chiavi di Diffie-Hellman[a]) si utilizza il trucco delle vernici mescolate:

- ognuno ha a disposizione molti colori con nomi precisi;
- ognuno ha un set di vernici identico agli altri;
- ognuno ha sufficienti barattoli;
- ognuno può mescolare le vernici non visto dagli altri.

In questo esempio non ci sono comunicazioni segrete, ma solo pubbliche. Il trucco consiste nel fatto che per separare i colori bisogna sapere con quali colori sono formati. Questa è un'attività semplice ma interessante per spiegare il concetto di chiave pubblica e chiave privata.

[a]Spiegato in dettaglio nel corso di "Sicurezza"

Capitolo 5

Problem solving

5.1 Formulare e comprendere i problemi

Definizione 5.1.1: Compiti di realtà

Negli ultimi anni si fanno svolgere dei *compiti di realtà*, ossia dei problemi basati sulla realtà.

Definizione 5.1.2: Obblighi impliciti

Spesso i bambini tendono a limitarsi pensando a come ci si aspetta che loro debbano rispondere.

Esempio 5.1.1 (Nonna Rosa e la spesa)

Testo: Nonna Rosa vuole realizzare una macedonia alla frutta per i suoi nipotini. Le servono 2 kg di albicocche e 3 kg di pesche. Va al mercato per acquistarle. Nel banco della signora Agata (banco A) le pesche costano 1 € al kg e le albicocche 2 € al kg. Nel banco del signor Bruno (banco B) le pesche costano 2 € al kg e le albicocche 1 € al kg. Come è più conveniente fare l'acquisto? Quanto spenderà Nonna Rosa?

Risposta semplice: il procedimento che la maggior parte dei bambini è eseguire il calcolo per il banco A e il calcolo per il banco B e scegliere il banco in cui costa meno.

Risposta creativa: si prendono le pesche nel banco A e le albicocche nel banco B.

Spiegazione: solitamente i bambini danno una risposta semplice perchè sono stati abituati al fatto che i problemi vengono posti in un certo modo anche se nel testo non è scritto che si debba scegliere un banco.

Note:-

È importante pensare al come enunciare i problemi. La formulazione di un problema in quanto tale deve modellare un bisogno reale[a]. Per progettare un compito di realtà bisogna comprendere le possibili interpretazioni dello stesso.

[a] il problema di nonna Rosa insegna che con la matematica si può risparmiare

Esempio 5.1.2 (Come porre i problemi)

Formulazione puramente algoritmica: problemi che nella formulazione si riferiscono direttamente alle strutture dati e variabili che verranno usate nel programma progettato per risolverli.

Formulazione algoritmico-narrativa: problemi che nella formulazione non si riferiscono esplicitamente alle strutture dati e variabili che verranno usate nel programma progettato per risolverli. La formulazione è inserita in una storia/narrazione. Per risolvere il problema in questo caso si deve prima interpretare il testo.

Compito	Formulazione puramente algoritmica	Formulazione algoritmico-narrativa
Trovare il massimo in una lista di numeri	Scrivere un programma/algoritmo in pseudo codice che ritorna il numero massimo in una lista di numeri	Si svolge una gara di salto in lungo. Trovare l'atleta che ha percorso la distanza maggiore
Dire se un array è ordinato oppure no	Scrivere un programma/algoritmo in pseudo codice che ritorna vero se un dato array è ordinato falso altrimenti	Si ha un mazzo di carte e si vuole sapere se le carte sono messe in ordine
Crittografare messaggi simulando un cifrario di Cesare	Scrivere un programma/algoritmo in pseudo codice che cambi ogni lettera con la sua successiva	Dei partner finanziari devono scambiarsi messaggi in codice. I messaggi possono contenere parole, spazi e punti. Scrivere un metodo che ritorni un messaggio codificato in modo che ogni lettera sia rimpiazzata dalla successiva rispetto all'alfabeto

5.2 I problemi

Domanda 5.1

Che cos'è un problema?

Risposta: un problema è un'entità sconosciuta in qualche situazione. Ovviamente è necessario che ci sia qualcuno interessato a trovare una soluzione.

Note:-

La motivazione, specialmente nei compiti di realtà, ricopre un ruolo fondamentale.

Definizione 5.2.1: Problem solving

Il *problem solving* è una sequenza di attività cognitive orientate a un obiettio unita alla manipolazione dello spazio del problema

Note:-

Un problema ben strutturato:

- presenta tutti gli elementi necessari;
- richiede l'applicazione di un certo numero di regole e princìpi organizzati in un modo predittivo e prescritto;
- presenta soluzioni conoscibili e comprensibili.

Note:-

Un problema non strutturato:

- presenta elementi sconosciuti;

- ha più soluzioni o nessuna soluzione;
- ha più criteri di valutazione;
- richiede di esprimere giudizi e opinioni personali.

5.2.1 Complessità dei problemi

Definizione 5.2.2: Complessità

La complessità di un problema è definita da:

- numero di aspetti, funzioni o variabili coinvolte;
- connettività tra queste proprietà;
- relazioni tra proprietà e stabilita nel tempo.

Note:-

Se un problema è mal strutturato si ha una maggior complessità. I problemi ben strutturati coinvolgono un insieme vincolato di variabili prevedibili e comlessità minore.

Note:-

Si ha un problema simile nel corso di "Basi di dati" in fase di progettazione.

Capitolo 6

Definizione di algoritmo

6.1 Cos'è un algoritmo?

Note:-

Nelle seguenti definizioni si fa implicitamente riferimento al solo paradigma imperativo.

Definizione 6.1.1: Algoritmo

Un *algoritmo*:

- necessità di una condizione di terminazione → termina sempre;
- deve avere chiarezza e precisione nelle istruzioni → non si devono lasciare ambiguità.

Corollario 6.1.1 Proprietà di un algoritmo

- Finitezza: termina in un numero finito di passi;
- Precisione: ogni passo è precisamente definito;
- I/O: un algoritmo ha 0+ input e ha 1+ output;
- Fattibilità: un algoritmo deve essere effettivamente eseguibile;
- Correttezza;
- Efficienza.

Note:-

In una scuola secondaria bisogna utilizzare un linguaggio più semplice rispetto ai termini accademici. Inoltre per definire un algoritmo o uno pseudo-algoritmo è utile immaginare un'interprete meccanico che deve avere istruzioni per ogni caso possibile.

Definizione 6.1.2: Problemi computazionali

Un *problema computazionale* è una collezione di domande, le istanze, per cui si sia stabilito un criterio astratto per riconoscere le risposte corrette.

Esempio 6.1.1 (Massimo comun divisore)

Ingressi:

- Coppie di interi a, b che non siano entrambi nulli.

Uscite:

- Un intero c tale che:
  - c divide sia a che b;
  - se d divide a e b allora d divide c.

Capitolo 7

Didattica della programmazione

7.1 L'apprendimento

Note:-

I seguenti risultati (teoria dei magazzini di memoria) è frutto degli studi del cognitivismo.

Definizione 7.1.1: Registro sensoriale

L'input sensoriale è ciò che viene acquisito dai 5 sensi.

Note:-

Nell'apprendimento l'input è solitamente l'udito o la vista.

Definizione 7.1.2: Memoria a breve termine (o memoria di lavoro)

Serve attenzione per passare dal registro sensoriale alla memoria di lavoro. Ha una capienza limitata in termini di spazio e tempo (circa 10 secondi, aumentabli con la ripetizione.

Note:-

Quando si sta imparando la memoria di lavoro è totalmente concentrata su un compito.

Definizione 7.1.3: Memoria a lungo termine

Ha capacità potenzialmente illimitate[a]. Non si è coscenti di queste memorie che devono essere portate, ogni volta, nella memoria di lavoro.

[a] Non esattamente

Definizione 7.1.4: Apprendimento

L'apprendimento richiede che la conoscenza avviene solo quando il concetto passa dalla memoria a breve termine alla memoria a lungo termine.

Note:-

Se si "impara" una cosa, ma il giorno dopo non si riesce più a replicarla non si ha apprendimento.

7.2 La programmazione

Definizione 7.2.1: Programmare

La *programmazione* è una rappresentazione di base fatta di schemi[a], le soluzioni e le informazioni associate.

[a] Problemi

Note:-

Ciò che distingue un esperto da un principiante è la capacità di attingere a molti più schemi memorizzati nella memoria a lungo termine.

Domanda 7.1

Com'è possibile che l'uomo riesca a svolgere attività mentali complesse con una memoria di lavoro tanto limitata?

- Si apprende meglio quando la memoria di lavoro non è troppo vuota o troppo piena;
- Il carico cognitivo è legato alla memoria di lavoro;
- Il carico cognitivo può essere:
  1. Intrinseco: imposto dal compito;
  2. Pertinente: usate per creare o modificare schemi, ma non indispensabile;
  3. Estraneo: inutile.

Note:-

- Si deve mantenere un carico pertinente elevato;
- Se il carico intrinseco è elevato non si apprende[a];
- Se il carico cognitivo è basso ci si annoia;
- Se possibile conviene ridurre il carico cognitivo di compiti difficili.

[a] Ci si concentra sul problema, ma non sugli schemi

7.3 Pattern

Esempio 7.3.1 (Un abuso di pattern)

```
Program Average
VAR count : INTEGER;
    sum, average, number : REAL;
BEGIN
  sum := -99999;
  count := -1;
  REPEAT
      writeln('please input a number');
      read (number)
      sum := sum + number;
      count := count + 1;
    UNTIL (number = 99999);
  average := sum/count;
  writeln('the average is:  ',average);
END.
```

Questo programma è stato scritto da uno studente che aveva intenzione di usare il pattern [Repeat Until] per risolvere il problema della media di una serie di input in cui il valore 99999 funge da guardia. Durante la progettazione lo studente si è trovato con il valore 99999 nella media (che ovviamente non deve essere incluso), motivo per cui ha dovuto inizializzare sum a -99999 (in questo modo i suoi valori si annullano)[a]. Bisogna far capire allo studente che ci sono costrutti più indicati per risolvere questo problema, per esempio il while. Oltre a questo un errore è il count a - 1 che causa una divisione per 0 se si inserisce 99999 all'inizio, ma questo rappresenta un problema successivo allo studio dei pattern[b].

[a] Personalmente la ritengo una soluzione creativa, ma i docenti non sono dello stesso avviso
[b] Casistica

Esempio 7.3.2 (I pattern)

Problema: Si scriva un programma che trovi il minimo in un vettore non ordinato di dimensione nota, maggiore di 0, in cui ogni valore può raggiungere al massimo 99999.

1. Input: vettore V[i], dimensione i;
2. Si inizializza min a 99999, che andrà a trovare il minimo
3. Si effettua una [scansione elementare semplice] con l'indice i che viene decrementato a ogni iterazione;
4. Durante la scansione si effettua una [Guarded-Action];
5. Se la guardia è soddisfatta si aggiorna la variabile min;
6. Se la guardia non è soddisfatta si continua finchè il ciclo non termina;
7. Si stampa il valore min.

Definizione 7.3.1: Possibile modo di definire un Pattern algoritmico

Name: il nome del pattern
Initial state: l'input del programma **Goal:** l'obiettivo del programma **Algorithm:** l'algoritmo che risolve il problema **Remarks:** eventuali osservazioni importanti sul pattern e sul suo utilizzo

Corollario 7.3.1 Tipi di pattern

⇒ **Soluzione a un problema specifico:** non costituisce un vero pattern;

⇒ **Mezza soluzione - Mezzo pattern:** in parte rappresenta un pattern, ma è ancora troppo specifico;

⇒ **Quasi pattern:** si può generalizzare, ma lo pseudocodice è specifico;

⇒ **Pattern:** è generalizzato e lo pseudocodice è generico;

⇒ **Meta-pattern:** è un pattern che descrive un pattern[a].

[a] Troppo generico.

7.4 Il ruolo delle variabili

Domanda 7.2

Come viene usata una specifica variabile in un programma? Qual è il suo ruolo?

Note:-

Si consiglia di non fare un elenco di tutti i vari tipi di utilizzo di una variabile, ma di presentarli agli studenti quando si devono effettivamente utilizzare in un programma.

Definizione 7.4.1: Variabile - Valore fissato

Una variabile assume il ruolo di valore fissato se non verrà mai modificato durante l'esecuzione del programma.

Note:-

Un esempio è il valore di π.

Definizione 7.4.2: Variabile - Contatore o indice

Una variabile assume il ruolo di contatore o indice se viene usata per scorrere una successione di valori in modo sistematico.

Note:-

Un esempio è l'indice di un vettore.

Definizione 7.4.3: Variabile - Valore più recente

Una variabile assume il ruolo di valore più recente se viene usata per memorizzare l'ultimo valore letto da un input.

Note:-

Un esempio è la variabile che memorizza l'ultimo valore letto da un input.

Definizione 7.4.4: Variabile - Valore più desiderato

Una variabile assume il ruolo di valore più desiderato se viene usata per memorizzare il "miglior" valore incontrato fino a quel momento.

Note:-

Un esempio è la variabile che memorizza il massimo di una successione.

7.4.1 Expert blind spot

Domanda 7.3

Si è certi che gli studenti siano in grado di riconoscere i ruoli delle variabili?

Definizione 7.4.5: Expert blind spot

L'expert blind spot è la difficoltà di un esperto di un argomento a comprendere le difficoltà che un principiante incontra quando si avvicina a quell'argomento.

Note:-

Ricorda l'effetto Dunning-Kruger.

7.5 Comprensione del codice

Ricapitolando, programmare vuol dire:

1. formalizzare un problema e la sua soluzione;
2. individuare un procedimento risolutivo;
3. implementare il procedimento in un linguaggio di programmazione in modo che sia eseguibile.

Definizione 7.5.1: Conoscenze

⇒ **Conoscenza sintattica:** conoscenza della sintassi del linguaggio di programmazione;

⇒ **Conoscenza concettuale:** conoscenza di come funzionano i costrutti del linguaggio di programmazione;

⇒ **Conoscenza strategica:** rappresenta la capacità di applicare le precedenti conoscenza per risolvere nuovi problemi.

Domanda 7.4

Perchè nei corsi di introduzione alla programmazione si hanno spesso episodi di scarsa performance (big failure)?

⇒ Scarse competenze di problem solving;

⇒ Scarsa comprensione dei costrutti necessari per programmare;

⇒ Scarsa capacità di applicare le conoscenze assimilate.

Note:-

Comprendere il codice e scrivere programmi sono due attività diverse.

Definizione 7.5.2: Comprensione del codice

La comprensione del codice è il processo cognitivo in cui un individuo costruisce una rappresentazione mentale di un programma.

Corollario 7.5.1 Compito di comprensione del codice

Il compito di comprensione del codice è un'attività tramite la quale si propone l'interazione con un pezzo di codice. Tramite questa interazione si costruisce una rappresentazione mentale del codice.

Proposizione 7.5.1 Strategie nella didattica della programmazione

⇒ Insegnamento esplicito della conoscenza strategica;

⇒ Analisi delle specifiche;

⇒ Analisi del comportamento del codice;

⇒ Tracciatura (simulazione del comportamento del programma);

⇒ Comprensione del codice ("spiega con parole tue");

⇒ Parsons problems (riordinare le righe di codice);

⇒ Refactoring di codice;

⇒ Adattamento di un programma a un compito simile;

⇒ Completamento di un programma;

⇒ Debugging;

⇒ Scrittura di codice.

7.5.1 Block Model

Definizione 7.5.3: Block Model

Il Block Model è un framework usato per classificare e analizzare vari aspetti della comprensione del codice.

(M) Macrostructure	Understanding the overall structure of the program text.	Understanding the *algorithm* underlying a program.	Understanding the goal/purpose of the program (in the context at hand).
(R) Relationships	Relations & references between blocks (e.g. method calls, object creation, data access...).	Sequence of method calls, *object sequence diagrams*.	Understanding how subgoals are related to goals, how function is achieved by subfunctions.
(B) Blocks (Chunks)	*Regions of Interest* (ROI) that syntactically or semantically build a unit.	Operations of a block, a method, or a ROI (chunk from a set of statements).	Understanding the function of a block, seen as a subgoal.
(A) Atoms	Language elements.	Operation of a statement.	Function of a statement: its purpose can only be understood in a context.
	(T) Text Surface	(P) Program Execution	(F) Function/Purpose
Duality	Architecture/Structure Dimensions		Relevance/Intention Dimension

Figure 7.1: Block Model

Esempio 7.5.1 (Block Model)

Esercizio	*Comprensione*	*Strategia*
Indicare la struttura a blocchi di un programma	T	M
Determinare la presenza di codice ridondante	P	M
Riassumere un programma in una breve frase	F	M
Identificare ogni assegnamento nel codice	T	A
Determinare il valore di una variabile dopo l'esecuzione di un programma	P	A
Spiega lo scopo di un elemento di un programma	F	A
Identifica lo scopo di una variabile	F	R
Identifica i blocchi di un programma	T	B
Completa il codice e un diagramma di flusso	P	B
Rifletti su un codice	T	R
Parson's problem	P	R
Spiegare lo scopo di un programma	F	B

7.5.2 Errori degli studenti

Definizione 7.5.4: I distrattori

I distrattori sono le risposte sbagliate che gli studenti danno più frequentemente nei test a risposta multipla. Esse sono impostate in modo da corrispondere a specifici errori concettuali.

Note:-

In questi casi l'insegnate può facilmente individuare cosa lo studente non abbia capito[a].

[a] Assumendo che lo studente non abbia copiato o tirato a caso.

Corollario 7.5.2 5 livelli di risposte

⇒ **Prestructural:** la risposta dell'alunno è la meno sofisticata tra quelle possibili[a];

⇒ **Unistructural:** la risposta dell'alunno mostra una comprensione parziale del problema, una sorta di "congettura istruita";

⇒ **Multistructural:** la risposta dell'alunno rivela una consapevolezza di tutte le parti del problema, ma non riesce a collegarle tra loro[b];

⇒ **Relational:** la risposta dell'alunno rappresenta un'integrazione delle parti del problema;

⇒ **Extended abstract:** la risposta dell'alunno va al di là del problema immediato ed è collegata a un contesto più ampio;

[a] È sintomo di un'idea sbagliata o di un preconcetto irrilevante.
[b] "Not seeing the forest for the trees".

Definizione 7.5.5: Scaffolding

Lo scaffolding è un processo di supporto che aiuta gli studenti:

- Ridurre il carico cognitivo;
- Fornire un modello di soluzione adeguato;
- Favorire lo sviluppo di una conoscenza strategica.

Note:-

Ovviamente lo scaffolding deve essere piano piano rimosso.

7.6 Approcci innovativi

7.6.1 PRIMM

Definizione 7.6.1: PRIMM

PRIMM è un framework per la progettazione di attività didattiche per l'insegnamento della programmazione comprendente le seguenti fasi:

- **Predizione:** gli studenti devono prevedere il comportamento di un programma;
- **Esecuzione:** gli studenti devono eseguire il programma e verificare la predizione;
- **Investigazione:** gli studenti devono correggere la predizione in caso di errore;
- **Modifica:** gli studenti devono modificare il programma in modo che si comporti in un modo diverso;
- **Risoluzione:** gli studenti devono risolvere un problema usando il programma.

7.6.2 POGIL

Definizione 7.6.2: POGIL

POGIL è un framework per la progettazione di attività didattiche per l'insegnamento della programmazione. Durante queste attività gli studenti attraversano un ciclo di esplorazione, concettualizzazione e applicazione. Gli studenti scoprono i concetti chiave e costruiscono la propria conoscenza attraverso l'interazione con i compagni e con il docente.

7.6.3 NDL

Definizione 7.6.3: NDL

NDL (Necessity design learning) è un framework per la progettazione di attività didattiche per l'insegnamento della programmazione. Sostanzialmente si dà un problema risolvibile con un certo costrutto senza introdurlo. Successivamente, prima che lo studente si scoraggi, si introduce il costrutto.

Capitolo 8

La natura dei programmi

⇒ I programmi sono dappertutto;

⇒ La programmazione è sempre più presesente nell'ambito scolastico (coding).

8.1 Che cos'è un programma?

Questa domanda è molto complicata, la cui risposta cambia a seconda della persona a cui la si pone:

- se una persona utilizza principalmente applicativi può vedere i programmi come strumenti per lavorare, divertirsi, ecc. . . ;
- se una persona è un programmatore può vedere i programmi come una sequenza di istruzioni legata agli algoritmi.

> **Obiettivo 8.1.1**
> Fornire agli insegnanti una visione ampia di cosa sia un programma, al di là di definizioni riduttive o stereotipate.

8.1.1 Un quadro di riferimento sulla natura dei programmi

Questo *framework* può essere utile per:

- capire la *centralità dei programmi* ai giorni nostri;
- orientare le *scelte didattiche*.

Il termine *programma*, in senso lato, è usato:

- dai programmi che si scrivono a scuola ai sistemi operativi.

Si considerano solo programmi "*tradizionali*":

- che implementano *algoritmi*;
- per cui si può *spiegare il risultato* ottenuto;
- esclusi i programmi prodotti da macchine.

8.2 Le sfaccettature dei programmi

I programmi come *strumenti*:

- utili nel lavoro, nel tempo libero, ecc. . . ;
- vengono visti come scontati.

I programmi come *opere dell'uomo*:

- sono *solitamente* prodotti con uno scopo preciso;
- sono il risultato di scelte.

I programmi come *oggetti fisici*:

- risiedono su un mezzo fisico;
- la loro esecuzione richiede tempo ed energia;
- hanno bisogno di un dispositivo fisico per essere eseguiti.

I programmi come *entità astratte*:

- manipolano nozioni e concetti astratti;
- elaborano simboli;
- gli effetti fisici sono in funzione del risultato astratto;
- gli algoritmi sono astratti.

I programmi come *entità eseguibili*:

- sono eseguiti da un calcolatore;
- possono essere ripetuti;
- l'esecutore è un agente che elabora segnali.

I programmi come *manufatti linguistico-notazionali*:

- rispettano una specifica sintassi;
- sono un modo per esprimere idee;
- sono un modo per comunicare idee;
- possono essere tradotti in altre notazioni.

Note:-

I programmi sono scritti da persone per essere compresi da altre persone e per essere eseguiti da una macchina che non li comprende.

Capitolo 9

Cenni sulla valutazione dell'apprendimento

Una rubrica *valutativa* non serve solamente per assegnare un voto a uno studente, ma anche per *aiutarlo a capire* quali sono i suoi punti di *forza* e di *debolezza* per poter migliorare.

Note:-

Lo studente deve essere parte attivà del processo di valutazione.

9.1 Conoscenze, abilità e competenze

Definizione 9.1.1: Conoscenze

Le *conoscenze* indicano i risultati delle informazioni assimilate.

Note:-

Le conoscenze sono descritte come teoriche e/o pratiche.

Definizione 9.1.2: Abilità

Le *abilità* indicano la capacità di applicare le conoscenze.

Note:-

Le abilità sono descritte come cognitive (uso del pensiero logico) e pratiche (che richiedono l'uso di materiali).

Definizione 9.1.3: Competenze

Le *competenze* indicano la capacità di applicare le conoscenze e le abilità in un contesto specifico.

Note:-

Le competenze sono descritte in termini di responsabilità e autonomia.

Capitolo 10

Letture

10.1 Manifesto di vienna per l'umanesimo digitale

Il sistema sta fallendo. Tim Berners-Lee[1] sostiene che la digitalizzazione porti con sè vari problemi: perdità di *privacy*, insorgere di *comportamenti estremisti*, formazione di *bolle informative*, etc. Per sostenere l'*innovazione tecnologica* c'è bisogno di un vasto *impegno sociale*.

Il manifesto. Si rivolge alle comunità *accademiche* e *professionali*, ai leader *industriali* e *politici*.

Tecnologia e società. I recenti sviluppi tecnologici *creano* e *distruggono* posti di lavoro e ricchezza. Viene modificata la gerarchia tra *uomo* e *macchina*.

L'approccio illuminista e umanista. Il manifesto rappresenta un'estensione della tradizione intellettuale dell'*umanesimo* che ambisce a creare un'umanità *illuminata*.

Le tecnologie digitali. Esse nascono da scelte *implicite* ed *esplicite* che promuovono valori, norme, interessi economici, etc.

Umanesimo digitale. Si deve spostare il *focus* dalle tecnologie all'uomo.

I *princìpi fondamentali* sono:

- Le tecnologie digitali dovrebbero essere progettate per promuovere
  la democrazia e l'inclusione;
- La privacy e la libertà di parola sono valori essenziali per la democrazia e dovrebbero essere al centro delle nostre attività;
- Devono essere stabilite norme, regole e leggi efficaci, basate sul dibattito pubblico e su un ampio consenso;
- I regolatori devono intervenire sui monopòli tecnologici;
- Decisioni le cui conseguenze possono influire sui diritti umani individuali o collettivi devono continuare a essere prese dalle persone;
- Approcci scientifici interdisciplinari sono un prerequisito per affrontare le sfide future;
- Le università sono il luogo in cui si producono nuove conoscenze e si coltiva il pensiero critico;
- I ricercatori accademici e industriali devono aprirsi al dialogo con la società e valutare criticamente i propri approcci;

[1] Il fondatore del web

- I professionisti di tutto il mondo dovrebbero riconoscere la loro corresponsabilità nell'impatto sociale delle tecnologie digitali;
- È necessaria una visione che consideri nuovi programmi di studio che combinino la conoscenza delle scienze umane e sociali e degli studi scientifico-ingegneristici;
- L'educazione all'informatica e al suo impatto sociale devono iniziare il prima possibile.

10.2 Informatica: la terza rivoluzione "dei rapporti di potere"

L'*informatica* è una vera e propria rivoluzione per la razza umana, la terza dopo quella della *stampa* e quella *industriale*.

L'invenzione della stampa nel XV secolo è stata una rivoluzione sia di tipo *tecnico*[2] che di tipo *sociale*[3]. Il potere della *conoscenza* non è più confinato alle persone che lo posseggono, perchè ogni testo può essere *replicato*. La diffusione di testi scientifici, giuridici e letterari hanno portato la società a essere più *democratica*.

Ed è proprio la conoscenza scientifica[4] a portare alla rivoluzione industriale. A partire dal '700 la disponibilità di *macchine* rende possibile l'*automatizzazione* del lavoro fisico delle persone. Questa rivoluzione ha carattere *tecnico* perchè permette di ricreare velocemente i manufatti. Inoltre le macchine non si stancano, quindi possono lavorare giorno e notte. Questo mette in discussione il potere della *natura*: l'uomo la assoggetta e ne superà i limiti.

Nella seconda metà del '900 inizia la terza grande rivoluzione: l'informatica. Essa non è più una replica della conoscenza statica dei libri o della forza fisica delle persone, ma è "*conoscenza in azione*". Il sapere non è una rappresentazione *statica* dei fatti, ma uno scambio di dati *interattivo* tra soggetto e realtà. Il potere che viene minato è l'*intelligenza umana*: essa può essere in qualche modo replicata dai programmi. Basti pensare agli scacchi in cui il computer è in grado di battere un campione del mondo. Oppure ai recenti sviluppi dell'*intelligenza artificiale*. Tuttavia ci sono due cose che le "macchine cognitive" non sono ancora riuscite a emulare: la *flessibilità* e l'*adattabilità*.

In conclusione: è inevitabile il diffondersi di queste nuove tecnologie e dei cambiamenti a loro collegati, ma è importante che ogni individuo sia istruito e formato sulle basi concettuali che permettono di costruire queste macchine.

[2] Si producono testi più velocemente e più economicamente
[3] Favorisce una maggiore diffusione della conoscenza
[4] In primis il metodo Galileiano

10.3 Informatica e competenze digitali: cosa insegnare?

Risposta breve: entrambe.

Risposta lunga: questo intero articolo.

Nel mondo scolastico c'è confusione tra i termini "*informatiche*" e "*digitali*". Purtroppo negli ultimi 30 anni si sono usati in modo intercambiabile questi due termini per riferirsi a: programmare in Pascal, usare Word, scrivere e-mail, coding, usare i social, etc.

Digitale si riferisce alla rappresentazione di un dato con un simbolo numerico. Informatico si riferisce alla capacità di *elaborazione automatica* dei dati resa possibile dai metodi e dalle teorie dell'informatica. Usare dei simboli numerici non è una novità, ma lo è elaborare le informazioni in modo automatico, come se si usasse un gigantesco *orologio*.

L'orologio è solo un esecutore meccanico che non sa che cosa rappresenta o come sia costruito. Il computer quindi può manipolare simboli e istruzione che per lui non hanno significato, ma che lo hanno per l'uomo. Questo è ala base della terza rivoluzione "dei rapporti di potere".

C'è ancora confusione anche sui documenti ufficiali dell'unione europea in cui, per esempio, la programmazione rientrà tra le competenze digitali, sebbene non lo sia. Negli USA o in UK è una competenza informatica. In questi paesi, l'attuale società è semplicemente un'estensione della società *industriale*. Infatti la società *digitale* utilizza delle macchine[5]. Per trasmettere la comprensione della società digitale è necessario portare nelle scuole l'informatica. Ciò non nega l'insegnamento delle competenze digitali che occupano un ruolo chiave nella condivisione della conoscenza.

[5]Di tipo cognitivo, non meccanico

Capitolo 11

Domande e risposte

11.1 La natura dell'informatica

Domanda 11.1

Quali sono i problemi dell'informatica?

Risposta: il termine "informatica" è spesso utilizzato in modo improprio nel linguaggio comune. per esempio, viene usato per riferirsi a computers, cellulari, televisori, etc. Ma l'informatica non rigurda i computers che sono solo uno strumento, non il fine. Un altro problema è il termine "digitale" che, solitamente, viene usato come sinonimo di "informatica". In realtà, digitale si riferisce alla rappresentazione di un dato mediante un simbolo numerico e a tutte le tecnologie basate sui computers.

Domanda 11.2

Quali sono i problemi relativi all'insegnamento dell'informatica nel nostro sistema scolastico?

Risposta: l'informatica viene spesso trattata come una non-materia, per cui chiunque sappia usare un banale applicativo come Word o Excel tende a comportarsi come un esperto (si è vista in precedenza la distinzione tra informatico e digitale). Inoltre un problema endemico della scuola italiana riguarda la carenza di informatici che insegnino la materia, infatti spesso si ricorre a docenti di matematica (la cui classe di concorso li rende "idonei" a insegnare informatica).

Domanda 11.3

Perchè è importante insegnare informatica fin dalla scuola primaria?

Risposta: Come le scienze naturali (di cui fa parte) l'informatica offre una chiave di lettura del mondo che ci circonda, per cui è opportuno iniziare a studiarla il prima possibile e che ogni studente ne abbia una conoscenza di base. Inoltre, l'informatica è una materia che si presta molto bene a essere insegnata in modo interdisciplinare.

Domanda 11.4

Perchè è importante riflettere sulla natura dell'informatica prima di insegnarla?

Risposta: perchè ciò andrà a impattare sul modo di insegnare la materia. A seconda della propria visione del mondo si privilegeranno alcuni aspetti rispetto ad altri. Inoltre bisogna sempre chiedersi il perchè si studia informatica: essa dà una chiave di lettura del mondo digitale che è sempre più presente nella nostra quotidianità.

Domanda 11.5

Quali sono le 3 anime/paradigmi che abbiamo discusso per inquadrare la natura dell'informatica?

Risposta:

- matematico: l'informatica vista come una scienza matematica che formalizza i problemi e li risolve mediante algoritmi;
- ingegneristico: l'informatica vista come un'ingegneria che si occupa di progettare e realizzare sistemi software;
- scientifico: l'informatica vista come una scienza che studia i sistemi software e i processi di sviluppo empiricamente.

11.2 Teorie dell'apprendimento

Domanda 11.6

Che cos'è il comportamentismo?

Risposta: il comportamentismo è una teoria dell'apprendimento che si basa sull'osservazione del comportamento. Essa prevede di modellare un comportamento desiderabile. La valutazione si basa sui cambiamenti nei comportamenti degli alunni visti come "tabule rase". Spesso erano previsti "rinforzi" ossia punizioni corporali. Questo approccio è istruttivista.

Domanda 11.7

Che cos'è il cognitivismo?

Risposta: il cognitivismo rappresenta un superamento del comportamentismo. Il cognitivismo è una teoria dell'apprendimento che si basa su alcune idee principali:

- il carico cognitivo: il carico di lavoro mentale che un individuo deve sostenere per svolgere un compito;
- gli schemi e i modelli mentali.

L'approccio cognitivista punta a far ricordare e applicare la conoscenza.

Domanda 11.8

Che cos'è il costruttivismo? (socio-costruttivismo e costruttivismo cognitivo)

Risposta: il costruttivismo (ideato da J. Piaget) è una teoria dell'apprendimento che si basa sullo scetticismo:

- la conoscenza è frutto dell'esperienza;
- non c'è modo di sapere la "vera" verità.

Si ricorre alla "viabilità" per valutare la conoscenza: un'idea è valida se ha funzionato fino a quel momento. Per le azioni fisiche è viabile tutto ciò che porta a un risultato. Sul piano concettuale ci si basa sulla non contradditorietà.

⇒ Socio-costruttivismo: l'apprendimento è un processo sociale che avviene in un contesto sociale. Si crea insieme nuova conoscenza;

⇒ Costruttivismo cognitivo: l'apprendimento è il processo di costruzione del significato. L'insegnante deve solo facilitare la scoperta offrendo le risorse necessarie.

Domanda 11.9

Che cos'è il costruzionismo?

Risposta: il costruzionismo (ideato da S. Papert) si basa sull'idea costruttivista di conoscenza. Ma a ciò viene aggiunta l'idea che la conoscenza deve essere finalizzata alla costruzione di artefatti.

Domanda 11.10

Quali sono i punti fondamentali del brano di Papert "Gli ingranaggi della mia infanzia"?

Risposta:

- è più facile apprendere qualcosa se lo si assimila a modelli già posseduti (per Papert gli ingranaggi e il differenziale);
- al contrario l'apprendimento è più ostico se non si ha un modello di riferimento;
- oltre agli aspetti cognitivi, l'apprendimento è influenzato da aspetti emotivi;
- gli ingranaggi sono un qualcosa di personale per Papert, non funzionerebbero come modello per tutti.

Domanda 11.11

Cos'è l'assimilazione?

Risposta: l'assimilazione è l'incorporazione di un determinato concetto in uno schema che è stato già acquisito.

Domanda 11.12

Cos'è l'accomodamento?

Risposta: l'accomodamento è la modifica di una struttura cognitiva in relazione al contatto con nuove informazioni.

Domanda 11.13

Cos'è la zona di sviluppo prossimo (ZSP)?

Risposta: la zona di sviluppo prossimo è la seconda delle tre aree di apprendimento di un bambino. Nella ZSP il bambino è in grado di apprendere solo con il supporto del docente ed è in quest'area che l'insegnante può intervenire.

Domanda 11.14

Quali sono le caratteristiche principali dell'apprendimento attivo?

Risposta: l'apprendimento attivo nasce dalla didattica costruttivista.

- l'apprendimento non avvene attraverso fasi standard;
- ogni studente ha la possibilità di stabilire il proprio percorso;
- l'insegnante è un facilitatore;
- le parole e le azioni del docente sono strumenti per apprendere;
- si dà la priorità all'esperienza diretta (gestita dall'insegnente).

11.3 Problemi, compiti, spazio di rappresentazione del problema

Domanda 11.15

Descrvere la differenza tra compito, problema ben/mal strutturato, formulazione narrativa e algoritmica di un problema.

Risposta:

- compito: è un'entità sconosciuta in qualche situazione, per cui è necessario trovare una soluzione. Può essere presentato sotto forma di "compito di realtà", ossia basato su eventi verosimili;
- problema ben strutturato: è un problema che presenta tutti gli elementi necessari alla risoluzione, richiede l'applicazione di regole e princìpi organizzati in modo predittivo e prescrittivo, ha una soluzione conoscibile e comprensibile;
- problema mal strutturato: è un problema che presenta elementi sconosciuti, ha più soluzioni o nessuna soluzione, ha più criteri di valutazione, richiede di esprimere giudizi e valutazioni;
- formulazione narrativa: è una descrizione del problema in linguaggio naturale che racconta qualcosa;
- formulazione algoritmica: è una descrizione del problema in linguaggio formale riferendosi direttamente alle strutture dati e alle variabili richieste.

Domanda 11.16

Spiegare il ruolo e l'importanza della rappresentazione e manipolazione dello spazio del problema come parte delle strategie di problem solving.

Risposta: È importante che i problemi siano simulazioni di problemi quotidiani e professionali, in cui l'insegnante abbia scelto che componenti includere e come rappresentarle. Se il problema è ben strutturato, la rappresentazione dello spazio è semplice per via del modo prevedibile in cui si comportano le variabili, quindi è, generalmente, più semplice. Viceversa con un problema mal strutturato la rappresentazione dello spazio è più complessa e richiede più tempo.

11.4 Struttura dell'esame

1. **Discussione dell'attività didattica (40% del voto):** i membri del gruppo presentano il lavoro svolto e rispondono a eventuali domande;
2. **Discussione delle consegne (30% del voto):** si devono discutere alcune delle consegne svolte. In questa parte si può guardare ciò che si è scritto;
3. **Domande (30% del voto):** vengono poste domande sul corso[1].

Note:-

Se si svolge in gruppi ricordare:

- di parlare in maniera omogenea, si raccomanda di prepararsi ognuno una propria parte e di non improvvisare;
- che ogni membro deve comunque sapere tutto dell'attività;
- che la discussione sulle consegne è individuale.

[1] Presenti in questa sezione.