Tradizione e novità nella didattica dell'informatica

L'Istituto è quello che noi siamo (docenti, studenti…). L'idea che l'insegnante "segue" i programmi ministeriali è fortunatamente tramontata con l'autonomia del 2000. Insieme ragioniamo sulle prospettive del nostro indirizzo di studi. L'evoluzione dell'informatica è, in sintesi, un processo di astrazione sia dal punto di vista del sw che dell'hw. Il processo di astrazione è stato inizialmente lento e poi dal 1946 sempre più rapido.

Linguaggi di programmazione a basso livello (coincide con il linguaggio macchina oppure il livello di Astrazione dal linguaggio macchina è basso) Linguaggi di programmazione a medio e alto livello (il livello di Astrazione dal linguaggio macchina è crescente) Linguaggi di interfaccia a basso livello (schde perforate, a linea di comando) Linguaggi di interfaccia ad alto livello (dalla Gui in poi)

Il nostro sforzo formativo è finalizzato:

1. a dare spazio alla creatività
2. a migliorare le capacità logiche, di astrazione, di formalizzazione, di analisi e sintesi di situazioni problematiche (pensiero computazionale);
3. a far utilizzare le tecnologie informatiche nelle varie situazioni anche in modo operativo piuttosto che riflessivo (assemblaggio pc, installazione sw di base, il webmaster 21, cablaggio reti e cavi, riparazione errori sw di base, mantenimento hard disk, ecc.);
4. a far utilizzare le tecnologie informatiche nei vari ambiti operativi (amministratore di reti, amministratore di database, ecc.) e nelle reti informatiche collegate a dispositivi periferici (stampanti 3D, robot, droni, arduino, sensori, attuatori ecc.);

5. far documentare in modo corretto, completo e secondo modalità previste in ingegneria del software.

Quello educativo:

1. Prove oggettive (docimologia) e analisi sommative dei dati;
2. Consapevolezza dell'apprendimento e analisi delle proprie prestazioni.

Indirizzo di studi "Informatica e telecomunicazioni"

BIENNIO

CLASSE PRIMA Materia **"Tecnologie informatiche"**

Tradizionale

- Le architetture del computer;
- Hardware: CPU,RAM, ROM, PROM, EPROM, FLASH-EPROM, CACHE, BUS, Memorie di Massa fisse e rimovibili (capacità e tempo di accesso, tempo di posizionamento e di latenza, cilindro, settore, traccia, cache da disco, SSD, DVD, BLU-RAY), Periferiche, Tipi di computer;
- Software: Il sistema operativo, la struttura a moduli di astrazione crescente, impostazioni (sicurezza, account utente, aspetto, reti ed internet, hardware e suoni, accessibilità, area geografica, programmi), la gestione dell'archiviazione, interfaccia GUI e a linea di comandi;
- Legislazione e informatica(ergonomia, virus e protezione dati, licenze d'uso, il copyright, dati personali e dati sensibili, privacy, crimini informatici);
- Scrivere testi (Word processor): dalla macchina da scrivere al word processor, concetto di documento, paragrafo, carattere, stampa unione, correzione automatica, formattazione (documento, testo, paragrafo, carattere), disegno e immagini, oggetto, elenchi, bordi, sfondi;
- Automatizzare la risoluzione di problemi (Foglio elettronico): foglio di lavoro e celle, impostazione del problema nelle celle e nei fogli (dal problema all'algoritmo, dati iniziali, soluzioni come percorso nei dati intermedi, risultato come dato finale, esecutore), Formule e funzioni, riferimenti e grafici. Risoluzione di problemi matematici, statistici e di gestione di informazioni, VBA (algoritmo, programma, selezione);
- Preparare presentazioni: le diapositive e lo schema delle diapositive, animazioni, audio;
- Ipertesti, multimedialità e storia della comunicazione, il WWW, html;
- La digitalizzazione del suono, acquisizione di suoni(scheda e periferiche), i formati dei file audio, la compressione dei file audio;
- Le immagini digitali, pixel, risoluzione, visualizzazione, colore (il modello RGB), profondità di colore, compressione, formati (bitmap o a punti o raster, vettoriale o orientata agli oggetti);
- Il video digitale, i formati, il codec, i player, acquisizione, formati, conversione, compressione;

- Il sistema di numerazione multibase;
- Le unità di misura: frequenza, memoria, tempo di trasferimento dati; bitrate file audio, risoluzione immagini;
- Telematica e reti di computer, gli elementi della comunicazione (soggetti, messaggio, mezzo trasmissivo, linguaggio e regole), tipi di reti, canali di comunicazione, tecniche di commutazione di canale, reti analogiche e digitali, banda larga, tipologie di reti, creazione di una semplice LAN, i servizi di internet (ftp, www), browser ;
- Sicurezza in Rete, i cookie, i firewall, antivirus, antispyware, crittografia, firma digitale;
- La logica con il foglio elettronico (tavole di verità di operatori logici, proposizioni composte, proprietà associative), la geometria con il foglio elettronico (retta, fascio), la matematica con il foglio elettronico (progressioni, equazioni e disequazioni), la statistica con il foglio elettronico;
- La programmazione con LOGO: movimenti della tartaruga, la geometria con la tartaruga, selezione ed iterazione definita, le procedure e i parametri;
- Programmare Arduino con Scratch: selezione ed iterazione;
- Mappe mentali e concettuali.
- Simulatore di circuiti digitali e porte logiche***;***
- Teoria del colore, app design (https://it.yeeply.com/blog/app-design-e-limportanza-del-colore/);
- - CAD

Novità

- Fogli elettronici: Vba;
- Presentazioni: Sway;
- Scratch e logo;
- Simulatore di circuiti digitali e porte logiche (ad. es. digitalWorks);
- Progettazione e documentazione CAD.

CLASSE SECONDA Materia **"Scienze e tecnologie Applicate"**

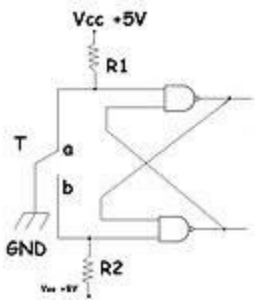
Tradizionale

- Sistemi e Modelli
- Automi (diagramma degli stati, tabelle di transizione, automi riconoscitori, automi di Mealy, di Moore e senza uscite)

- Trasmettitore e ricevitore, il sottosistema canale, i mezzi trasmissivi, il rumore di un canale, multiplazione, le reti di comunicazione, la commutazione, la trasmissione si segnali digitali, campionamento e quantizzazione di un segnale analogico
- Aritmetica del computer (sistemi addizionali e posizionali, sistema decimale, binario, ottale ed esadecimale, conversioni)
- Rappresentazione dell'informazione numerica e alfanumerica, Rappresentazione dell'informazione multimediale (la codifica delle immagini, i sistemi di compressione, grafica raster e grafica vettoriale, immagini bitmap, il formato jpeg e la geometria frattale, la grafica tridimensionale, l'audio digitale, il video digitale)
- Algoritmi e flow-chart (formulazione, astrazione e analisi di problemi, scissione di un problema, applicazioni matematiche e logiche)
- Mappe mentali
- Javascript e le pagine web (HTML)

Novità

- Arduino
- Scratch e logo
- Algebra di Boole e i suoi modelli: Algebra delle Proposizioni e Algebra dei Circuiti
- Simulatore di circuiti digitali e porte logiche
- Mappe mentali



TRIENNIO

CLASSE TERZA INFORMATICA E TELECOMUNICAZIONI articolazione Informatica Materia **"Informatica"**

- ***Tradizionale***
- La *comunicazione* (tema interdisciplinare e pluridisciplinare delle discipline umanistiche, scientifiche e tecniche)
- Origini matematiche, tecnologiche dell'informatica e dei linguaggi di programmazione
- Astrazione ed evoluzione dell'informatica
- La modellizzazione de problemi e la strategia risolutiva
- Descrizioni rigorose , l'algoritmo e rappresentazioni dell'algoritmo
- Confronto tra i vari paradigmi di programmazione

Obiettivi standard

(ciascun obiettivo deve essere descritto con un verbo che indica un'azione "misurabile" (ad es. "deve scegliere o deve riconoscere")**, dalle condizioni** (ad es. "tra 5 sequenze possibili**"), dal tempo per eseguire l'azione indicata dal verbo** (ad. es. in due minuti"**) :**

Conoscenze:

1. L'alunno riconosce la definizione corretta di "problema" tra 5 possibili risposte (4 distrattori) in 2 minuti;

2. L'alunno riconosce una sequenza top-down o bottom-up tra 5 sequenze(4 distrattori) in 2 minuti;

3. L'alunno indica uno dei 5 motivi per utilizzare i sottoprogrammi tra 5 possibili risposte (4 distrattori) in 2 minuti;

4. L'alunno indica la differenza tra sottoprogramma e sottoalgoritmo tra 5 possibili risposte (4 distrattori) in 2 minuti;

5. L'alunno indica la differenza tra procedura e funzione tra 5 possibili risposte (4 distrattori) in 2 minuti;

6. L'alunno indica le variabili locali e quelle globali in un codice di max 30 righe in 2 minuti;

7. L'alunno indica i parametri formali ed attuali in un codice di max 30 righe in 2 minuti;

8. L'alunno indica il risultato corretto di un algoritmo espresso in pseudolinguaggio di max 30 righe contenente due funzioni che scambiano i parametri per valore e per indirizzo in 5 minuti;

9. L'alunno riconosce una categoria del software tra 5 possibili risposte (4 distrattori) in 2 minuti;

10. L'alunno riconosce una funzione del sistema operativo tra 5 possibili risposte (4 distrattori) in 2 minuti;

11. L'alunno riconosce la sequenza di bootstrap tra 5 possibili risposte (4 distrattori) in 2 minuti;

12. L'alunno riconosce il paradigma di programmazione tra 5 possibili risposte (4 distrattori) in 2 minuti;

13. L'alunno riconosce la sequenza di fasi del compilatore o di un interprete tra 5 possibili risposte (4 distrattori) in 2 minuti;

14. L'alunno riconosce una funzione del sistema operativo tra 5 possibili risposte (4 distrattori) in 2 minuti;

15. L'alunno riconosce gli aspetti della sicurezza di un sistema informatico in una lista di 10 possibili risposte in 2 minuti;

16. L'alunno indica la definizione corretta di cookie tra 5 possibili risposte (4 distrattori) in 2 minuti;

17. L'alunno riconosce una tecnica di sniffing o di spoofing o di spamming o di backdoor o di nuking tra 5 possibili risposte (4 distrattori) in 2 minuti;

18. L'alunno riconosce un sistema a crittografia simmetrica o asimetrica tra 5 possibili risposte (4 distrattori) in 2 minuti;

19. L'alunno riconosce una frase inerente il decreto legge 196/2003 o Dlgs 82 7 marzo 2005 o art. 15 legge 59 15 marzo 1997 o art. 615 ter legge 547/93 tra 5 frasi (4 distrattori) in 2 minuti;

20. L'alunno riconosce la strategia risolutiva di un algoritmo per le operazioni sui vettori: "shift degli elementi", "rotazione di un vettore", "vettori paralleli", "ordinamento a cicli fissi", "ordinamento a bolle", "ricerca sequenziale", "ricerca binaria".

Abilità:

1. L'alunno disegna il flow-chart della soluzione di un problema risolvibile mediante una o due istruzioni di selezione a una via o due vie in 5 minuti;

2. L'alunno disegna il flow-chart della soluzione di un problema risolvibile mediante una istruzione di iterazione definita in 5 minuti;

3. L'alunno disegna il flow-chart della soluzione di un problema risolvibile mediante una istruzione di iterazione indefinita in 5 minuti;

4. L'alunno codifica il flow-chart della soluzione di un problema risolvibile mediante una o due istruzioni di selezione a una o due vie in 5 minuti;

5. L'alunno codifica il flow-chart della soluzione di un problema risolvibile mediante una istruzione di iterazione definita in 5 minuti;

6. L'alunno codifica il flow-chart della soluzione di un problema risolvibile mediante una istruzione di iterazione indefinita in 5 minuti;

7. L'alunno codifica una procedura o funzione con parametri e la richiama nel main in 5 minuti;

8. L'alunno codifica una procedura o funzione con parametri e variabili locali e la richiama nel main in 10 minuti;

9. L'alunno codifica il caricamento di un vettore e una delle seguenti tecniche fondamentali: somma degli elementi, contatore di un elemento, calcolo del max/min, ordinamento a cicli fissi, ricerca lineare di un elemento in 30 minuti;

10. L'alunno codifica una delle seguenti operazioni sui vettori: "shift degli elementi" , "rotazione di un vettore", "caricamento vettori paralleli", "ordinamento a bolle", "ricerca binaria";

11. L'alunno codifica il caricamento di una matrice bidimensionale e una delle seguenti tecniche fondamentali: somma degli elementi, contatore di un elemento, calcolo del max/min, somma degli elementi di una riga o di una colonna, diagonale principale e secondaria di una matrice quadrata in 30 minuti;

12. L'alunno codifica il caricamento e la lettura di un vettore di records in 20 minuti;

13. L'alunno codifica il calcolo di una operazione relativa ad un campo di un array di records in 30 minuti ;

14. L'alunno codifica una procedura o funzione con parametri ricorsiva e la richiama nel main (Fibonacci, potenza) in 20 minuti;

15. L'alunno calcola il costo di un costrutto iterativo in 5 minuti;

Competenze:

1. L'alunno valuta la complessità computazionale tra 2 oppure 3 algoritmi di max 10 righe in 20 minuti; 2. L'alunno scrive in linguaggio naturale la strategia risolutiva di una operazione fondamentale sugli array e array di records; 3. L'alunno codifica la strategia risolutiva di una operazione fondamentale sugli array di records.

Novità

- Python (https://www.programmareinpython.it/)
- C#
- App Android (utilizzando Visual Studio C# e/o Eclipse Java)
- Prolog e Logica (http://www.ce.unipr.it/research/HYPERPROLOG/manuale.html)
- Mappe mentali
- Arduino (http://pensierocomputazionale.altervista.org/)
- Scratch e logo
- Html e javascript
- Linguaggi e grammatiche
- Teoria del colore, app design (https://it.yeeply.com/blog/app-design-e-limportanza-del-colore/)

INFORMATICA E MATEMATICA CLASSE III Introduzione di alcune definizioni dell'**algebra lineare**, che è la branca della matematica che si occupa dello studio delle principali regole per la manipolazione di vettori e matrici.

Definizioni

In matematica una **matrice** è un insieme di numeri disposti su righe e colonne, ad esempio:

$$A = \begin{pmatrix} 2 & 3 & 1 \\ 4 & -7 & 5 \end{pmatrix}$$

A è una matrice formata da 2 righe e 3 colonne.

La **dimensione** di una matrice si esprime con: *n. di righe x n. di colonne*. La matrice precedente si dice essere una matrice *2 x 3* o una matrice di tipo *(2, 3)*. L'**elemento di una matrice** appartenente alla riga *i-esima* e alla colonna *j-esima* si indica con:

$$a_{ij}$$

Nel caso della matrice precedente: a11=2, a12=3, a13=1, a21=4, a22=-7, a23=5.

Due matrici si dicono **uguali** se hanno ordinatamente uguali tutti i loro elementi.

I **vettori** sono delle matrici particolari formate da una sola riga o da una sola colonna:

$$V1 = \begin{pmatrix} 4 & 2 & 5 \end{pmatrix} \ , \ V2 = \begin{pmatrix} 6 \\ 10 \\ 2 \end{pmatrix}$$

V1 è un **vettore riga** di dimensione 1 x 3, V2 è un **vettore colonna** di dimensione 3 x 1.

Alcune **matrici particolari** sono:

- la *matrice nulla*: composta da elementi tutti nulli;
- la *matrice quadrata*: quando il n. di righe coincide con quello delle colonne;
- la *matrice diagonale*: è una matrice quadrata avente tutti gli elementi uguali a zero, eccetto quelli sulla diagonale principale

aij=0 per i diverso da j

Sistemi e Tecnologie

Tradizionale

- Assembler, sistemi addizionali e sistemi posizionali, sistemi di numerazione decimale, binario, ottale ed esadecimale e relative conversioni, aritmetica binaria, rappresentazine delle informazioni; il codice, rappresentazione alfanumerica, dei numeri interi e reali

- La codifica delle immagini e dei suoni, la compressione
- Sistemi Operativi (Gestione memoria centrale, Gestione memoria permanente, Programmazione concorrente)
- I sistemi, i sistemi di controllo a catena aperta e a catena chiusa, classificazione dei sistemi
- Rappresentazione dei sistemi: i modelli, classificazione dei modelli,
- Automi e rappresentazione di automi con diagrammi degli stati e con tabelle di transizione, automi riconoscitori
- Automi di Mealy, di Moore e senza uscite
- Trasmettitore e ricevitore, il sottoinsieme canale, segnali analogici e digitali, i mezzi trasmissivi, il rumore di un canale, multiplazione, le reti di telecomunicazione, la commutazione, trasmissione di segnali digitali, campionamento e quantizzazione di un segnale analogico
- Sicurezza: Backup, Https, Virus, Crittografia, Firma digitale, …
- Le reti, Amministrazione di una rete

Novità

- Linguaggi formali: Grammatiche, riconoscitori automatici,...
- Origini ed evoluzione dell'informatica
- Legislazione:
  - Il **codice per la protezione dei dati personali,codice della privacy,** il Decreto legislativo 30 giugno 2003, n. 196, in vigore dal 1º gennaio 2004,
  - D.Lgs. n. 235 del 30 dicembre 2010 sul nuovo Codice dell'amministrazione digitale
  - Provvedimento del Garante sulla privacy per gli amministratori di sistema del 27 novembre 2008
  - Legge 48 del 18 marzo 2008 sul crimine informatico
  - DL 31 dicembre 2007, n. 248 con l'art. 34 per la proroga della Legge 31 luglio 2005, n. 155 sull'antiterrorismo al 31/12/2008.
  - "Linee guida del Garante per posta elettronica e Internet", Gazzetta Ufficiale n. 58 del 10 marzo 2007.
  - Legge 231/2007 su antiriciclaggio http://www.camera.it/parlam/leggi/deleghe/testi/07231dl.htm
  - D. Lgs.82/2005 - Codice dell'Amministrazione Digitale.
  - Legge 31 luglio 2005, n. 155 sull'antiterrorismo.
  - D.Lgs. 9 aprile 2003 n. 68 - Attuazione della direttiva 2001/29/CE sull'armonizzazione di taluni aspetti del diritto d'autore e dei diritti connessi nella società dell'informazione.

- D.Lgs. 196/2003 Codice in materia di protezione dei dati personali.
- Direttiva del Presidente del Consiglio dei Ministri - 16 gennaio 2002. Dipartimento per l'Innovazione e le Tecnologie pubblicata sulla G.U. n.69 del 22 marzo 2002 "Sicurezza Informatica e delle Telecomunicazioni nelle Pubbliche Amministrazioni Statali".
- D.Lgs. 231/ 2001 - Disciplina della responsabilita' amministrativa delle persone giuridiche, delle societa' e delle associazioni anche prive di personalita' giuridica, a norma dell'articolo 11 della legge 29 settembre 2000, n. 300 http://www.camera.it/parlam/leggi/deleghe/testi/01231dl.htm
- Legge 18 agosto 2000 n. 248 contenente nuove norme di tutela del diritto d'autore. Legge 547/93 sulla criminalità informatica.
- Legge 633/41 e D. Lgs 518/92 Titolarità diritti sul software (legge diritto d'autore).
- **Codice Penale**
- Art. 615 ter c.p. Accesso abusivo ad un sistema informatico o telematico.
- Art. 615 quater c.p. Detenzione e diffusione abusiva di codici di accesso a sistemi informatici e telematici.
- Art. 615 quinquies c.p. Diffusione di programmi diretti a danneggiare o interrompere un sistema informatico.
- Art. 640 ter c.p. Frode informatica.
- **Direttive Europee**
- Direttiva 2002/21/CE del Parlamento europeo e del Consiglio del 7 marzo 2002, che istituisce un quadro normativo comune per le reti ed i servizi di comunicazione elettronica (direttiva quadro).
- Direttiva 2006/24/CE del Parlamento Europeo e del Consiglio del 15 marzo 2006 riguardante la conservazione di dati generati o trattati nell'ambito della fornitura di servizi di comunicazione elettronica accessibili al pubblico o di reti pubbliche di comunicazione (modifica la direttiva 2002/58/CE).
- Linee Guida dell'OCSE sulla sicurezza dei sistemi e delle reti d'informazione: verso una cultura della sicurezza del luglio 2002.

- Origini ed evoluzione dell'informatica
- Linguaggi formali: Grammatiche, riconoscitori automatici,...
- Alan Turing (la macchina di Turing, la crittografia, l'intelligenza artificiale)

Telecomunicazione

- Elettronica analogica
- Elettronica digitale

- I mezzi trasmissivi

$$D_{n\,x\,n} = \begin{pmatrix} d_{11} & 0 & 0 & \dots & 0 \\ 0 & d_{22} & 0 & \dots & 0 \\ 0 & 0 & d_{33} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & d_{nn} \end{pmatrix}$$

- la *matrice identità*: caso particolare di matrice diagonale avente tutti gli elementi della diagonale principale uguali a 1

aij=0 per i diverso da j e *aij=1 per i = j*

$$I_{n\,x\,n} = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{pmatrix}$$

- la *matrice simmetrica*: una matrice quadrata con aij=aji per ogni i e j, con i diverso da j

$$S = \begin{pmatrix} 2 & 1 & 3 \\ 1 & 5 & 10 \\ 3 & 10 & 8 \end{pmatrix}$$

- la *matrice triangolare*: una matrice quadrata i cui elementi al di sopra (triangolare inferiore) o al di sotto (triangolare superiore) della diagonale sono tutti nulli. Ad esempio, la seguente matrice T è triangolare superiore:

$$T_{n\,x\,n} = \begin{pmatrix} t_{11} & t_{12} & t_{13} & \dots & t_{1n} \\ 0 & t_{22} & t_{23} & \dots & t_{2n} \\ 0 & 0 & t_{33} & \dots & t_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & t_{nn} \end{pmatrix}$$

Algebra delle matrici

- ***Somma di due matrici***

Siano A e B due matrici m x n, ossia (attenzione) con le stesse dimensioni. La somma di A e di B è una matrice C m x n i cui elementi sono la somma degli elementi aventi la stessa posizione in A e in B:

cij=aij+bij per ogni i e j

Per la somma di due matrici valgono le seguenti proprietà:

- A+(B+C) = (A+B)+C **è** ASSOCIATIVA
- A+B = B+A **è** COMMUTATIVA

- ***Prodotto di un numero reale per una matrice***

Sia A una matrice m x n e r un numero reale. Si dice prodotto di r per A la matrice C i cui elementi sono quelli della matrice A moltiplicati per r:

cij=raij per ogni i e j

- ***Trasposta di una matrice***

Sia **A** una matrice **m x n**, si dice trasposta di A, e si indica con **AT**, la matrice **n x m** ottenuta da A scambiando ordinatamente le righe con le colonne:

$$A = \begin{pmatrix} 2 & 3 & 1 \\ 4 & 7 & 5 \end{pmatrix} \qquad A^T = \begin{pmatrix} 2 & 4 \\ 3 & 7 \\ 1 & 5 \end{pmatrix}$$

Si fa notare che se una matrice è simmetrica allora AT=A.

- ***Prodotto di due matrici***

Siano **A** una matrice **m x q** e **B** una matrice **q x n**, ossia tali che (attenzione) il numero delle colonne della prima matrice sia uguale al numero delle righe della seconda, si definisce matrice prodotto P la matrice i cui elementi pij sono ottenuti come somma dei prodotti degli elementi della riga i-esima di A per gli elementi della colonna j-esima di B:

$$p_{ij} = \sum_{k=1}^{q} a_{ik} b_{kj} \qquad per\ ogni\ i\ e\ j$$

Esempi

1. Prodotto matrice-matrice:

$$A = \begin{pmatrix} 2 & 3 & 1 \\ 4 & 7 & 5 \end{pmatrix} \qquad B = \begin{pmatrix} 1 & 3 \\ 2 & 5 \\ 1 & 1 \end{pmatrix}$$
$$C = AB = \begin{pmatrix} 2+6+1 & 6+15+1 \\ 4+14+5 & 12+35+5 \end{pmatrix} = \begin{pmatrix} 9 & 22 \\ 23 & 52 \end{pmatrix}$$

2. Prodotto matrice-vettore colonna

$$A = \begin{pmatrix} 2 & 3 \\ 4 & 6 \end{pmatrix} \qquad B = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$$
$$C = AB = \begin{pmatrix} 4+3 \\ 8+6 \end{pmatrix} = \begin{pmatrix} 7 \\ 14 \end{pmatrix}$$

Per il prodotto di due matrici valgono le seguenti proprietà:

- A(BC) = (AB)C **è** ASSOCIATIVO
- AB != BA **non è** COMMUTATIVO

Scriviamo un semplice programma che dopo aver caricato una matrice, A, di dimensione *n x m* e averla visualizza a video, calcola la trasposta AT e il prodotto P = AAT.

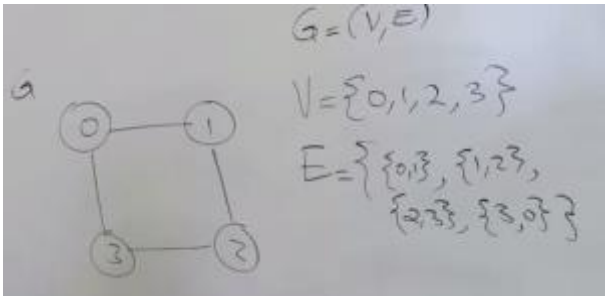
```
#include <iostream>

using namespace std; const int N=2, M=3; void leggiMatrice(int matrice[N][M]);

void visualizzaMatrice(int matrice[N][M]);
```

```
void calcolaVisualizzaTrasposta(int m[N][M], int t[M][N]);
void calcolaVisualizzaProdotto(int m1[N][M],int m2[M][N], int prodotto[N][N]); int main(){
   int matrice[N][M], trasposta[M][N], prodotto[N][N];
   leggiMatrice(matrice);
   visualizzaMatrice(matrice);
   calcolaVisualizzaTrasposta(matrice,trasposta);
   calcolaVisualizzaProdotto(matrice,trasposta,prodotto);
} void leggiMatrice(int matrice[N][M]){
    cout << "Inserisci la matrice " << N << " x " << M << ": " <<endl;
    for (int i=0; i<N; i++)
       for(int j=0; j<M; j++)
          cin >> matrice[i][j];
    return;
}
void visualizzaMatrice(int matrice[N][M]){
    cout << "La matrice inserita e': " <<endl;
    for (int i=0; i<N; i++){
       for(int j=0; j<M; j++){
          cout << matrice[i][j] << "\t";
       }
       cout <<endl;
    }
    return;
}
void calcolaVisualizzaTrasposta(int m[N][M], int t[M][N]){
    cout << "La matrice trasposta e': " <<endl;
    //determina la matrice trasposta
    for (int i=0; i<N; i++){
       for(int j=0; j<M; j++){
          t[j][i]=m[i][j];
       }
```

```
    }
    //visualizza la matrice trasposta
    for(int i=0; i<M; i++){
       for(int j=0; j<N; j++){
          cout << t[i][j] << "\t";
       }
       cout <<endl;
    }
    return;
} void calcolaVisualizzaProdotto(int m1[N][M],int m2[M][N], int prodotto[N][N]){
   int k;
   //determina la matrice prodotto
   for(int i=0; i<N; i++){ //i: blocca la riga di m1 da moltiplicare m1[i][?]
      for(int j=0; j<N; j++){ //j: blocca la colonna di m2 da moltiplicare m2[?][j]
         prodotto[i][j] = 0;
         for(k=0; k<M; k++){
            prodotto[i][j] += m1[i][k] * m2[k][j];
         }
      }
   }
   //visualizza la matrice prodotto
   cout << "La matrice prodotto delle due e': " <<endl;
   for (int i=0; i<N; i++){
      for(int j=0; j<N; j++){
         cout << prodotto[i][j] << "\t";
      }
      cout <<endl;
   }
   return;
}
```

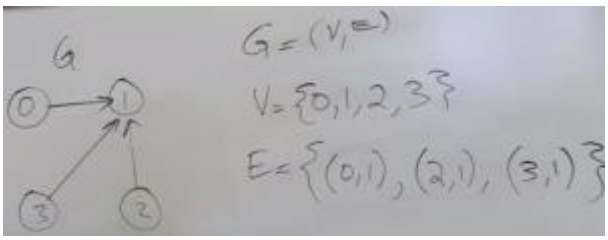
GRAFI Effettuiamo innanzitutto una distinzione fra due tipi di grafi:

- **grafi non-orientati** (o **indiretti**): viene formalmente descritto da due elementi V ed E. V è l'insieme dei nodi che deve essere finito e non vuoto, mentre E è l'insieme dei lati non-orientati. Un lato non-orientato può essere visto essenzialmente come l'insieme dei due nodi che vengono collegati dal lato stesso. Un esempio di grafo non orientato può essere dato dall'immagine sottostante:



in cui il grafo G è costituito dalla coppia V ed E. L'insieme dei vertici è costituito da 0, 1, 2, 3 e l'insieme dei lati è costituito dagli insiemi {0,1}, {1,2}, {2,3}, {3,0}. Visto che i lati non possiedono un verso (sono non-orientati) queste coppie possono essere rappresentate dall'insieme contenente i due vertici in cui l'ordine non conta, come si può vedere dalla figura. Come conseguenza del fatto che i lato sono non orientati, notiamo anche che non è necessario inserire nei lati entrambe le coppie (0,1) e (1,0), ma solamente l'insieme che non tiene conto dell'ordine. Inoltre non è importante nemmeno l'ordine con cui i vertici vengono elencati in V o con cui gli insiemi rappresentanti i lati vengono elencati in E.

grafi orientati (o **diretti**): viene formalmente descritto da due elementi V ed E. V è l'insieme dei nodi che deve essere finito e non vuoto, mentre E è l'insieme dei lati orientati. Un lato orientato può essere visto essenzialmente come la coppia formata dai due nodi, dove il primo è la coda e il secondo la testa, verso cui punta il lato. Un esempio di grafo orientato può essere dato dall'immagine sottostante:



in cui il grafo G è costituito dalle coppie V ed E. L'insieme V è identico a quello del grafo non orientato precedente, dato che i vertici sono gli stessi. I lati E sono ora rappresentati da frecce per distinguere il vertice coda (quello da cui la freccia parte), dal vertice testa (quello verso cui la freccia punta). E viene ora indicato come l'insieme delle coppie ordinate (0,1), (2,1) e (3,1).

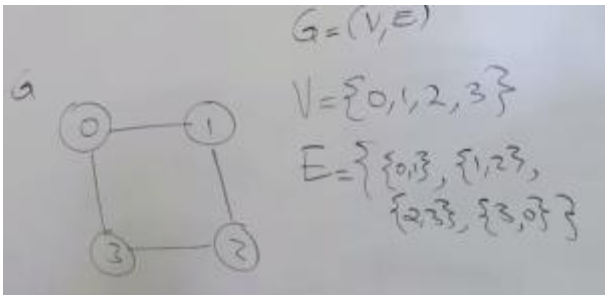
In questo corso non ammetteremo la presenza di lati paralleli, ovvero il caso in cui ci siano ad esempio due lati da 0 ad 1, né la presenza di cicli su un singolo elemento, ad esempio un lato che commette 0 con se stesso. Da notare il fatto che nei grafi orientati la presenza di due lati, uno che va dall'elemento A all'elemento B e uno che va dall'elemento B all'elemento A non sono considerati lati paralleli, lo sarebbero solo se avessero lo stesso verso.

Quello che abbiamo visto finora sono rappresentazioni grafiche di grafi. Noi avremo bisogno di utilizzare i grafi come input per i nostri algoritmi e, ovviamente, non potremo dar loro in pasto questa rappresentazione grafica, sperando che possano comprenderla e analizzarla. Dobbiamo quindi rappresentare i grafi in modo differente. Neanche la rappresentazione mediante gli insiemi V ed E è adatta da utilizzare all'interno degli algoritmi, non perché un algoritmo non possa analizzarla, ma per problemi di occupazione di spazio in memoria, man mano che i grafi aumentano di dimensione.

Rappresentazione

Le rappresentazioni più comuni che vengono utilizzate in informatica sono:

liste di adiacenza: la lista di adiacenza è composta da coppie. Esiste una coppia per ogni nodo del grafo. Il primo elemento della coppia è il nodo che si sta analizzando, il secondo è l'insieme formato da tutti i vertici ad esso adiacenti, ovvero a lui collegati tramite un lato. Se consideriamo ad esempio il grafo non orientato di prima:

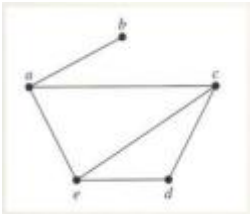


i legami presenti saranno i seguenti: NodiNodi Adiacenti0{1,3}1{0,2}2{1,3}3{0,2} dove l'ordine dei vertici all'interno dell'insieme di adiacenza è ininfluente. Da notare il fatto che la rappresentazione tramite liste di adiacenza di un grafo indiretto porta ad avere una ridondanza. Ad esempio indica che il nodo 0 è collegato al nodo 1, ma anche che il nodo 1 è collegato al nodo 0. Le liste di adiacenza non ci permettono di eliminare questa ridondanza.

Per un grafo orientato la ridondanza non è più presente, in quanto come primo elemento della coppia si avrà il nodo coda e come elementi dell'insieme tutti i nodi testa dei lati che partono dal primo.

In questo caso, se si avesse sia il legame fra 0 ed 1 e fra 1 e 0, significherebbe che esistono due lati uno da 0 ad 1 e uno da 1 a 0.

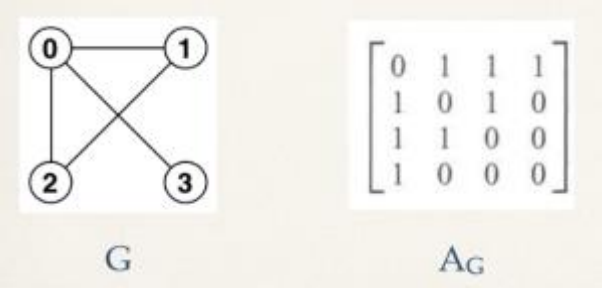
Vediamo un altro esempio per maggiore chiarezza:



Node	Adjacent nodes
a	b, c, e
b	a
c	a, d, e
d	c, e
e	a, c, d

matrici di adiacenza: le matrici di adiacenza sono matrici quadrate, aventi come dimensione il numero di nodi presenti nel grafo. Ad esempio se il nostro grafo avesse N nodi, la matrice avrebbe N righe ed N colonne. Se consideriamo il primo esempio del punto precedente, avremo una matrice

4x4, dove sia le righe rappresenteranno i nodi da 0 a 3 e lo stesso varrà per le colonne. Supponendo di chiamare la matrice A, l'elemento alla riga i e colonna j (indicato con A[i,j]), varrà 1 se il nodo i è collegato da un lato al nodo j, o alrimenti. Si otterrà quindi: 0123**0**0101**1**1010**2**0101**3**1010
Vediamo un altro esempio per maggiore chiarezza:



E' importante notare che se consideriamo un grafo non-orientato avremo sempre una simmetrica, cosa che non avviene nel caso in cui il grafo sia orientato. Come nella rappresentazione a liste di adiacenza, nel caso in cui abbiamo un grafo indiretto, avremo una certa ridondanza.

Per quanto riguarda la scelta del metodo di rappresentazione possiamo dire che, se il grafo ha pochi nodi, è conveniente utilizzare una lista di adiacenza, nel caso in cui abbiamo molti nodi è invece utile usare una matrice di adiacenza. Programmare Grafi in Python

Logica e linguaggi formali - Un percorso didattico integrato di matematica e informatica ***Rappresentazione di informazioni*** Le tabelle, i grafi, i grafici, impostazione di un foglio elettronico per risolvere problemi di varia natura (in particolare quelli matematici da sviluppare in modo interdisciplinare e pluridisciplinare); ***Calcolo degli enunciati e algebra di boole*** Operazioni con gli enunciati e valori di verità, l'implicazione, algebra di Boole Rappresentazione dei dati e Aritmetica dell'orologio ***Computabilità e macchina di Turing*** Automi, Automi di Mealy e Moore Macchine combinatorie e macchine sequenziali Rappresentazione degli automi a stati finiti Diagrammi degli stati La matrice di transizione La macchina si Turing Tesi di Church (computabilità) ***Linguaggi e grammatica*** Grammatiche generative Metalinguaggio Diagrammi sintattici Automi riconoscitori ***Programmazione Logica*** Problemi euristici Problemi algoritmici Problemi Logici Rappresentazione della conoscenza: fatti e regole Obiettivi logici Processo di unificazione Backtracking Cut e fail

*1 - **Logo***

Riflessioni, esercizi e coding per potenziare il pensiero computazionale

Problema: Tre missionari e tre cannibali cercano di attraversare un fiume dalla riva sinistra alla riva destra. C'è una barca che ha due posti e che può navigare con una qualsiasi combinazione di missionari e cannibali di una o due persone. In qualunque momento, se i missionari su una riva del

fiume sono in minoranza rispetto ai cannibali, questi ultimi cederanno alle loro tendenze antropofaghe. Quando la barca è ormeggiata la si considera appartenente alla riva.

trovare la più semplice successione di traversate che consenta a tutti i missionari e i cannibali di attraversare il fiume sani e salvi.

· descriviamo in modo simbolico gli stati delle rive del fiume in questo modo:

per es. M M C C / BARCA M C significa che sulla riva sx ci sono 2 missionari e due cannibali e sulla riva dx la barca un missionario e un cannibale.

· rappresentiamo gli stati e gli oggetti come liste e i movimenti da una riva all'altra (mosse) con procedure di modifica di tali liste.

es. STATO M M C C / BARCA M C con

rivasx [M M C C]

rivadx [BARCA M C]

es. MOSSA con

una lista di oggetti da trasferire

[M C BARCA]

· i tentativi di soluzione del problema possono essere rappresentati da un ramo di un albero in cui i nodi sono etichettati da stati delle rive e gli archi da mosse. In tale albero una soluzione è rappresentata da un ramo che va dallo stato iniziale allo stato finale cioè:

[M M M C C C BARCA] / [ ] ® [ ] / [M M M C C C BARCA]

· la soluzione può essere trovata esplorando uno per uno tutti i rami dell'albero.

· se proviamo a risolvere il problema tentando una dopo l'altra sequenze di traversate il nostro comportamento può essere descritto come una ricerca in "profondità" lungo l'albero.

La **mossa**, quindi, rappresenta una lista di "oggetti" che denominiamo MOVLIST da trasferire dalla lista RSIN alla RDES o viceversa, una procedura "muovi-da-sinistra-a-destra" denominata MSD, una procedura "muovi-da-destra-a-sinistra" denominata MDS.

procedura MSD

assegna alla lista RSIN la lista RSIN senza gli elementi di MOVLIST

assegna alla lista RDES la lista RDES con gli elementi di MOVLIST

fine

La barca è sempre inclusa nella MOVLIST.

Sviluppiamo un programma **logo** che permetta di controllare le potenziali soluzioni del problema dei missionari

“a mano” , usando il computer solo per tenere nota dello stato in cui siamo e, poi, possiamo migliorare il programma verificando automaticamente la soluzione del problema. Problema :Un contadino deve attraversare un fiume portando con sè un lupo, una capra e un cavolo. Egli dispone di una barca che può trasportare uno solo dei tre “passeggeri” oltre se stesso. Trovare una sequenza di azioni per portare il contadino sull’altra sponda con le sue cose tenendo presente che, in assenza del contadino, il lupo mangerebbe la capra e la capra mangerebbe il cavolo. Codifica logo e prolog

LISP

- Linguaggi imperativi: orientati alla descrizione di algoritmi; strettamente legati al modello di macchina detto di von Neumann (questo gruppo comprende quasi tutti i linguaggi noti);
- Linguaggi dichiarativi: approccio matematico, legato al concetto di funzione, come il Lisp, oppure in termini di relazioni logiche (calcolo dei predicati),come il Prolog.

L'idea di fondo che il Lisp in attesa di una espressione da valutare. Legge un'espressione, la valuta e ne scrive il valore: si dice che l'interprete si trova in una read-eval-print loop. In Lisp la possibilità di definire nuove funzioni fondamentale! Si usa la forma defun che prende come argomenti il nome della nuova funzione, la lista dei suoi argomenti e il corpo (body) che la definisce. Il Lisp restituisce sempre un valore. Paradigma funzionale Introduzione al Lisp Esempi

;Problema: definire una funzione che, dato un numero restituisca il suo cubo. (defun CUBO (N)

(* N N N)

); Eccoi una possibile chiamata della funzione:

;[1]> (cubo 5)

; 125

; se la funzione è stata definita in un file, per caricare cubo.lsp [1]>(load "cubo.lsp) ;Problema : definire una funzione che determina il fattoriale di un numero. (defun fatt (n) (if (zerop n) 1 (* n (fatt (1- n)))))

;DEFINIZIONE RICORSIVA DI UNA FUNZIONE

;Problema: definire una funzione che restituisca la lunghezza di una lista

; [1]> (LUNG '(R U S))

;3 (defun LUNG (L)

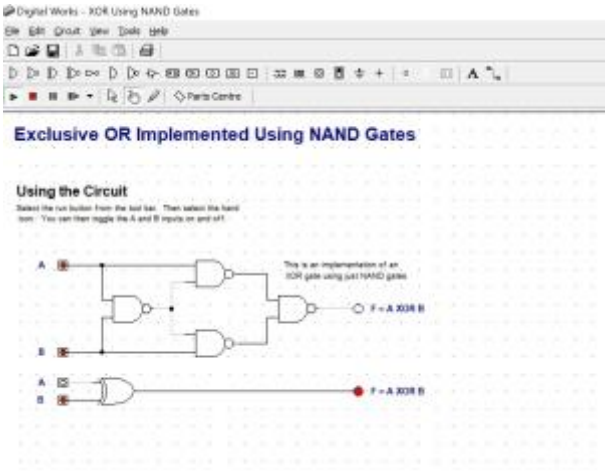
(cond

((null L) 0)

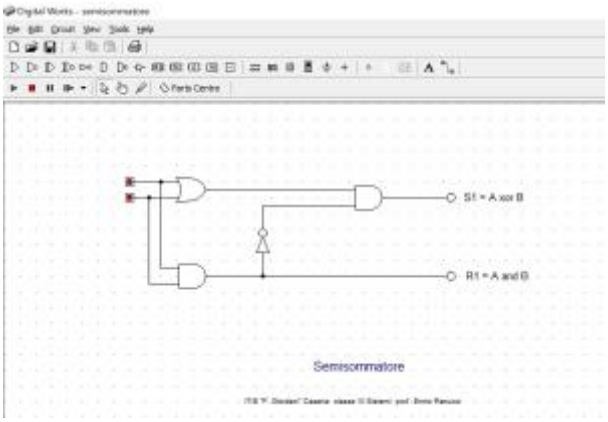
(t (+ (LUNG (cdr L)) 1))

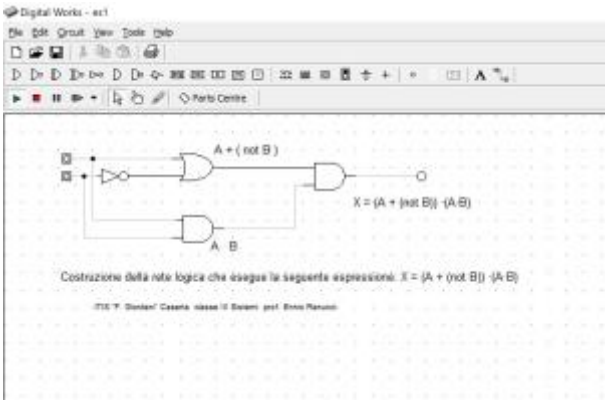
)

) ; t sta per true



2 - **Simulazione di circuiti digitali per potenziare il pensiero computazionale** *Realizzare un OR Esclusivo usando porte NAND*





Algebra di Boole e i suoi modelli:

Algebra delle Proposizioni e Algebra dei Circuiti