In questo libretto sono illustrati i quesiti proposti nella seconda edizione del Kangourou dell'Informatica, fase eliminatoria, che si è svolta il 10 marzo 2010. Il software con il quale i quesiti sono stati presentati ai concorrenti è scaricabile dal sito `http://kangourou.dico.unimi.it/`. La gara ha impegnato squadre di quattro persone. Durante la competizione era consentito servirsi di libri, appunti, ricerche in rete.

La gara era suddivisa in due categorie: "Medie" per gli studenti della scuola secondaria di primo grado e "Biennio" per gli studenti delle classi prima e seconda della scuola secondaria di secondo grado.

Il libretto, che fa seguito all'analoga iniziativa dello scorso anno, si rivolge sia agli alunni, che abbiano o no partecipato alla gara, sia agli insegnanti, nell'intento di proporre qualche approfondimento e di rinnovare l'interesse e il divertimento suscitati dai quesiti e dalla gara.

I quesiti, cosí come riportati nelle due categorie di gara, sono presentati nella prima parte. Nella seconda sono raccolte le soluzioni, alcuni suggerimenti su come ottenerle, un'indicazione di possibili difficoltà o errori e un cenno piú o meno ampio al contesto in cui il quesito può essere inquadrato nell'ambito dell'informatica; vengono anche individuate delle parole chiave che possono essere utili per ricerche in rete o per trovare connessioni tra i diversi quesiti proposti.

Naturalmente lo scopo ultimo è promuovere l'informatica come disciplina scientifica.

Violetta Lonati
Mattia Monga
Anna Morpurgo
Lorenzo Repetto
Mauro Torelli

## Quesiti per la categoria “Medie”



## Quesiti per la categoria “Biennio”

# Quesiti per la categoria "Medie"

# Salvaschermo lampeggiante ( max 6 punti )

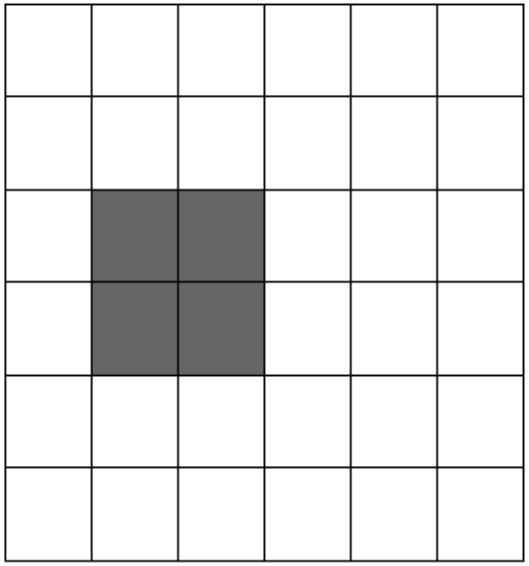
Il signor Lexip vuole produrre un salvaschermo in cui i pixel dello schermo sono sincronizzati (i pixel cambiano stato tutti contemporaneamente, in base al numero di pixel accesi prima del cambiamento) e ad ogni secondo ciascun pixel passerà da acceso a spento o viceversa secondo le regole seguenti:

- se un pixel è acceso e meno di due o più di tre degli otto pixel ad esso adiacenti sono pure accesi, allora si spegnerà;
- se un pixel è spento ed esattamente tre degli otto pixel ad esso adiacenti sono accesi, allora si accenderà;
- in tutti gli altri casi, il pixel rimane nello stato in cui si trova.

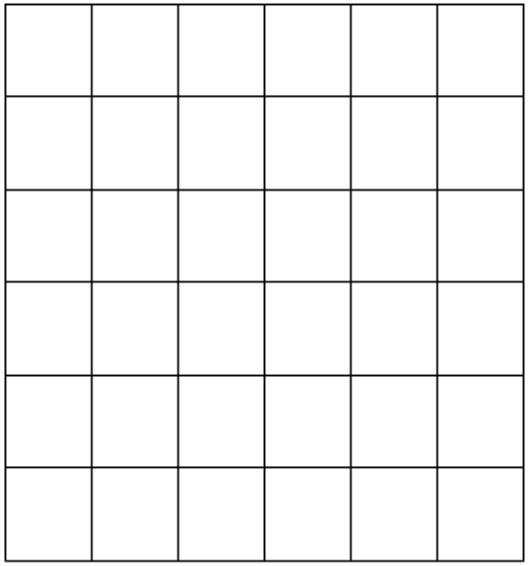
Supponendo che un quadratino nero rappresenti una cella accesa e un quadratino bianco una cella spenta, disegnare per il display 1 e per il display 2 che cosa apparirà nei successivi due secondi.

**Display 1 (2 punti)** | **Display 2 (4 punti)**

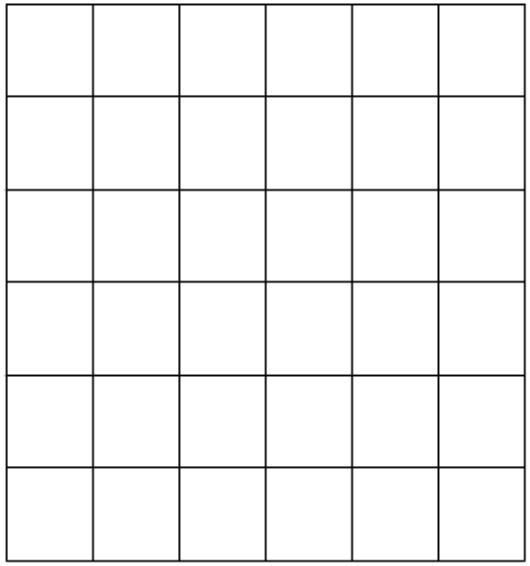
Tempo 0:



Tempo 1:



Tempo 2:



# Salvaschermo lampeggiante ( max 6 punti )

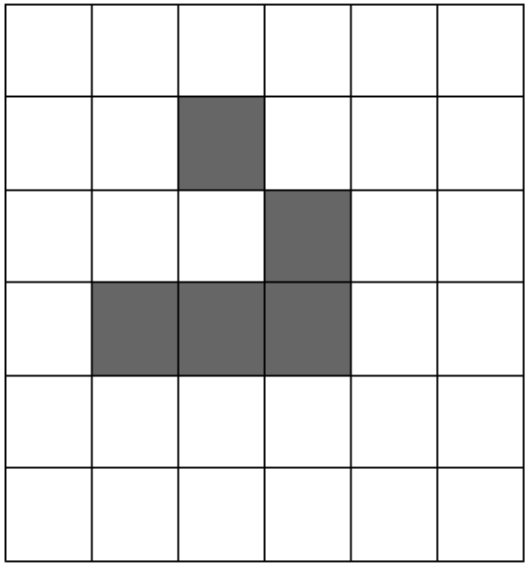
Il signor Lexip vuole produrre un salvaschermo in cui i pixel dello schermo sono sincronizzati (i pixel cambiano stato tutti contemporaneamente, in base al numero di pixel accesi prima del cambiamento) e ad ogni secondo ciascun pixel passerà da acceso a spento o viceversa secondo le regole seguenti:

- se un pixel è acceso e meno di due o più di tre degli otto pixel ad esso adiacenti sono pure accesi, allora si spegnerà;
- se un pixel è spento ed esattamente tre degli otto pixel ad esso adiacenti sono accesi, allora si accenderà;
- in tutti gli altri casi, il pixel rimane nello stato in cui si trova.

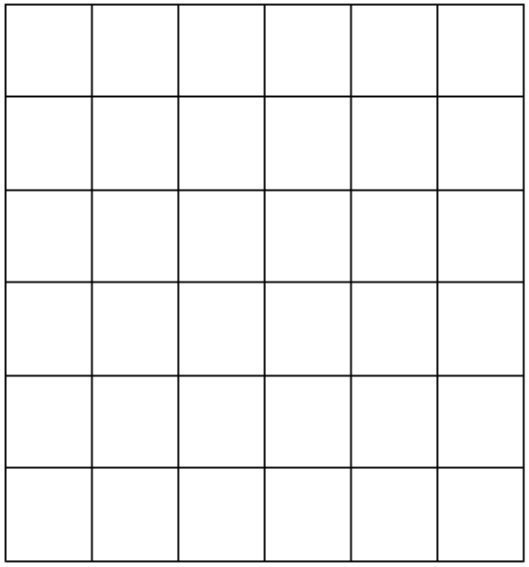
Supponendo che un quadratino nero rappresenti una cella accesa e un quadratino bianco una cella spenta, disegnare per il display 1 e per il display 2 che cosa apparirà nei successivi due secondi.

**Display 1 (2 punti)** **Display 2 (4 punti)**

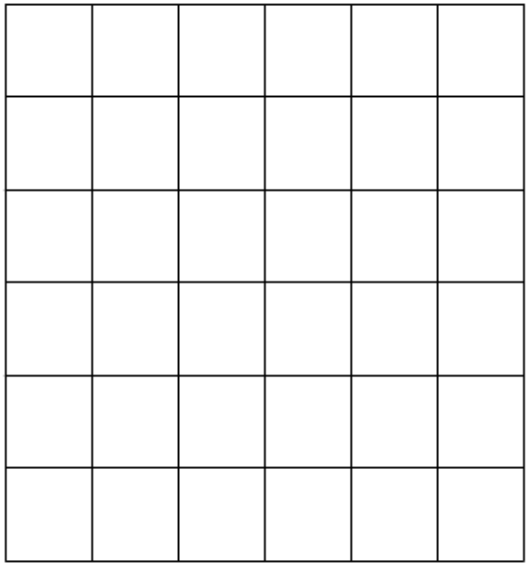
Tempo 0:



Tempo 1:



Tempo 2:



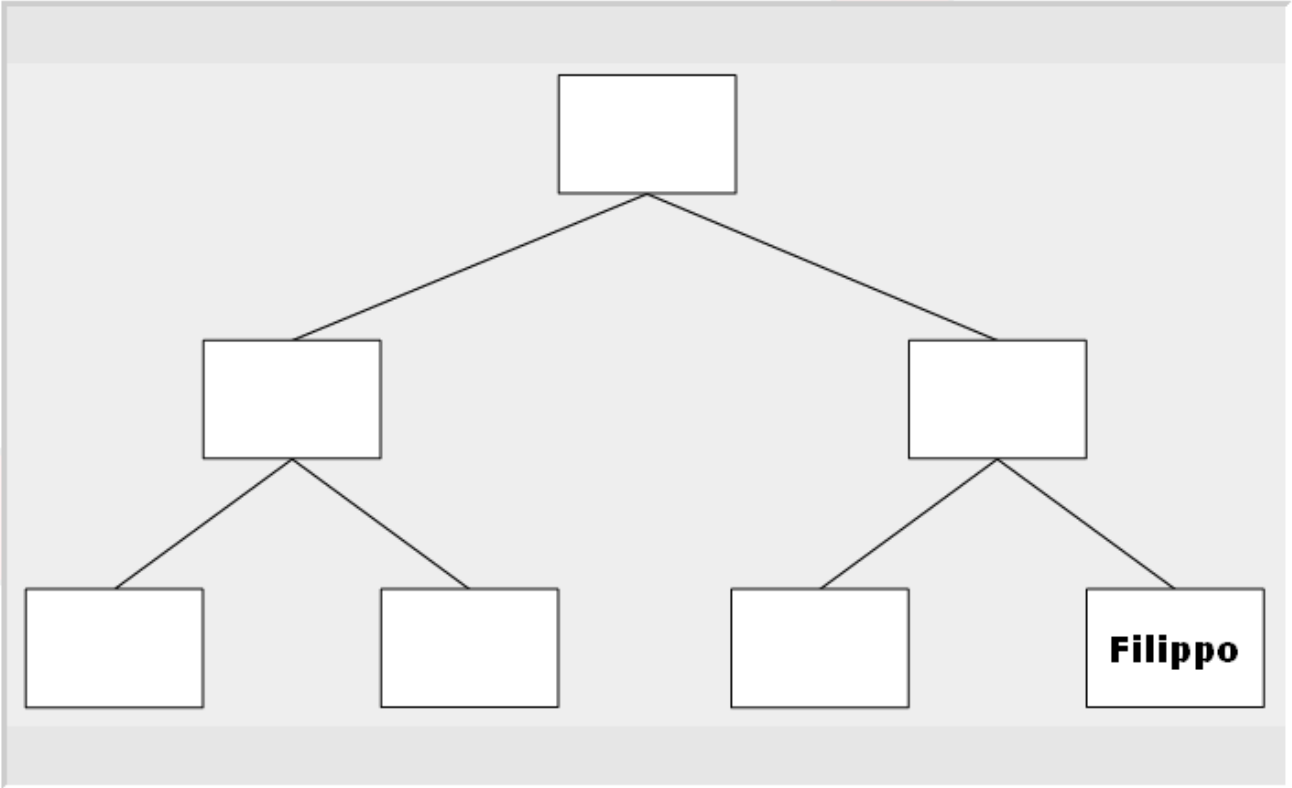
# La famiglia Pre ( max 6 punti )

Nella famiglia Pre vige una curiosa regola dei nomi: il nonno ha stabilito che a ogni nuovo nato della sua famiglia venga imposto il nome dell'ultimo nato con l'aggiunta, in fondo, di una nuova lettera. La famiglia festeggia la nascita dell'ultimo nato, Filippo, ed è particolarmente felice perché il nome del padre di ciascuno è lungo la metà (arrotondata per difetto quando il nome ha un numero dispari di lettere) del nome del figlio. Per esempio, il papà di Filippo si chiama Fil.

Completa l'albero genealogico inserendo un nome in ogni casella.

I figli devono essere ordinati in modo che il maggiore sia a sinistra.

Per ogni nome corretto, guadagnerai un punto.

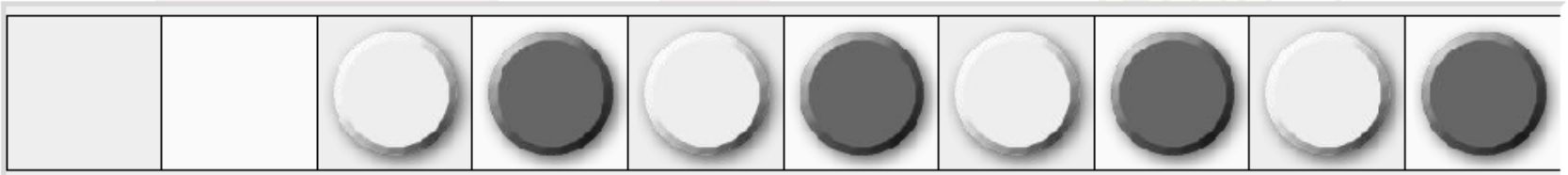


# Giochino giapponese ( max 6 punti )

In questo solitario, ad ogni mossa si possono spostare nelle due caselle libere due pedine qualsiasi, purché siano contigue e vengano lasciate nello stesso ordine.
Sei capace di passare da questa situazione iniziale:

a questa situazione finale?

Per fare una mossa, fai clic su una pedina per spostare lei e la sua vicina di destra nelle caselle vuote. Meno mosse farai, più alto sarà il punteggio.



# Colora la mappa ( max 6 punti )

Colora la mappa rispettando queste tre regole:
- scegli un colore per ogni regione
- evita di assegnare colori uguali a regioni confinanti
- meno colori userai, maggiore sarà il punteggio

Per cambiare il colore di una regione, fai click sulla regione stessa.

Qualche utile precisazione:
- non considerare San Marino e le isole minori
- la Sicilia e la Sardegna non confinano con nessuna regione
- il Piemonte confina con l'Emilia Romagna, ma la Lombardia non confina con la Liguria; il Lazio confina con le Marche, ma l'Umbria non confina con l'Abruzzo.

# Uno strano gioco dell'oca ( 6 punti )

Aldo ha inventato un nuovo gioco dell'oca da proporre ai suoi amici. Invece di tirare i dadi, ad ogni turno ciascuno deve far avanzare la propria pedina secondo le regole seguenti:

1. se la casella numerata su cui si trova la pedina ha un numero pari, si può scegliere tra due mosse:
   Mossa A: andare alla casella successiva;
   Mossa B: andare alla casella con numero doppio di quella attuale (per esempio, se sono alla casella 6 vado alla casella 12);
2. se la casella su cui si trova la pedina ha un numero dispari si può fare soltanto la mossa B (per esempio, se sono alla casella 5 devo andare alla casella 10).

Chi supera l'ultima casella (il traguardo) deve ripartire dalla casella numero 1.

Il gioco ha 27 caselle e si parte dalla casella numero 1. Qual è la sequenza di caselle che porta direttamente al traguardo?

Un vecchio proverbio cinese suggerisce
"lo stolto si ferma al principio, il saggio comincia dalla fine"...

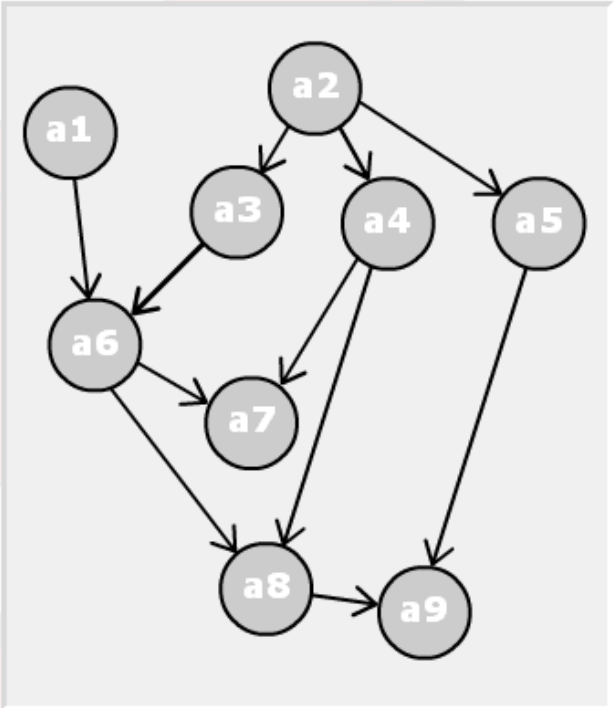
# Lavori di manutenzione ( 6 punti )

**Aldo, Bruno e Carlo stanno pianificando i lavori di manutenzione del loro "rifugio"... Come tecnici provetti, desiderano ultimare tutti i compiti nel più breve lasso di tempo. Così individuano ed elencano le diverse attività da svolgere, con le rispettive durate previste (in ore di lavoro) e le precedenze che devono essere rispettate.**

**Tralasciando qui le descrizioni dei lavori che costituiscono ciascuna attività, il loro piano dovrà rispettare questo schema:**

| Attività | Durata |
|---|---|
| a1 | 2 |
| a2 | 8 |
| a3 | 5 |
| a4 | 9 |
| a5 | 7 |
| a6 | 3 |
| a7 | 5 |
| a8 | 4 |
| a9 | 3 |

Il grafo disegnato qui a destra stabilisce le precedenze tra le attività: a1 e a2 possono cominciare subito; a2 è preliminare ad a3, a4 e a5, cioè queste potranno cominciare soltanto quando a2 sarà terminata; a6 potrà cominciare soltanto quando saranno terminate entrambe le attività a1 e a3, e così via. Ciascuna attività sarà svolta da uno solo dei tre ragazzi; anche se colui che si prende carico di una certa attività fosse aiutato da qualcuno dei suoi amici, la durata di quell'attività non cambierebbe...



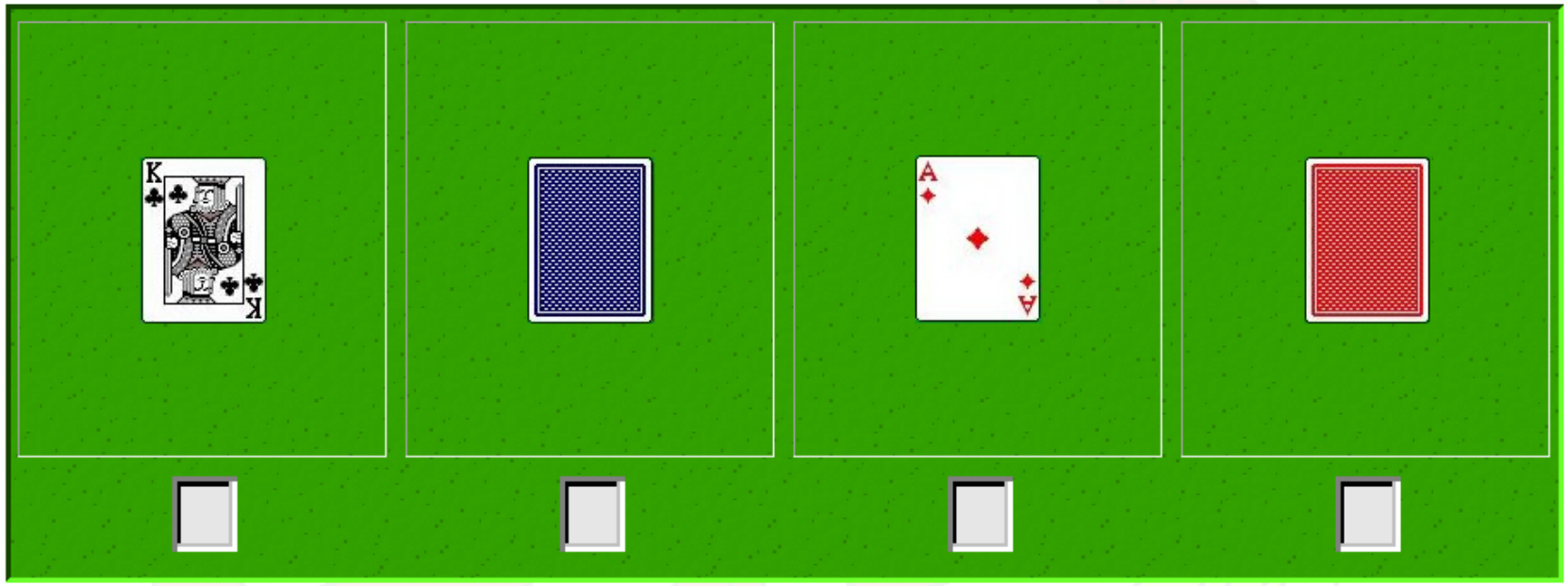

**In quante ore, come minimo, potranno essere completate tutte le attività previste?**

- 21
- 23
- 24
- 34
- 46

Nota: ai fini del punteggio conta solo la scelta del numero di ore; per aiutarti a trovare la soluzione, puoi associare un colore a ciascuno dei tre amici, quindi colorare le attività nel grafo (ad ogni clic cambierà il colore assegnato).

# Carte rosse e carte blu ( 6 punti )

Guarda le quattro carte sul tavolo: chissà se ogni carta col retro blu è un re...



Seleziona col mouse le carte che, girate, forniscono informazioni utili a stabilire con certezza se la seguente affermazione è vera: ogni carta col retro blu che si trova sul tavolo è un re.

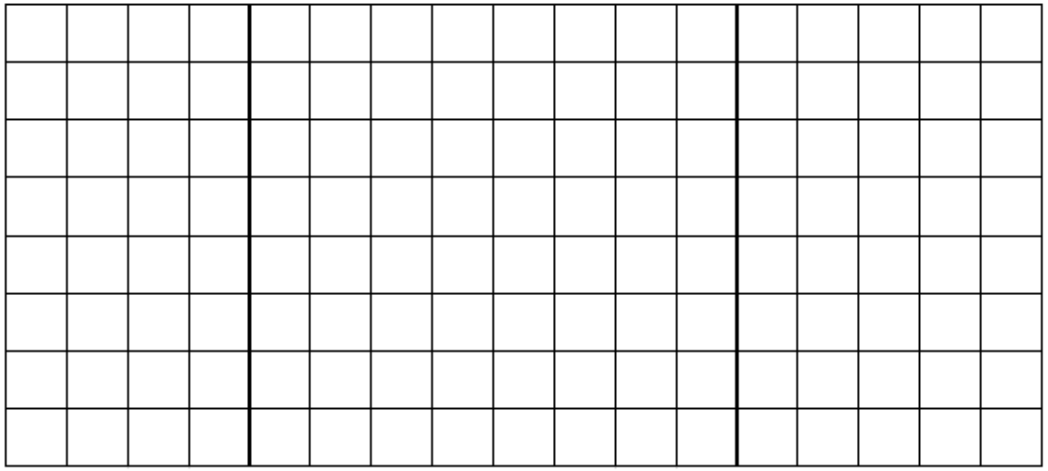
# I vasi di fiori ( 6 punti )

Il geometra Dueceli vuole sistemare i vasi di fiori nel suo cortile (rettangolo di base 17 e altezza 8 piastrelle, con i lati orientati secondo i quattro punti cardinali) in modo che formino un triangolo con la base lungo il lato Sud.

L'anno scorso aveva a disposizione 16 vasi di fiori e se l'è cavata con il procedimento mostrato a destra.

1- Parte dalla piastrella d'angolo a Nord-Ovest
2- Chiama P il numero di piastrelle sul lato corto del cortile
3- Chiama V il numero di vasi a disposizione
4- Segna su un foglio il numero 1
5- Calcola 1/4 * V e lo chiama N
6- Ripete P volte:
   a. Calcola P - (valore sul foglio)
   b. Si sposta a Est del numero di piastrelle calcolato al punto precedente
   c. Calcola 2*[ (valore sul foglio) - (P - N) ] - 1 e lo chiama M
   d. Se M > 0, sistema M vasi, uno per piastrella, a partire da quella in cui si trova e procedendo verso Est
   e. Aumenta di uno il valore scritto sul foglio
   f. Se possibile, si sposta verso Sud di una piastrella e raggiunge la piastrella più a Ovest

Posiziona nella griglia i 16 vasi usando il procedimento indicato.



# Crucipuzzle informatico ( max 10 punti )

| T | H | L | S | O | F | T | W | A | R | E |
|---|---|---|---|---|---|---|---|---|---|---|
| A | A | T | E | C | I | C | P | U | F | I |
| S | R | N | U | T | A | E | E | O | I | N |
| T | D | S | L | R | I | N | M | L | L | T |
| I | W | I | O | R | I | T | D | A | E | E |
| E | A | G | O | R | I | N | C | I | G | R |
| R | R | M | L | R | G | S | G | E | R | N |
| E | E | E | O | N | A | E | P | B | Z | E |
| M | E | G | G | P | A | J | N | I | R | T |
| T | L | I | I | F | I | C | I | T | A | L |
| A | P | I | N | G | U | I | N | O | E | I |

| Griglia e Definizioni | | Istruzioni |
|---|---|---|
| 1- | | procedimento, specificato rigorosamente e senza ambiguità, per la risoluzione di un problema |
| 2- | | cifra binaria |
| 3- | | acronimo inglese della "unità centrale di elaborazione" di un computer |
| 4- | | unità, costituita da una sequenza di byte e individuata da un nome, secondo cui sono organizzati i dati su un hard disk o altro dispositivo di memoria permanente |
| 5- | | insieme delle parti fisiche di un computer |
| 6- | | la rete delle reti |
| 7- | | un formato standard per rappresentare immagini |
| 8- | | procedura di accesso ad un sistema o un'applicazione informatica, in cui a volte si richiede anche l'inserimento di una password |
| 9- | | Io sono ad esempio la RAM, l'hard disk, una chiavetta USB |
| 10- | | linguaggio di programmazione così chiamato in onore di un grande matematico e filosofo francese |
| 11- | | mascotte del sistema operativo libero Linux |
| 12- | | acquisire un'immagine usando uno scanner |
| 13- | | la parte che non si tocca dell'informatica... |
| 14- | | si chiama così il codice di un programma formato da istruzioni appartenenti ad un determinato linguaggio di programmazione |
| 15- | | ce ne sono QWERTY e DVORAK |
| 16- | | matematico inglese, considerato uno dei padri dell'informatica, al quale è dedicato un premio internazionale |

# Crucipuzzle informatico ( max 10 punti )

**Griglia e Definizioni** | **Istruzioni**

All'interno della griglia sono nascoste 16 parole che hanno a che fare con l'informatica: possono trovarsi in verticale, dall'alto al basso, oppure in orizzontale o in diagonale (discendente o ascendente), da sinistra a destra. Inoltre, una parola può essere parzialmente sovrapposta ad altre.

Trovale e associa ciascuna di esse alla giusta definizione: ogni 2 parole corrette otterrai un punto. Per aiutarti, puoi colorare le caselle della griglia, con un clic.

Le lettere rimaste (lette da sinistra a destra e procedendo dall'alto al basso) formano un'espressione misteriosa: se riuscirai a scoprirla guadagnerai altri due punti!

L'espressione chiarirà il soggetto a cui si riferiva Dijkstra quando affermò che "chiedersi se una macchina può pensare è come chiedersi se un sommergibile può nuotare".

Espressione misteriosa: 

# Quesiti per la categoria "Biennio"

# Comic-con 2010 (max 6 punti)

Pippo non vede l'ora di partecipare alla fantasmagorica Comic-con, la fiera internazionale di fumetti, videogiochi, telefilm, fantascienza... La fiera è ricca di eventi, molti dei quali però si sovrappongono. Pippo decide di sceglierne solo alcuni, che non si sovrappongano (nemmeno parzialmente). Vuole però sceglierne il massimo numero possibile.

Quale delle seguenti strategie gli conviene seguire? (3 punti)

- ○ Sceglie le attività che finiscono prima, evitando le sovrapposizioni
- ○ Sceglie le attività più brevi, evitando le sovrapposizioni
- ○ Sceglie le attività che iniziano per prime, evitando le sovrapposizioni
- ○ Sceglie le attività che hanno meno sovrapposizioni con le altre

Quali attività sceglierà con questa strategia? (3 punti)

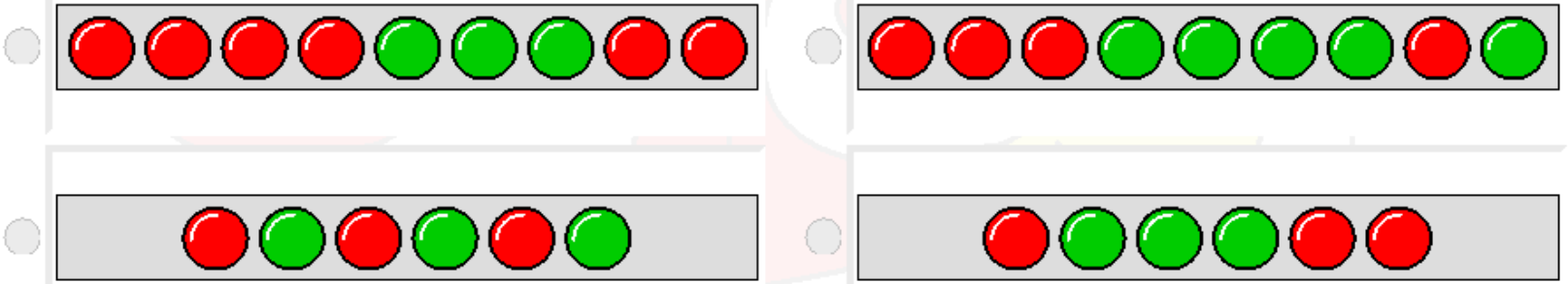
- ☐ Dalle 09.00 alle 10.00: inaugurazione della fiera con passerella dei vip
- ☐ Dalle 09.30 alle 11.30: Dave Mc Kean, illustratore del romanzo "Coraline", firma autografi
- ☐ Dalle 09.30 alle 11.30: proiezione del film "Avatar"
- ☐ Dalle 14.30 alle 16.00: presentazione del nuovo videogioco per la Wii
- ☐ Dalle 19.00 alle 20.30: sfilata delle Winx
- ☐ Dalle 11.00 alle 15.00: offerta speciale per l'acquisto di vecchi numeri di Topolino
- ☐ Dalle 15.30 alle 19.30: torneo di Playstation Pro Evolution Soccer 2009
- ☐ Dalle 10.30 alle 20.30: maratona de "Il signore degli anelli" in versione integrale
- ☐ Dalle 20.00 alle 21.00: trailer "Maga Martina e il libro magico" e chiusura dell'evento
- ☐ Dalle 19.00 alle 20.30: conferenza con gli sceneggiatori e gli attori del telefilm "Lost"



# Gessetti colorati ( 6 punti )

I figli dei signori Rossi e Verdi (entrambe famiglie molto numerose) hanno scoperto una stanza che contiene una lavagna e gessetti colorati a non finire. Per evitare che la stanza diventi troppo affollata, hanno deciso di mettere una scatola davanti alla porta, con gessetti rossi e verdi. L'ingresso è poi regolato in questo modo: quando qualcuno vuole entrare nella stanza deve prendere un gessetto dalla scatola, rosso per i componenti della famiglia Rossi e verde per i componenti della famiglia Verdi; altrimenti, se non ci sono gessetti del colore giusto a disposizione, non è permesso entrare. All'uscita dalla stanza è invece obbligatorio mettere nella scatola un gessetto del colore corrispondente all'altra famiglia.

Supponiamo che ciascuno, una volta nella stanza, disegni un solo pallino del colore della propria famiglia (in sequenza orizzontale da sinistra a destra, se ce ne sono già). Se più di una persona è nella stanza non è possibile sapere in che ordine verranno disegnati i pallini. Quale di queste sequenze NON può verificarsi se all'inizio nella scatola all'ingresso ci sono 5 gessetti rossi e 1 verde?



# La data della Pasqua ( 6 punti )

Il Concilio di Nicea del 325 stabilì la data della Pasqua cristiana nella domenica successiva al primo plenilunio dopo l'equinozio di primavera; ne risulta che può cadere tra il 22 marzo e il 25 aprile (compresi). Qui è descritto un algoritmo (l'originale, più complesso e generale, è dovuto a Gauss e risale all'800) per calcolare la data della Pasqua di un ANNO qualsiasi tra il 1900 e il 2099 compresi:

1. Poni A = resto della divisione tra ANNO e 19.
2. Poni B = resto della divisione tra ANNO e 4.
3. Poni C = resto della divisione tra ANNO e 7.
4. Poni D = resto della divisione tra (19*A + 24) e 30.
5. Poni E = resto della divisione tra (2*B + 4*C + 6*D + 5) e 7.
6. Se D è uguale a 28 ed E è uguale a 6, allora il risultato è il giorno 18 aprile. FINE.
7. Se D è uguale a 29 ed E è uguale a 6, allora il risultato è il giorno 19 aprile. FINE.
8. Poni F = D + E.
9. Se F è minore o uguale a 9, allora il risultato è il giorno (F + 22) marzo; altrimenti il risultato è il giorno (F - 9) aprile. FINE.

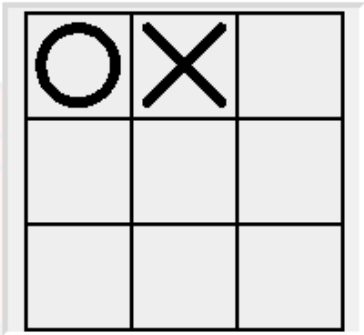
| | Anno 2021 |
|---|---|
| A | |
| B | |
| C | |
| D | |
| E | |
| F | |
| Giorno di Pasqua | |

Applicate questo algoritmo per calcolare in quale giorno cadrà la Pasqua nell'anno 2021, esplicitando i valori calcolati ad ogni passo.



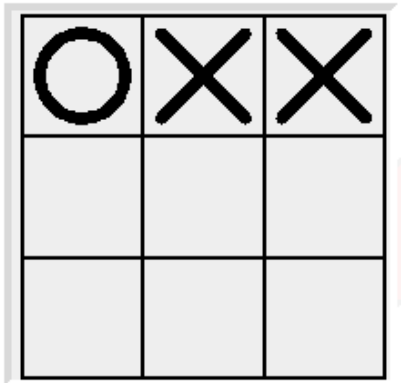
# Una partita a tris ( 6 punti )

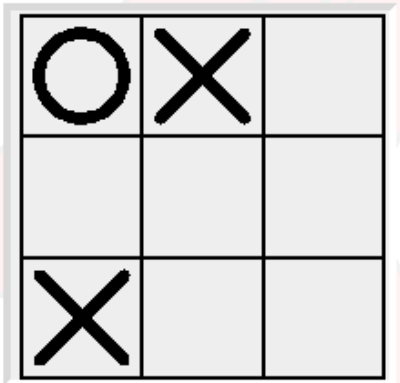
Sir Cross e Lady Circle stanno giocando una partita al gioco del tris (o "filetto"), che certamente tutti conoscete: vince chi per primo fa tris, in orizzontale o verticale o diagonale, e se nessuno fa tris la partita finisce in parità.

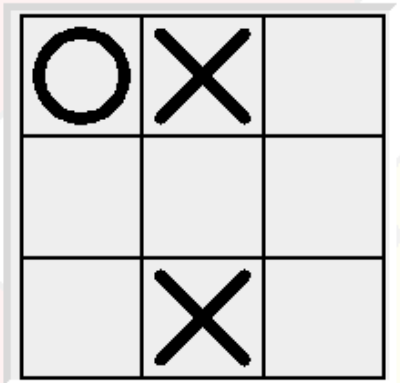


Ha iniziato Sir Cross, mettendo una croce in mezzo a uno dei lati; Lady Circle ha risposto disegnando un cerchio in uno degli angoli a fianco della croce, come mostrato nella figura qui a destra.

Quale tra le seguenti mosse deve fare adesso Sir Cross per assicurarsi almeno il pareggio?







Nessuna di quelle qui accanto:
Sir Cross ha ormai perduto la partita!

# Salvaschermo lampeggiante ( 6 punti )

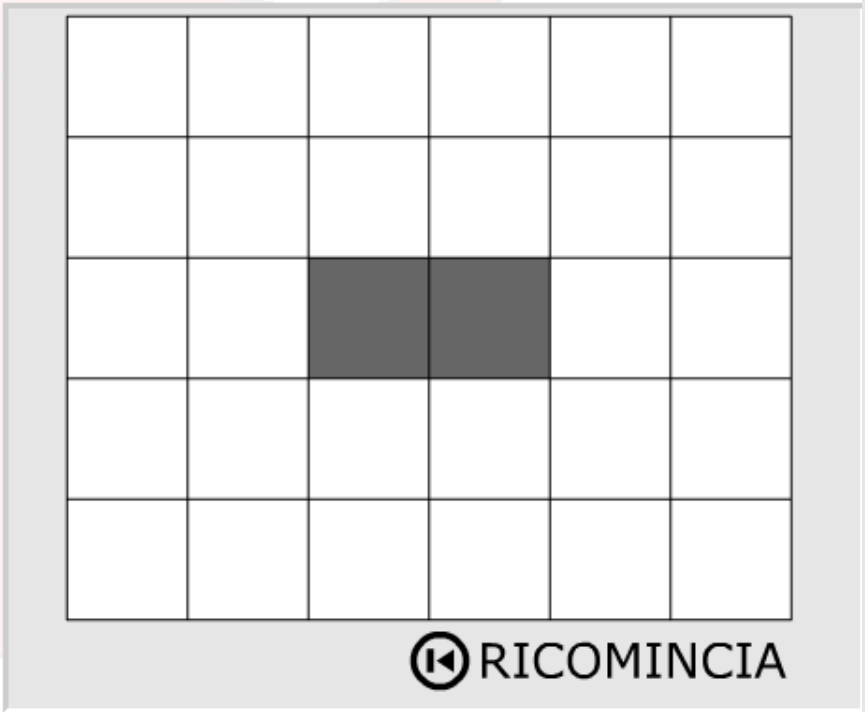
Il signor Lexip vuole produrre un salvaschermo in cui i pixel dello schermo sono sincronizzati (i pixel cambiano stato tutti contemporaneamente, in base al numero di pixel accesi prima del cambiamento) e ad ogni secondo ciascun pixel passerà da acceso a spento o viceversa secondo le regole seguenti:

- se un pixel è acceso e meno di due o più di tre degli otto pixel ad esso adiacenti sono pure accesi, allora si spegnerà;
- se un pixel è spento ed esattamente tre degli otto pixel ad esso adiacenti sono accesi, allora si accenderà;
- in tutti gli altri casi, il pixel rimane nello stato in cui si trova.

Affinché il salvaschermo sia efficace, la figura ad ogni secondo deve essere diversa da quella del secondo precedente.

Ma provando la figura a lato (in cui un quadratino nero rappresenta un pixel acceso e uno bianco uno spento) il signor Lexip si accorge che in breve lo schermo diventa fisso e completamente bianco!

In quale posizione deve aggiungere un solo quadratino nero per ottenere un salvaschermo efficace?



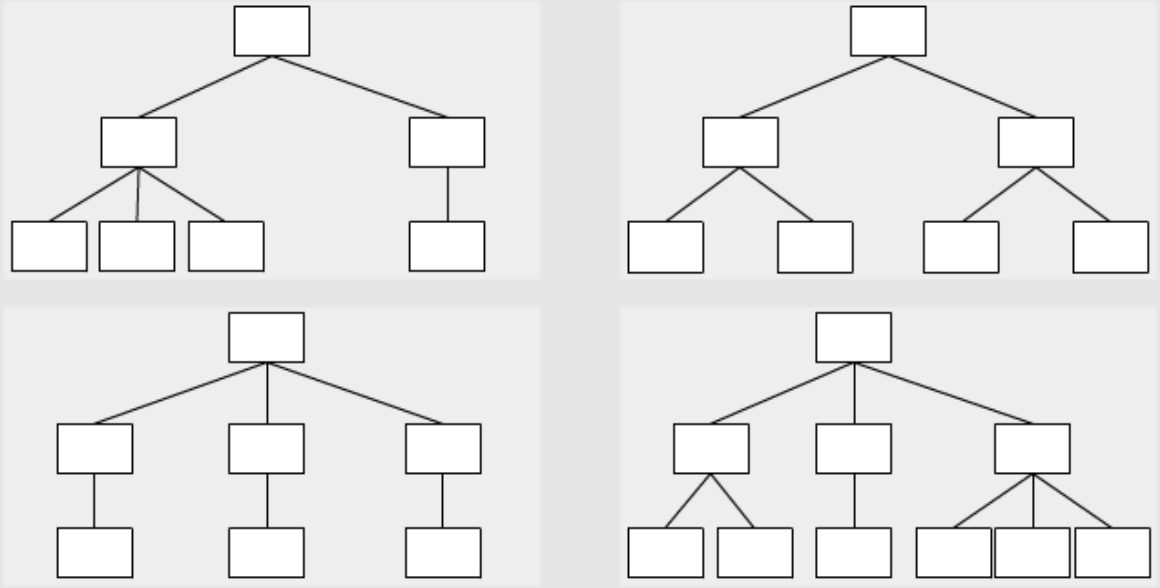
# La famiglia Pre ( max 8 punti )

Nella famiglia Pre vige una curiosa regola dei nomi: il nonno ha stabilito che a ogni nuovo nato della sua famiglia venga imposto il nome dell'ultimo nato con l'aggiunta, in fondo, di una nuova lettera. La famiglia festeggia la nascita dell'ultimo nato, Filippo, ed è particolarmente felice perché il nome del padre di ciascuno è lungo la metà (arrotondata per difetto quando il nome ha un numero dispari di lettere) del nome del figlio. Per esempio, il papà di Filippo si chiama Fil.

Che forma ha l'albero genealogico dei maschi della famiglia Pre? (3 punti)

Completa l'albero genealogico inserendo un nome in ogni casella. (5 punti)

I figli devono essere ordinati in modo che il maggiore sia a sinistra.



# Uno strano gioco dell'oca ( max 8 punti )

Aldo ha inventato un nuovo gioco dell'oca da proporre ai suoi amici. Invece di tirare i dadi, ad ogni turno ciascuno deve far avanzare la propria pedina secondo le regole seguenti:

1. se la casella numerata su cui si trova la pedina ha un numero pari, si può scegliere tra due mosse:
   Mossa A: andare alla casella successiva;
   Mossa B: andare alla casella con numero doppio di quella attuale (per esempio, se sono alla casella 6 vado alla casella 12);
2. se la casella su cui si trova la pedina ha un numero dispari si può fare soltanto la mossa B (per esempio, se sono alla casella 5 devo andare alla casella 10).

Chi supera l'ultima casella (il traguardo) deve ripartire dalla casella numero 1.

Quesito 1 (3 punti): Il gioco ha 27 caselle e si parte dalla casella numero 1. Qual è la sequenza di caselle che porta direttamente al traguardo?

Quesito 2 (5 punti): E quale sarebbe la sequenza di caselle se il gioco avesse 502 caselle?

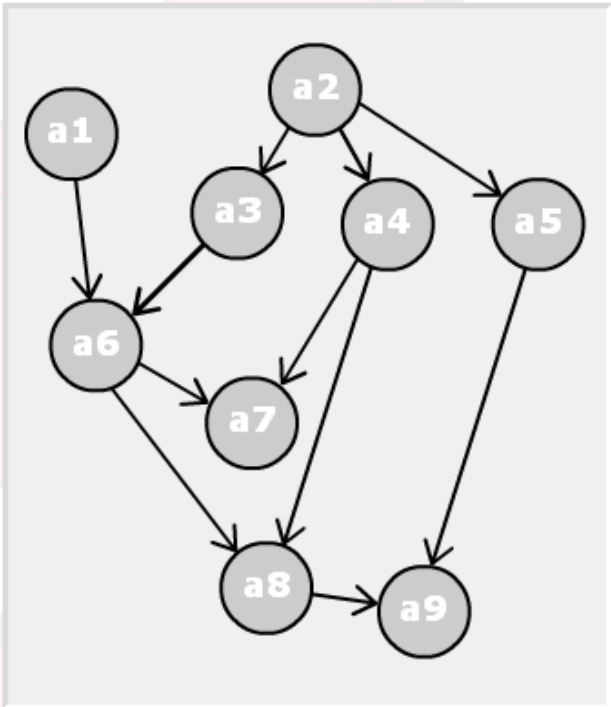
# Lavori di manutenzione ( 8 punti )

**Aldo, Bruno e Carlo stanno pianificando i lavori di manutenzione del loro "rifugio"... Come tecnici provetti, desiderano ultimare tutti i compiti nel più breve lasso di tempo. Così individuano ed elencano le diverse attività da svolgere, con le rispettive durate previste (in ore di lavoro) e le precedenze che devono essere rispettate.**

**Tralasciando qui le descrizioni dei lavori che costituiscono ciascuna attività, il loro piano dovrà rispettare questo schema:**

| Attività | Durata |
|---|---|
| a1 | 2 |
| a2 | 8 |
| a3 | 5 |
| a4 | 9 |
| a5 | 7 |
| a6 | 3 |
| a7 | 5 |
| a8 | 4 |
| a9 | 3 |

Il grafo disegnato qui a destra stabilisce le precedenze tra le attività: a1 e a2 possono cominciare subito; a2 è preliminare ad a3, a4 e a5, cioè queste potranno cominciare soltanto quando a2 sarà terminata; a6 potrà cominciare soltanto quando saranno terminate entrambe le attività a1 e a3, e così via. Ciascuna attività sarà svolta da uno solo dei tre ragazzi; anche se colui che si prende carico di una certa attività fosse aiutato da qualcuno dei suoi amici, la durata di quell'attività non cambierebbe...

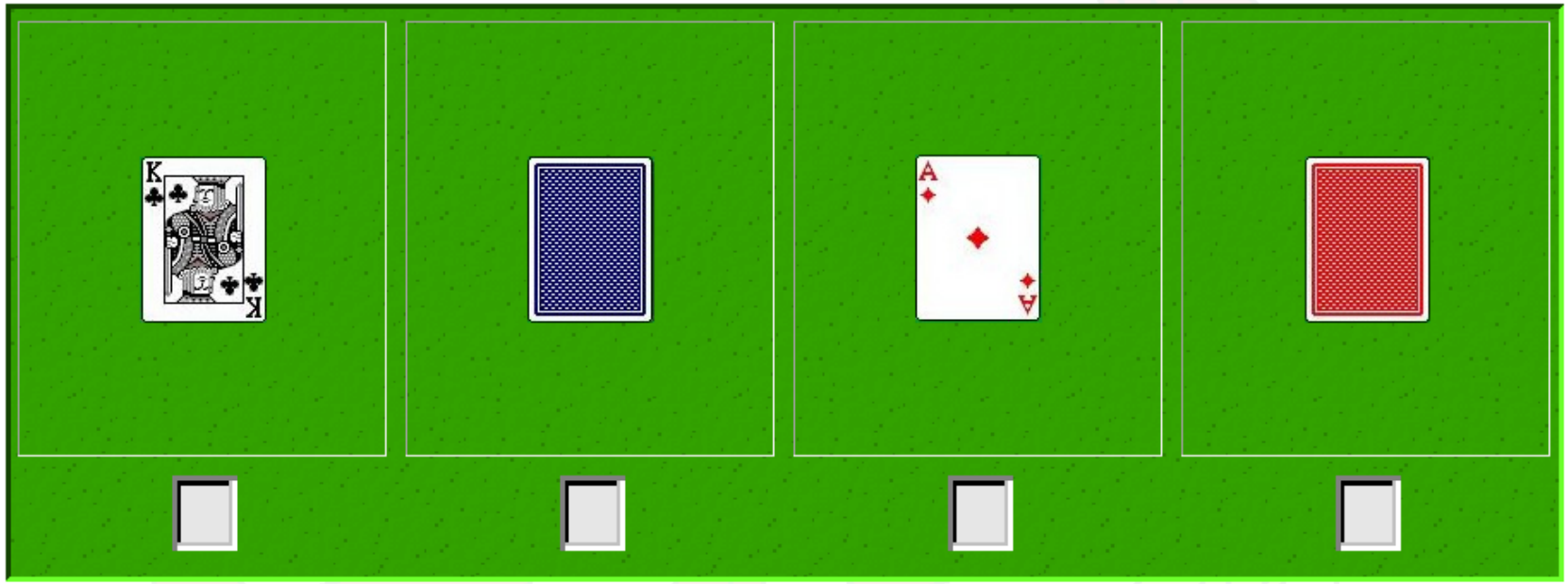



**In quante ore, come minimo, potranno essere completate tutte le attività previste?**

Nota: ai fini del punteggio conta solo la scelta del numero di ore; per aiutarti a trovare la soluzione, puoi associare un colore a ciascuno dei tre amici, quindi colorare le attività nel grafo (ad ogni clic cambierà il colore assegnato).

# Carte rosse e carte blu ( 6 punti )

Guarda le quattro carte sul tavolo: chissà se ogni carta col retro blu è un re...



Seleziona col mouse le carte che, girate, forniscono informazioni utili a stabilire con certezza se la seguente affermazione è vera: ogni carta col retro blu che si trova sul tavolo è un re.

# Acchiappa bug ( max 10 punti )

## Quesito 1 (4 punti)

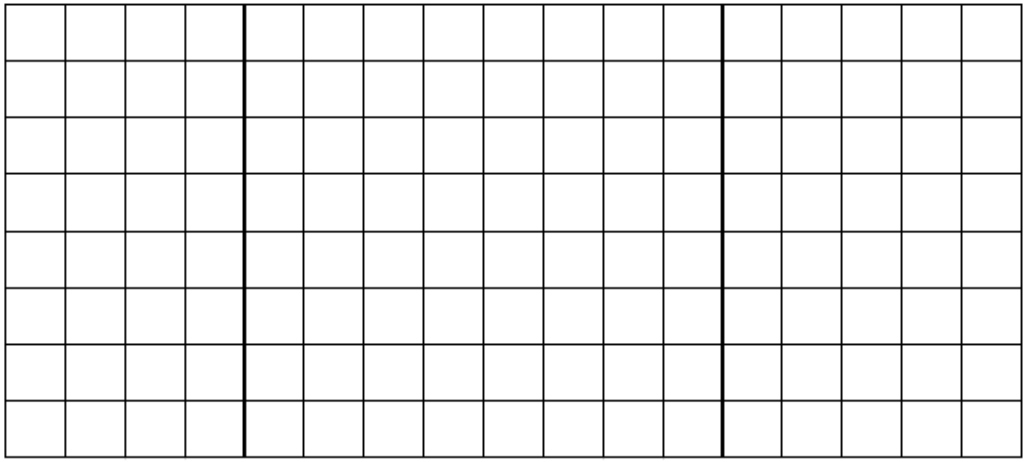
Il geometra Dueceli vuole sistemare i vasi di fiori nel suo cortile (rettangolo di base 17 e altezza 8 piastrelle, con i lati orientati secondo i quattro punti cardinali) in modo che formino un triangolo con la base lungo il lato Sud.

L'anno scorso aveva a disposizione 16 vasi di fiori e se l'è cavata con il procedimento mostrato a destra.

## Quesito 2 (6 punti)

1- Parte dalla piastrella d'angolo a Nord-Ovest
2- Chiama P il numero di piastrelle sul lato corto del cortile
3- Chiama V il numero di vasi a disposizione
4- Segna su un foglio il numero 1
5- Calcola 1/4 * V e lo chiama N
6- Ripete P volte:
   a. Calcola P - (valore sul foglio)
   b. Si sposta a Est del numero di piastrelle calcolato al punto precedente
   c. Calcola 2*[ (valore sul foglio) - (P - N) ] - 1 e lo chiama M
   d. Se M > 0, sistema M vasi, uno per piastrella, a partire da quella in cui si trova e procedendo verso Est
   e. Aumenta di uno il valore scritto sul foglio
   f. Se possibile, si sposta verso Sud di una piastrella e raggiunge la piastrella più a Ovest

Posiziona nella griglia i 16 vasi usando il procedimento indicato.



# Acchiappa bug ( max 10 punti )

**Quesito 1 (4 punti)** | **Quesito 2 (6 punti)**

Quest'anno ha 36 vasi e provando lo stesso procedimento si accorge che non funziona! Che cosa deve cambiare perché la sequenza di passi sia valida in entrambi i casi (ossia con 16 e 36 vasi)? E' consentito un solo cambiamento!

1- Parte dalla piastrella d'angolo a Nord-Ovest
2- Chiama P il numero di piastrelle sul lato corto del cortile
3- Chiama V il numero di vasi a disposizione
4- Segna su un foglio il numero 1

5- Calcola 1/4 * V e lo chiama N

6- Ripete P volte:

a. Calcola P - (valore sul foglio)

b. Si sposta a Est del numero di piastrelle calcolato al punto precedente

c. Calcola 2 * [ (valore sul foglio) - ( P - N ) ] - 1 e lo chiama M

d. Se M > 0 , sistema M vasi, uno per piastrella, a partire da quella in cui si trova e procedendo verso Est

e. Aumenta di uno il valore scritto sul foglio
f. Se possibile, si sposta verso Sud di una piastrella e raggiunge la piastrella più a Ovest

# Crucipuzzle informatico ( max 10 punti )

| T | H | L | S | O | F | T | W | A | R | E |
|---|---|---|---|---|---|---|---|---|---|---|
| A | A | T | E | C | I | C | P | U | F | I |
| S | R | N | U | T | A | E | E | O | I | N |
| T | D | S | L | R | I | N | M | L | L | T |
| I | W | I | O | R | I | T | D | A | E | E |
| E | A | G | O | R | I | N | C | I | G | R |
| R | R | M | L | R | G | S | G | E | R | N |
| E | E | E | O | N | A | E | P | B | Z | E |
| M | E | G | G | P | A | J | N | I | R | T |
| T | L | I | I | F | I | C | I | T | A | L |
| A | P | I | N | G | U | I | N | O | E | I |

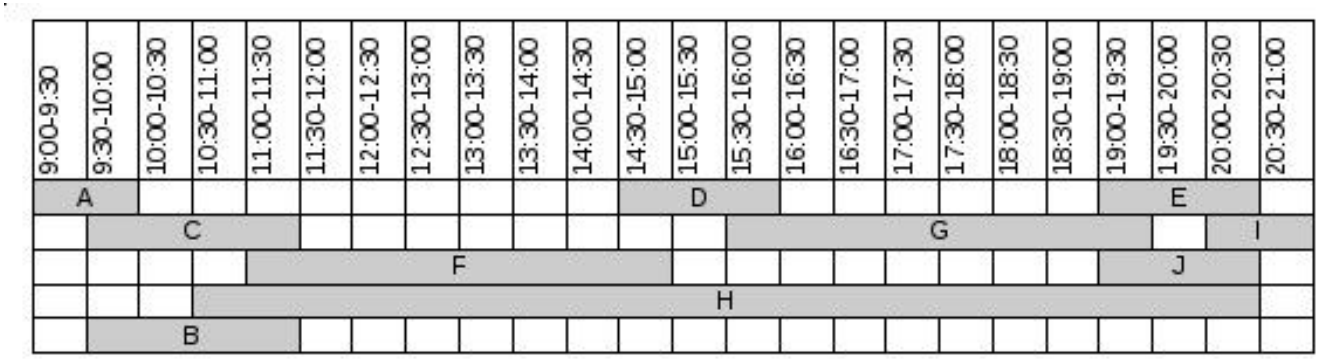
| Griglia e Definizioni | | Istruzioni |
|---|---|---|
| 1- | | procedimento, specificato rigorosamente e senza ambiguità, per la risoluzione di un problema |
| 2- | | cifra binaria |
| 3- | | acronimo inglese della "unità centrale di elaborazione" di un computer |
| 4- | | unità, costituita da una sequenza di byte e individuata da un nome, secondo cui sono organizzati i dati su un hard disk o altro dispositivo di memoria permanente |
| 5- | | insieme delle parti fisiche di un computer |
| 6- | | la rete delle reti |
| 7- | | un formato standard per rappresentare immagini |
| 8- | | procedura di accesso ad un sistema o un'applicazione informatica, in cui a volte si richiede anche l'inserimento di una password |
| 9- | | Io sono ad esempio la RAM, l'hard disk, una chiavetta USB |
| 10- | | linguaggio di programmazione così chiamato in onore di un grande matematico e filosofo francese |
| 11- | | mascotte del sistema operativo libero Linux |
| 12- | | acquisire un'immagine usando uno scanner |
| 13- | | la parte che non si tocca dell'informatica... |
| 14- | | si chiama così il codice di un programma formato da istruzioni appartenenti ad un determinato linguaggio di programmazione |
| 15- | | ce ne sono QWERTY e DVORAK |
| 16- | | matematico inglese, considerato uno dei padri dell'informatica, al quale è dedicato un premio internazionale |

# Soluzioni dei quesiti



# Soluzione del quesito "Comic-con 2010"

**Soluzione.** Associando agli eventi le lettere dalla A alla J (in ordine), si può rappresentare il programma della fiera con uno schema come il seguente:




Alcune strategie consentono di partecipare al massimo a 3 eventi che non si sovrappongano:

- se si scelgono *gli eventi che iniziano per primi*, si sceglieranno A e H;
- se si scelgono *gli eventi piú brevi*, si sceglieranno A, I e D;
- se si scelgono *gli eventi che hanno meno sovrapposizioni*, si sceglieranno A (che si sovrappone solo ai due eventi B e C), D (che si sovrappone a F, G e H) e ancora I (che si sovrappone a E, H e J).

La strategia vincente è la restante, quella che sceglie *gli eventi che finiscono prima*, che consente di partecipare a quattro eventi: A, F, G e I.

**Difficoltà e errori frequenti.** Per questa prova era possibile tentare le quattro strategie diverse e verificare quale fosse la migliore. Come si sarebbe dovuto ragionare in generale? Volendo scegliere il massimo numero di

eventi che non si sovrappongano, una buona idea sembra essere quella di lasciare agli eventi ancora da scegliere, dopo aver effettuato una scelta precedente, la massima quantità possibile di tempo. Per questo la strategia giusta è quella di scegliere gli eventi che *finiscono prima*: lasciano il massimo spazio (di tempo) dopo!

**Contesto informatico e riferimenti.** Una strategia che a ogni passo sceglie l'elemento *migliore* per costruire una soluzione parziale, senza provare tutti i casi possibili, si dice *strategia ingorda* (o *golosa* o *miope*, in inglese *greedy*). L'efficacia di una strategia ingorda dipende da cosa si definisce come "migliore". Non sempre è possibile arrivare alla *soluzione ottima* effettuando le scelte che *sembrano* migliori in un dato momento! Per esempio, la strategia di scegliere gli eventi piú brevi poteva essere plausibile, ma non era efficace. Invece, si può dimostrare che scegliere ad ogni passo l'evento che finisce prima (senza sovrapposizioni) garantisce di trovare una *soluzione ottima*, quale che sia l'insieme degli eventi di partenza e la loro collocazione temporale.

Un altro esempio di strategia ingorda che risulta inefficace (in generale) è il seguente: quando dobbiamo pagare esattamente un determinato importo, di solito scegliamo le monete di valore maggiore che non superino la somma da pagare: $73 = 50 + 20 + 2 + 1$, e in questo modo *minimizziamo* il numero di monete da usare. Ma se avessimo a disposizione solo monete di valori 50, 40, 30, 10, 2 e 1, ecco che $73 = 50 + 10 + 10 + 2 + 1$ non sarebbe la soluzione col *minimo numero di monete*, perché $40 + 30 + 2 + 1$ userebbe una moneta in meno. Nel caso delle monete, la strategia "*prima il valore maggiore*" funziona per certi tagli (valori) delle monete, quelli in particolare comunemente disponibili nei vari paesi, ma non funziona per altri.

È possibile stabilire, in generale, quando una strategia ingorda trova una soluzione ottima e quando no, ma anche nei casi in cui l'ottimo non è garantito ci si può eventualmente accontentare di una soluzione non ottima, specie quando trovare una soluzione davvero ottima sia praticamente impossibile o comunque troppo oneroso in termini di tempo.

**Parole chiave:** algoritmi ingordi o *greedy*, problemi di ottimizzazione.



# Soluzione del quesito “Gessetti colorati”

**Soluzione.** Poiché nella scatola ci sono inizialmente cinque gessetti rossi e soltanto uno verde, la sequenza in figura risulta impossibile. Infatti l’ingresso di un componente della famiglia Rossi *consuma* dalla scatola un gessetto rosso e ne aggiunge uno verde. Viceversa, l’ingresso di un componente della famiglia Verdi consuma un gessetto verde e ne aggiunge uno rosso. La sequenza indicata è perciò impossibile, perché implica che nella scatola ci siano stati tre gessetti verdi, ma all’inizio ce n’era uno solo e il primo Rossi che è entrato può averne aggiunto solo un altro.

**Difficoltà e errori frequenti.** Capiti i meccanismi secondo i quali cresce e diminuisce il numero di gessetti nella scatola di un dato colore, il quesito non dovrebbe porre particolari difficoltà. Il sistema delineato è tipicamente *non deterministico,* in quanto non esiste un’unica evoluzione possibile: per esempio, nello scenario delineato nel testo, non sappiamo prevedere se il primo segno sulla lavagna sarà rosso o verde, anche se è rosso in tutte le soluzioni proposte.

**Contesto informatico e riferimenti.** Le attività che risultano dall’esecuzione di un programma prendono il nome di *processi*: ciascuno dei componenti delle due famiglie Rossi e Verdi costituisce un processo, che evolve secondo le regole stabilite dalla rispettiva appartenenza familiare. Naturalmente è possibile che vi siano piú processi in corso di esecuzione e che essi si trovino a dover condividere alcune risorse: si parla in questo caso di *processi concorrenti* che si contendono una risorsa, il cui uso va perciò regolato. Nel quesito la lavagna è contesa fra i signori Rossi e Verdi, che aderiscono ad un *protocollo di sincronizzazione* per evitare problemi. Il meccanismo dei gessetti ricalca quello dei *semafori.* Inventato da E. W. Dijkstra, il semaforo[1] consiste in una variabile condivisa contenente un numero intero. Dopo essere stata inizializzata, tale variabile può essere

[1] Il termine fa riferimento alle segnalazioni semaforiche utilizzate nelle ferrovie (*semaphore*) e non ai semafori utilizzati negli incroci stradali (*traffic light*). Attenzione a non confondere il rosso e verde che contraddistinguono le famiglie con i colori in uso nei semafori stradali!

manipolata solo tramite due operazioni che, per facilitare la comprensione, chiameremo `alza` e `abbassa-e-passa`. L'operazione `abbassa-e-passa` controlla se il semaforo è $> 0$: se lo è la variabile viene diminuita di un'unità (e l'attività può proseguire), altrimenti il processo che ha iniziato l'esecuzione della `abbassa-e-passa` viene sospeso e riprenderà solo quando la variabile assumerà di nuovo valori positivi (il che è possibile proprio grazie al fatto che essa è condivisa). L'operazione `alza` incrementa il semaforo ed eventualmente *risveglia* un processo sospeso da una `abbassa-e-passa`. I semafori possono essere usati per regolare l'accesso ad una risorsa condivisa. Si immagini di avere due stampanti e dieci processi concorrenti che hanno necessità di usarle. Come fare per garantire che non avvenga mai che piú di due processi finiscano per usare contemporaneamente le stampanti? Usiamo un semaforo, che inizialmente poniamo uguale a 2 e stabiliamo che ciascun processo inizi con una `abbassa-e-passa` e termini con una `alza`.

Nel quesito la sincronizzazione è un pochino piú complicata, perché i semafori coinvolti sono due: infatti l'ingresso di un componente della famiglia Rossi corrisponde ad una `abbassa-e-passa` su un semaforo ("rosso"), mentre l'uscita corrisponde ad una `alza` su un altro semaforo ("verde").

I semafori permettono di sincronizzare attività parallele senza predeterminare esattamente la sequenza di tutte le operazioni: i singoli processi possono quindi essere programmati in maniera abbastanza indipendente, almeno per quanto riguarda le operazioni non semaforiche. Queste ultime però vanno coordinate con molta attenzione, perché il rischio è quello di creare situazioni in cui nessun processo può procedere oltre perché tutti aspettano che qualcun altro faccia una `alza`, che nessuno può fare proprio perché sospeso: il cosiddetto *deadlock,* altrimenti noto ai fan del ragionier Fantozzi come *ingorgo a croce uncinata!*

**Parole chiave:** sincronizzazione, processi concorrenti, semafori, deadlock.



# Soluzione del quesito “La data della Pasqua”

**Soluzione.** Nell’anno 2021 la Pasqua cadrà il 4 aprile — proprio come quest’anno!

Esplicitando i calcoli fatti passo dopo passo dall’algoritmo, si ha infatti:

$$\begin{aligned} A &= 7 \\ B &= 1 \\ C &= 5 \\ D &= 7 \\ E &= 6 \\ F &= 13 \end{aligned}$$

Provate a cercare qualcuno degli anni per i quali il calcolo termina in uno dei casi particolari previsti ai passi 6 e 7.

**Difficoltà e errori frequenti.** L’interpretazione dell’algoritmo presentato e la successione dei calcoli non dovrebbero comportare alcuna difficoltà: per far presto, bisogna soltanto avere dimestichezza con l’operazione che produce il resto di una divisione tra interi, detta *modulo*. I risultati parziali, man mano che sono trovati, vengono memorizzati in diverse *variabili*, denotate dalle lettere $A$, $B$, …, $F$, per poterli usare ai passi successivi. Osservate che il termine *variabile* in informatica ha un significato molto diverso da quello comune in matematica: indica un elemento di *memoria* in cui viene conservato un valore che può variare nel tempo; un po’ come un foglio di carta su cui è possibile scrivere in matita un valore…

**Contesto informatico e riferimenti.** Oltre a istruzioni che *assegnano* un valore a una variabile o che *ritornano* il risultato e *terminano*, in questo algoritmo sono presenti delle istruzioni *condizionali*, della forma "se ... allora ... altrimenti ..." oppure piú semplicemente "se ... allora ...".

L'*aritmetica modulare* compare spesso nella vita di tutti i giorni — si pensi proprio ai calendari e ad altri strumenti per misurare il tempo — e riveste una notevole importanza pure in alcune componenti di un computer, sia software sia hardware!

Quando si dice, ad esempio, che 31 è uguale a 23 *modulo* 4, si intende affermare che la divisione tra 31 e 4 e la divisione tra 23 e 4 producono lo stesso *resto* (in questo caso 3, mentre i *quozienti* sono rispettivamente 7 e 5); i resti possibili della divisione per 4 sono quattro: 0, 1, 2, 3, e pertanto i numeri naturali si posso ripartire in quattro *classi di equivalenza*: due numeri stanno nella stessa classe se e soltanto se la loro divisione per 4 produce lo stesso resto.

In un orologio meccanico vige l'aritmetica modulo 12: infatti, i numeri delle ore vanno da 1 a 12; se adesso sono le 8, del mattino o della sera, tra 12 ore le lancette si troveranno di nuovo nella stessa posizione, e cosí tra 24 ore eccetera. Se, al posto del 12, sul quadrante ci fosse lo zero sarebbe lo stesso: in effetti, 12 è proprio uguale a 0 modulo 12.

Con quattro *bit* si può contare da 0 (0000) a 15 (1111); se arrivati a 15 si aggiunge 1, si dovrebbe ottenere 10000, ma siccome i bit sono soltanto quattro... si riparte da zero!

**Parole chiave:** istruzioni (di assegnazione, condizionali, di ritorno), aritmetica modulare.

# Soluzione del quesito “Una partita a tris”

**Soluzione.** La mossa giusta per Sir Cross è:

| o | x | |
|---|---|---|
| | | |
| x | | |

Infatti a questo punto Lady Circle non ha alcuna mossa vincente, e pertanto Sir Cross si assicura almeno il pareggio — se giocherà bene, s’intende! Gli altri due casi vanno esclusi perché c’è sempre almeno una mossa che consentirebbe a Lady Circle di vincere.

Nel caso illustrato nella figura seguente (a sinistra), la mossa vincente di Lady Circle è indifferentemente una delle due riportate al centro e a destra della stessa figura:

| o | x | x |
|---|---|---|
| | | |
| | | |

| o | x | x |
|---|---|---|
| o | | |
| | | |

| o | x | x |
|---|---|---|
| | | |
| o | | |

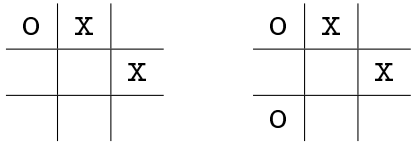
mentre nel caso della figura seguente (a sinistra), l’unica possibilità di vittoria per Lady Circle è la mossa illustrata a destra:

| o | x | |
|---|---|---|
| | | |
| | x | |

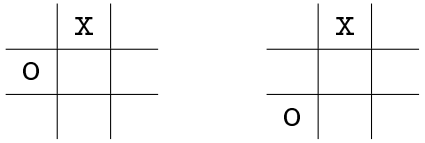
| o | x | |
|---|---|---|
| | o | |
| | x | |

Per inciso, oltre alle tre proposte nel quesito, Sir Cross aveva altre quattro possibili mosse (in questa situazione non c’è alcuna simmetria!), una soltanto delle quali gli sarebbe stata fatale, e precisamente quella della figura

seguente (a sinistra), alla quale Lady Circle, per assicurarsi la vittoria, avrebbe potuto rispondere come nella figura a destra:




**Difficoltà e errori frequenti.** Senza alcun dubbio il tris o "filetto" è universalmente conosciuto e ancora praticato da tanti ragazzi, sebbene sia altrettanto noto che si tratta di un gioco "alla pari": nessuno dei due giocatori riesce a vincere se l'avversario non commette errori! Occorre soltanto fare un po' di attenzione proprio nella fase iniziale. Ad esempio, se alla sua prima mossa Lady Circle avesse risposto in uno di questi due modi:




ovvero in uno dei due simmetrici, allora Sir Cross avrebbe facilmente vinto...

**Contesto informatico e riferimenti.** Questo gioco è stato oggetto non soltanto di articoli e libri interi ma, ovviamente, anche di analisi al calcolatore e persino del primo videogame, in grado di non perdere mai, progettato nel 1952 per il computer EDSAC. In verità, già un secolo avanti, Charles Babbage — che fu il primo a concepire l'idea di calcolatore *programmabile* — aveva progettato sulla carta un automa capace di giocare a *tit-tat-to* (questo il nome da lui usato per indicare il gioco; tra gli altri tuttora diffusi: *tic-tac-toe* o *ticktacktoe* o, con

riferimento ai simboli grafici, *noughts and crosses*, che il grande poeta romantico William Wordsworth chiama *crosses* e *cyphers* in un suo preludio sui ricordi di scuola).

Contando come una sola tutte le posizioni che si possono ottenere l'una dall'altra per qualche simmetria (rotazioni e/o ribaltamenti), quelle possibili sono 765 e le partite 26830: ben poca cosa per un computer, decisamente piú arduo compito un'elaborazione manuale. Come si dovrebbe procedere? Per costruire l'*albero di gioco* completo partiamo dall'alto, mettendo in *radice* lo *stato iniziale* (lo schema vuoto, ove inizierà chi mette la croce), e poi procediamo verso il basso, facendo in modo che ciascun nodo abbia come *figli* tutti gli stati che si possono originare con una mossa del giocatore al quale tocca muovere nello stato rappresentato da quel nodo. Ad esempio, i figli della radice saranno tre, poiché la croce può essere messa in sole tre posizioni essenzialmente differenti (al centro, o in uno degli angoli, o in mezzo a uno dei lati). Gli stati *finali* (in cui uno dei due giocatori ha fatto tris o tutte e nove le mosse sono state fatte) saranno le *foglie* dell'albero. E proprio dalle foglie dovremo partire, man mano risalendo, per analizzare completamente (o, come si dice, *risolvere*) il gioco, come tra poco vedremo.

Prima, però, una precisazione è doverosa. Abbiamo parlato di posizioni o stati, ossia in generale configurazioni del tavoliere di gioco (in cui va compresa l'informazione binaria che dice "a chi tocca muovere", se non è già implicita). Se pensiamo a tutti questi stati collegati da archi, in modo tale che vi sia un arco dallo stato $x$ allo stato $y$ se e soltanto se con una mossa si può passare da $x$ a $y$, allora in realtà non stiamo pensando a un albero ma piú in generale a un *grafo* (orientato). Se questo grafo avesse dei cicli, allora il gioco potrebbe continuare all'infinito... Tuttavia, è piú semplice e conveniente pensare a un albero, in cui piú nodi distinti possono rappresentare la stessa posizione, se a questa si può arrivare mediante diverse sequenze di mosse a partire dallo stato iniziale.

L'analisi del gioco si basa su questo ragionamento: se in un certo stato tocca a me muovere, cercherò la mossa che mi farà guadagnare il massimo possibile; se invece tocca al mio avversario, cercherò quella che farà guadagnare a lui il massimo possibile: suppongo infatti che pure lui giochi, come me, al meglio delle proprie possibilità. Parto dunque dalle foglie dell'albero, attribuisco a ciascuna di esse un punteggio: $-1$ se perdo, 0 se la partita è patta, 1 se vinco. Poi risalgo via via fino alla radice, attribuendo a ciascun nodo (di cui ho già valutato tutti i figli) il

*massimo* punteggio dei figli se in quel nodo tocca a me muovere, il *minimo* se tocca al mio avversario. Quando arrivo a valutare la radice, il gioco è risolto: se il suo punteggio è 1 io ho una strategia vincente, se è $-1$ ce l'ha il mio avversario, se è 0 la partita non potrà che finire in parità, se nessuno di noi due commetterà errori.

Si intuisce un'ipotesi, implicitamente fatta in questo discorso: che il gioco sia *a somma nulla*, vale a dire che uno svantaggio di uno dei due giocatori equivalga a un vantaggio di pari entità dell'altro giocatore; quindi il minimo guadagno di uno corrisponde al massimo dell'altro. L'utilità di tale ipotesi si apprezza ancor meglio se l'esplorazione dell'albero di gioco arriva soltanto fino a una certa profondità, a partire dallo stato attuale. Ciascuno dei due giocatori cercherà allora di *minimizzare* la propria *massima* perdita possibile (da cui il nome dell'algoritmo: *minimax*): questa idea sta alla base di un teorema formulato nel 1928 da John von Neumann, uno dei padri dei moderni calcolatori elettronici, che tra il 1926 e il 1944 diede importanti contributi alla teoria economica e alla nascente *teoria dei giochi*.

Quali sono le altre caratteristiche salienti di questo tipo di giochi, in cui ad esempio rientrano anche la dama e gli scacchi?

- I giocatori sono due e muovono a turno, *alternandosi*;
- il gioco è *finito*: ossia, l'albero di gioco è finito, sia in ampiezza (in ogni stato il numero di mosse lecite è limitato), sia in altezza (prima o poi la partita termina);
- il gioco è *a informazione perfetta*: ossia, in ogni momento, in particolare prima della scelta della mossa, entrambi i giocatori conoscono lo stato completo del gioco — nulla è tenuto nascosto o lasciato al caso.

In queste ipotesi il gioco si dice *strettamente determinato*: o uno dei due giocatori ha almeno una strategia che assicura la vittoria (cioè: ad ogni suo turno ha almeno una mossa che può portarlo poi a vincere, qualunque strategia adotti l'avversario), o entrambi hanno almeno una strategia che assicura il pareggio. In effetti, il procedimento che abbiamo delineato permette di decidere quale si verifica dei tre casi possibili già previsti da Ernst



Zermelo (celebre per i suoi contributi alla teoria assiomatica degli insiemi) nel 1912: salvo errori, vince chi inizia o chi risponde o è patta teorica.

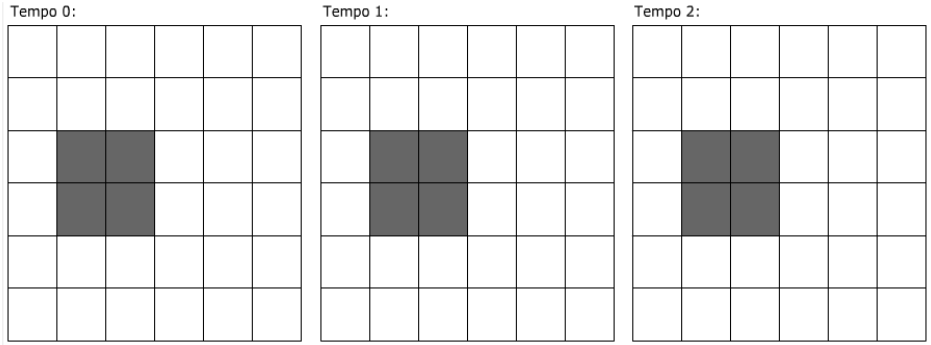
C'è un "però", come facilmente si intuisce da quanto accennato... L'albero di gioco può essere gigantesco, perfino in modo inimmaginabile: si pensi a Othello o agli scacchi, dove il numero di posizioni diverse pare abbia 29 e 47 cifre decimali, rispettivamente — dunque enormemente piú grande di 765 — figuriamoci il numero di partite che possono essere giocate! Come può agire allora un programma che abbia la pretesa di combattere contro un avversario? Anziché arrivare alle foglie, esplora l'albero dalla posizione attuale fino a una certa profondità, cercando di attribuire in modo efficace (e questo è un aspetto critico) un punteggio agli stati raggiunti, senza proseguire oltre (salvo casi particolari). In piú, tenta di "potare" l'albero, evitando di scendere lungo rami che non possono migliorare la valutazione finora fatta della posizione attuale. Non sempre è possibile assegnare un punteggio a uno stato che non sia finale, rinunciando ad indagare avanti: si pensi proprio al "filetto"; in alcuni casi può essere invece relativamente semplice, quando si riesce a quantificare bene il vantaggio di un giocatore in base al materiale o alla posizione. Si tratta comunque di una valutazione ispirata da criteri euristici.

Un'ultima curiosità: la dama giocata sulla tradizionale scacchiera $8\times8$ è stata risolta — a favore della parità — nel 2007, dopo 18 anni di calcoli da parte di qualche centinaio di computer. (Si tratta del piú grande gioco per cui sia stato trovato un risultato di questo tipo: il numero di posizioni ha 21 cifre... quantomeno decine di milioni di volte piú piccolo di quello stimato per Othello!) Ma questo fatto, ovviamente, non toglie nulla all'emozione di una partita a dama!

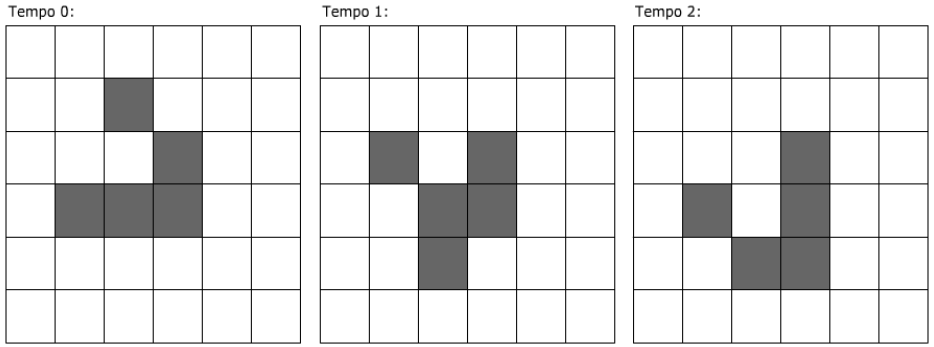
**Parole chiave:** teoria dei giochi, gioco strettamente determinato, albero di gioco, algoritmo *minimax*.

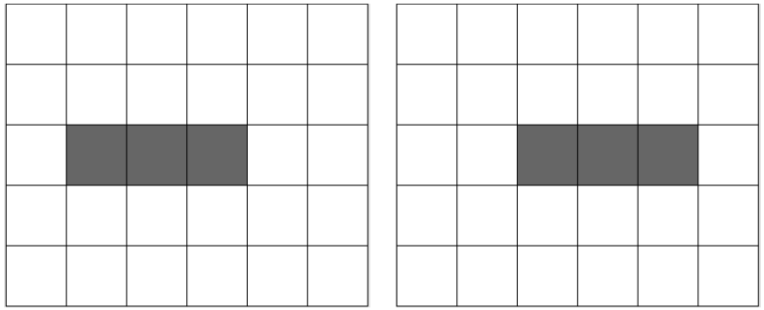
# Soluzione del quesito “Salvaschermo lampeggiante”

**Soluzione per la categoria “Medie”.** Per il display 1 la risposta corretta è quella illustrata a lato. All’inizio infatti ciascuno dei pixel accesi ha tre pixel vicini accesi, quindi la prima regola non si applica e i pixel accesi restano accesi. Neppure la seconda regola si applica, perché nessun pixel spento ha esattamente tre pixel vicini accesi. Pertanto, al secondo successivo (tempo 1, se si parte da 0), la figura rimane invariata — e cosí accadrà ovviamente anche per tutti i secondi a venire, poiché lo *stato* (acceso o spento) dei pixel dipende soltanto dallo stato al secondo precedente! Si può dire che la prima *configurazione* proposta è periodica di periodo una unità di tempo (nel nostro caso, secondi): in pratica, non cambia mai, è *stabile*.



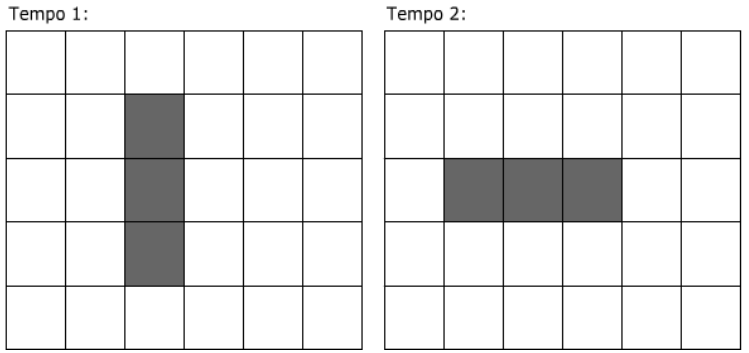

Nel caso del display 2 si ottengono invece le configurazioni illustrate a destra. Ad esempio, il pixel che si trova nella seconda riga e terza colonna (acceso al tempo 0) si spegnerà al tempo 1 poiché ha soltanto un vicino acceso (prima regola), mentre il pixel nella terza riga e seconda colonna (spento al tempo 0) si accenderà al tempo 1 perché ha esattamente tre vicini accesi (seconda regola). E cosí via...





**Soluzione per la categoria "Biennio".** Le soluzioni possibili sono due, come illustrato in figura. Bisogna quindi aggiungere un quadratino nero affiancandolo ai due già presenti, o a destra o a sinistra. In tal modo, dopo un secondo (cioè al tempo 1, se si parte da 0) si avranno tre pixel accesi in verticale: infatti, i due laterali dei tre accesi si spegneranno per la prima regola (uno solo dei vicini è acceso), quello centrale resterà acceso (ha due vicini accesi, quindi la prima regola non si applica) e il pixel sopra di lui e quello sotto di lui si accenderanno per la seconda regola (sono spenti e hanno esattamente tre vicini accesi). Ovviamente, per simmetria, dopo un altro secondo (cioè al tempo 2) si avrà di nuovo la *configurazione iniziale* (cioè quella al tempo 0).

Ecco le configurazioni al tempo 1 e al tempo 2, quando si aggiunge un quadratino nero a sinistra dei due già presenti: si ottiene cosí una configurazione oscillante, *periodica* di periodo due unità di tempo (nel nostro esempio secondi) o "bistabile", nota come *blinker* (lampeggiatore).




Vediamo perché è l'unica soluzione. Per simmetria, si possono considerare soltanto altre tre possibilità per il posizionamento di un quadratino nero (pixel acceso):

- se si mette subito al di sopra (o al di sotto) di uno dei due già presenti, ciascuno dei tre pixel accesi resterà acceso al tempo 1 (ha due vicini accesi), mentre si accenderà un solo pixel ora spento: quello che con i tre accesi forma un quadrato, e cosí si otterrebbe una configurazione stabile, che non cambia piú (vedi il display 1 nell'analogo quesito per la categoria "Medie");
- se si dispone diagonalmente rispetto a uno dei due già presenti, soltanto quello centrale dei tre rimarrà acceso

al tempo 1 e si accenderà però un solo altro pixel ora spento (l'unico che al tempo 0 ha intorno i tre accesi), sicché al tempo 2 si spegnerebbero entrambi per la prima regola e nessuno si accenderebbe piú;
- in qualsiasi altro caso, il quadratino nero aggiunto rimane isolato dai due già presenti e al tempo 1 si spegnerebbero tutti, senza che nessun altro si accenda.

**Difficoltà e errori frequenti.** Non vi sono particolari difficoltà. Bisogna soltanto rimanere concentrati per non rischiare di far confusione tra il tempo attuale ($i$) e quello subito successivo ($i+1$): per decidere se un pixel sarà acceso o spento al tempo $i+1$ dobbiamo guardare esclusivamente lo stato dei suoi vicini al tempo $i$ (contano quelli accesi), senza farci influenzare dal loro nuovo stato, se lo abbiamo appena calcolato!

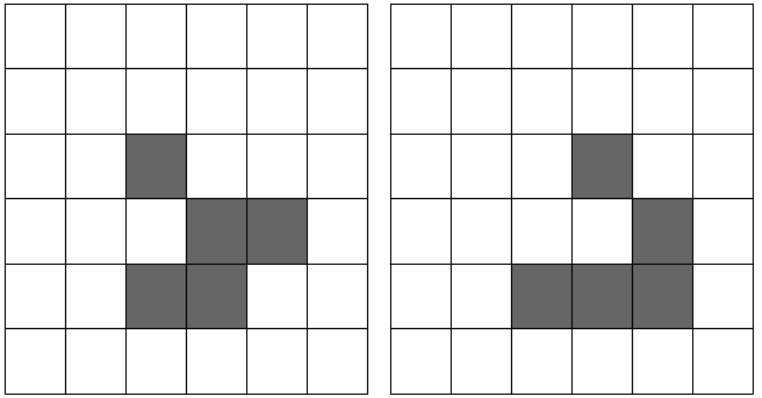
**Contesto informatico e riferimenti.** Le regole date nel testo del quesito costituiscono l'esempio piú famoso di *automa cellulare*. Si tratta del cosiddetto *Game of Life* (Gioco della Vita) o semplicemente *Life*, inventato dal matematico inglese John Horton Conway verso la fine degli anni '60.

Un automa cellulare è formato da un insieme di *cellule* (in numero finito e, in generale, di vari tipi e variamente connesse) che interagiscono sulla base di certe *regole*. Stabilita una *configurazione iniziale*, si può soltanto osservare come questo sistema evolve, senza piú intervenire: può dunque considerarsi un "gioco" senza alcun giocatore, se si escludono chi detta le regole e chi escogita la situazione di partenza! I piú semplici automi cellulari vivono in un mondo a una sola dimensione: la popolazione di cellule evolve su un "nastro", anziché su uno schermo bidimensionale. I primi furono ideati da Stanislaw Ulam e John von Neumann all'inizio degli anni '50 e da allora, specialmente grazie all'impiego dei computer, sono stati studiati e utilizzati in diversi campi, soprattutto per modellare e simulare vari fenomeni naturali.

Ritorniamo un attimo alla configurazione presentata nel display 2, per osservare che, se si va avanti di altri due secondi, al tempo 4 si ottiene la stessa figura di partenza (ossia quella al tempo 0) ma traslata di una riga verso



il basso e di una colonna verso destra: cosí, se lo schermo è sufficientemente esteso, si può vedere questa figurina, costituita da cinque pixel accesi, man mano che si sposta in direzione sud-est!



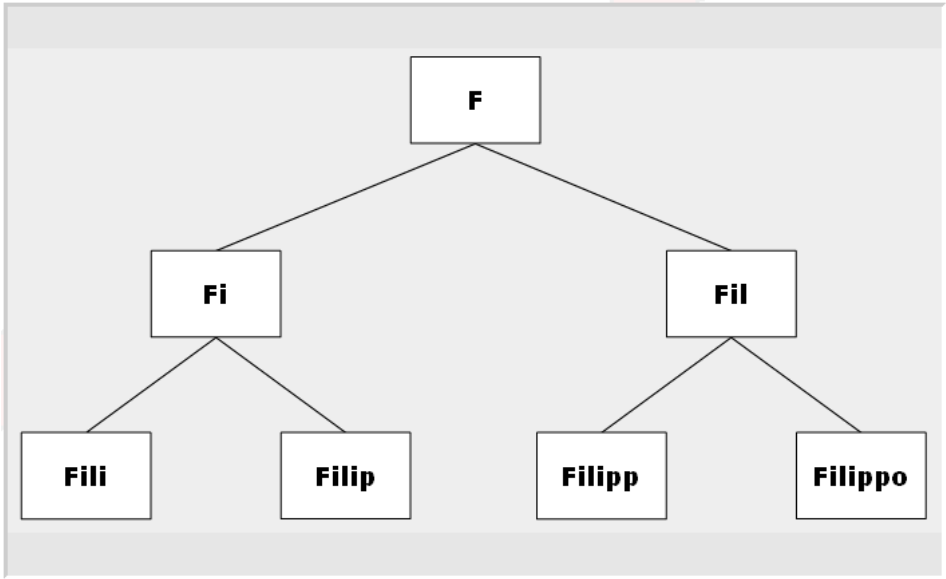
In effetti essa è nota col nome inglese *glider* (aliante), ed è la piú semplice *spaceship* (navicella), come sono dette quelle configurazioni in grado di traslare sul piano; si può dire che ha un periodo pari a quattro unità di tempo, poiché assume quattro differenti disposizioni durante la traslazione.

*Life* è stato pure impiegato per risolvere problemi matematici ed è particolarmente interessante anche dal punto di vista dell'informatica teorica: è stato infatti dimostrato che ha le potenzialità di un *computer universale*. Fu divulgato col numero di ottobre del 1970 della rivista mensile *Scientific American* (pubblicata in Italia come *Le Scienze*) da Martin Gardner, che per un quarto di secolo ne curò la rubrica di giochi matematici. Già in quello stesso anno furono trovate molte interessanti e affascinanti configurazioni, e non soltanto periodiche, oscillanti o navicelle, anche di lunghissimo periodo: alcune possono scomparire o stabilizzarsi, magari dopo molto tempo; altre crescono indefinitamente, come il "cannone" di Gosper, che "spara" periodicamente alianti che via via si allontanano... Vi consigliamo di consultare Wikipedia (l'edizione inglese è la piú ricca) e il vostro divertimento sarà assicurato!

**Parole chiave:** automa cellulare, tempo discreto, regole di transizione, configurazione iniziale, periodo.

# Soluzione del quesito “La famiglia Pre”

**Soluzione.** La soluzione corretta è evidenziata nella figura. Poiché il nome del padre di ogni nato nella famiglia è lungo la metà di quello di un figlio, ogni padre può avere *al massimo due figli*: se per esempio, come nel nostro caso, il nome di un figlio è di 7 lettere, il padre avrà un nome di 3 lettere, e potrà avere un altro figlio con un nome di 6 lettere, ma non altri.




**Difficoltà e errori frequenti.** Il quesito per la categoria “Medie” è risultato nettamente piú facile di quello della categoria “Biennio”. Nel primo caso, la forma dell’albero è già assegnata e questo evidentemente aiuta i giocatori ad impostare il ragionamento che porta al corretto inserimento dei nomi nell’albero. Nel secondo caso, invece, bisogna innanzitutto individuare la forma corretta dell’albero. L’osservazione chiave è quella che stabilisce che un padre non può avere piú di due figli e quindi vanno esclusi gli alberi in cui ci sono tre fratelli.

**Contesto informatico e riferimenti.** In informatica, il termine *albero* indica una *struttura di dati*, ossia un modo per organizzare delle informazioni. Gli elementi di questa struttura sono organizzati in modo gerarchico, come in un *albero genealogico*, dove la gerarchia è definita dalla relazione *padre-figlio*. Nel nostro caso, l’albero si dice *binario*, perché ogni padre ha al massimo due figli e inoltre vi è una chiara distinzione tra figlio *destro* e

figlio *sinistro*. Se non avessimo precisato nel testo della prova che il figlio maggiore (per età) deve essere collocato a sinistra, ci sarebbe stata un'ambiguità.

Le curiose proprietà dei nomi della famiglia Pre garantiscono qualcosa in più per l'albero binario: non solo conosciamo padri e figli, ma possiamo elencare i componenti della famiglia (ossia i *nodi* dell'albero), generazione per generazione, ossia livello per livello nell'albero, andando da sinistra a destra, in perfetta alternanza tra figli sinistri e figli destri. Questa proprietà consente di tenere i nodi semplicemente in una *tabella* (spesso chiamata *array*), più facile da gestire di quanto lo sia un albero.

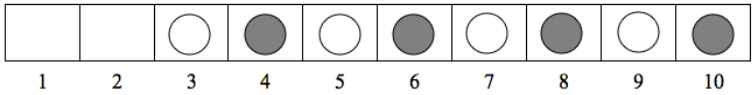
Si può cosí costruire un'altra importante struttura di dati: lo *heap* (*mucchio*, in italiano, ma quasi nessuno lo chiama cosí). In uno heap i dati devono essere collocati rispettando un'ulteriore regola: quella che *il dato associato a un figlio non sia maggiore del dato associato al padre*. Nel nostro caso un dato che evidentemente rispetta la regola potrebbe essere l'età: nessun padre ha età minore di quella dei figli!

Il punto importante è che nella *radice* dell'albero (il nonno per noi) ci sarà sempre il dato con il valore massimo: gli heap consentono di ottenere facilmente il valore *massimo* senza bisogno di tenere completamente ordinati i nostri dati, e quindi spesso risparmiando tempo. La regola, naturalmente, può anche essere invertita, sostituendo nella definizione *maggiore* con *minore*: in questo caso avremo nella radice il valore *minimo*, come accade per la famiglia Pre se il dato è la lunghezza del nome! Non solo, la nozione di heap è estendibile anche ad alberi in cui il numero dei figli può essere maggiore di due.

Un'ultima curiosità: perché la famiglia si chiama Pre? Perché i nomi dei componenti sono tutti *prefissi* del nome dell'ultimo nato.

**Parole chiave:** strutture di dati, alberi, alberi binari, heap, prefissi.

# Soluzione del quesito “Giochino giapponese”



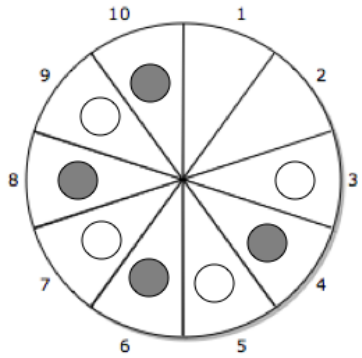

Si può codificare ciascuna mossa con un numero: se si numerano le caselle da 1 a 10, a partire da sinistra, la mossa $i$ consiste nello spostamento delle due pedine che occupano le caselle $i$ e $i + 1$ nei due posti vuoti. C’è una soluzione in cinque mosse: 9, 2, 5, 8, 1; ed è stato provato che è l’unica in cinque mosse ed è *la piú breve*: non c’è alcuna sequenza risolutiva che comporti un minor numero di mosse.

**Difficoltà e errori frequenti.** Trovare la soluzione piú breve è un problema piuttosto difficile; certamente piú semplice è arrivare a una delle ben 27 soluzioni in sei mosse (l’ultima delle quali è sempre 1)!

**Contesto informatico e riferimenti.** In verità, non sappiamo se questo puzzle provenga dal Giappone...

Quel che è certo è che la versione qui proposta è una variante di un problema che risale almeno agli anni ’80 dell’Ottocento, quando fu riportato da P. G. Tait, membro della Edinburgh Mathematical Society, e poi generalizzato e studiato da H. Delannoy, membro della Société Mathématique de France — come ricorda Edouard Lucas in una delle sue “Récréations mathématiques” pubblicata nel novembre del 1892, un mese dopo la sua morte. Nel gioco originale, la configurazione finale lascia libere le ultime due caselle a destra (anziché le prime due a sinistra): la soluzione col minor numero di mosse è unica e ne prevede soltanto quattro: 8, 5, 2, 9. Vedete l’analogia con quella del quesito che vi è stato proposto? Un fatto interessante: immaginando le caselle disposte in modo *circolare* come in figura, per risolvere il gioco originale, quello in cui la configurazione finale lascia libere le ultime due caselle a destra, basta spostare ad ogni mossa le due pedine che occupano il secondo e il terzo posto contando in senso antiorario rispetto alle due caselle vuote...

Se si vuol generalizzare il problema, aumentando (a destra) il numero di coppie di pedine (una bianca e una nera), il minimo numero di mosse necessario a risolverlo aumenta, in entrambi i casi, di un'unità per ogni coppia di pedine aggiunta. Tuttavia, l'unicità della soluzione piú breve si mantiene soltanto per la versione originale di Tait-Delannoy con 10 o 12 pedine in tutto; negli altri casi il numero delle diverse soluzioni di lunghezza minima aumenta piuttosto rapidamente col numero di coppie di pedine aggiunte.




Che cosa succede se si scambiano i colori delle pedine nella configurazione finale? L'unicità della soluzione piú breve è comunque perduta; inoltre, la versione che alla fine lascia libere le ultime due caselle a destra richiede una mossa in piú rispetto al caso "prima tutte le nere, poi tutte le bianche".

Le conclusioni sopra esposte sono state verificate mediante un programma al computer, che esplora "a tappeto" tutte le possibilità, e che noi abbiamo realizzato proprio per questo scopo. In sostanza, questo programma esprime un *algoritmo di esaurimento* (o *esauriente* o anche, usando un neologismo, *esaustivo*), che esamina *tutte* le possibili sequenze di mosse, lunghe al piú un dato numero (ad esempio 6), e stampa quelle risolutive, ossia che portano allo stato riconosciuto come "di successo". Per analizzare le varianti, basta cambiare la funzione che riconosce lo stato finale!

In generale, uno strumento di questo genere può rivelarsi utile finché la dimensione del problema rimane limitata: di solito, infatti, la complessità rispetto al tempo è di tipo *esponenziale* in qualche parametro ad essa legato. (Per il giochino che abbiamo proposto, ad esempio, se raddoppiamo il numero di pedine già occorrono ore di tempo-macchina per trovare e stampare tutte le soluzioni in nove mosse!) Sicché spesso succede che il tempo necessario diventi presto proibitivo... Vale la pena allora di applicare un po' di ingegno nel tentativo di capire quale sia la forma generale di una soluzione, ammesso naturalmente che ciò sia possibile!

**Parole chiave:** puzzle, struttura circolare, algoritmo di esaurimento, complessità esponenziale.

# Soluzione del quesito “Colora la mappa”

**Soluzione.** Occorrono quattro colori: infatti, ciascuna delle quattro regioni Marche, Toscana, Lazio e Umbria confina con le altre tre. D’altra parte, un celebre teorema, dimostrato nel 1976 da Kenneth Appel e Wolfgang Haken con l’ausilio del computer, afferma che quattro colori sono comunque sufficienti, per qualsiasi carta geografica. Uno dei tanti esempi di colorazione delle regioni italiane che usano il minor numero di colori è quello mostrato in figura.

**Difficoltà e errori frequenti.** Non vi sono particolari difficoltà. All’inizio, ovunque si parta, si deve cercare di usare il minor numero di colori, ovviamente rispettando la regola di non assegnare uno stesso colore a due regioni confinanti... Presto ci si accorge che ne servono almeno tre, sia al Nord sia al Sud, e quando si arriva al Centro si capisce che tre non bastano!

**Contesto informatico e riferimenti.** La dimostrazione del *teorema dei quattro colori* fu un evento di grande rilievo per la matematica “fatta con il calcolatore”. La storica questione venne posta nel 1852 da Francis Guthrie: quanti colori sono sufficienti per colorare una qualsiasi carta geografica in modo tale che due regioni adiacenti abbiano sempre colori diversi? Due precisazioni sono doverose: ciascuna regione è connessa (ossia “costituita da un solo pezzo”) e non si considerano adiacenti due regioni i cui confini si toccano in un numero finito di punti (altrimenti, per colorare una torta tagliata a fette, che si toccano nel centro, occorrerebbero tanti colori quante sono le fette): per dirsi “adiacenti” devono confinare per almeno un tratto.

Chi pose il problema considerò una mappa con quattro regioni, ciascuna adiacente alle altre tre (proprio come nella carta dell’Italia accade con Marche, Toscana, Lazio e Umbria), e osservò quindi che tre colori non bastano. Si noti che questa non è una condizione necessaria: riuscite a disegnare, ad esempio, una mappa con sole sei regioni, in cui non ve ne siano quattro mutuamente confinanti, eppure siano necessari quattro colori?

(Per inciso, stabilire se una data mappa può essere colorata con soli tre colori è un problema laborioso quanto quello del commesso viaggiatore, che abbiamo già incontrato lo scorso anno.) Piú tardi, fu dimostrato che cinque colori sono sempre sufficienti per qualsiasi mappa (Percy J. Heawood, 1890), ma soltanto nel 1976 fu annunciata una dimostrazione del fatto che ne bastano quattro. Per i matematici, l'aspetto davvero sorprendente — e per molti di essi quantomeno insoddisfacente — era il modo in cui si era giunti alla "dimostrazione": parti ingenti e cruciali di essa furono svolte da un computer, utilizzando risultati ottenuti a loro volta grazie ai calcolatori. La quantità di calcoli richiesta era tale da rendere praticamente impossibile il controllo di ogni passaggio da parte dell'uomo: il computer aveva dunque soppiantato il ricercatore nella costruzione di una parte significativa di una dimostrazione matematica. Oltre, ovviamente, a una massiccia analisi del problema e allo sviluppo con carta e matita delle complesse procedure di calcolo, erano state necessarie circa 1200 ore di tempo-macchina (di allora) per colorare effettivamente in tutti i modi possibili circa 2000 mappe, numero di configurazioni a cui i due matematici Appel e Haken erano riusciti a dimostrare — tutt'altro che banalmente — la riconducibilità di qualsiasi mappa.

Il grande matematico ungherese Paul Erdős non la riteneva una bella dimostrazione: avrebbe preferito "una dimostrazione che penetrasse il perché quattro colori sono sufficienti". Appel e Haken scrissero: "Se molti matematici si sentono in imbarazzo di fronte a lunghe dimostrazioni, può essere perché — fino a tempi assai recenti — essi hanno utilizzato soltanto quel genere di metodi che producono dimostrazioni brevi. Ancora non sappiamo se sia possibile o no trovare una dimostrazione breve del teorema dei quattro colori. [...] Noi siamo convinti che esistano teoremi di grande interesse matematico, la cui dimostrazione sia possibile soltanto grazie a tecniche che fanno uso di computer. [...] La nostra dimostrazione implica anche che nel passato l'importanza dei metodi di calcolo nelle dimostrazioni matematiche è stata decisamente sottovalutata, mentre è in pratica molto importante per il matematico determinare quali siano i poteri e i limiti dei propri metodi". E quest'ultima affermazione è senza dubbio condivisibile!

**Parole chiave:** colorazione di una mappa, teorema dei quattro colori, dimostrazione con l'ausilio del computer.

# Soluzione del quesito “Uno strano gioco dell’oca”

**Soluzione.** Per la categoria “Medie” la sequenza da trovare è la seguente:


Per la categoria “Biennio”, le sequenze da trovare sono quella nella figura qui sopra e la seguente:


Nella versione della prova per la categoria “Medie” abbiamo introdotto un suggerimento: “lo stolto si ferma al principio, il saggio comincia dalla fine”. I matematici non sono molto saggi: hanno impiegato secoli per capire che per risolvere, per esempio, un’equazione, conviene partire dalla soluzione, sia pure chiamandola “x”...

Ragioniamo dunque a rovescio (o meglio, all’indietro). Se siamo su una casella *dispari* (e maggiore di 1, se no siamo alla partenza) vuol dire che proveniamo, con una mossa A, dalla casella *immediatamente precedente*, perché la mossa B, quella a *salto* alla casella doppia, porta necessariamente a una casella pari.

Se viceversa siamo su una casella *pari*, allora dobbiamo aver fatto il salto (mossa B), perché se prima fossimo stati sulla casella immediatamente precedente saremmo stati costretti a saltare oltre, a meno di essere ora proprio sulla casella 2, a cui si arriva comunque dalla 1 (per la casella 1 le mosse A e B si equivalgono).

Abbiamo cosí scoperto che... procedendo all’indietro non abbiamo scelta! Se partiamo dunque dalla casella che vogliamo raggiungere e facciamo le mosse all’indietro, tutto diventa molto facile. Considerando i valori attribuiti alle caselle, per un valore *dispari*, *togliamo* 1, per un valore *pari*, *dimezziamo*. Pertanto, volendo arrivare alla

casella 27 basta costruire la sequenza $27 \leftarrow 26 \leftarrow 13 \leftarrow 12 \leftarrow 6 \leftarrow 3 \leftarrow 2 \leftarrow 1$ (il verso delle frecce indica lo spostamento nel gioco dell'oca, l'ordine della sequenza è quello in cui la costruiamo: i valori si potevano collocare nelle caselle della prova sia a partire da sinistra sia da destra).

**Difficoltà e errori frequenti.** La difficoltà del procedere in avanti è evidente: occorre cercare di anticipare le mosse successive, per fare le scelte giuste, e non è facile. Non è ovvio neppure che la sequenza da costruire sia unica: è proprio questo fatto che rende invece cosí semplice procedere al contrario; partendo dal punto d'arrivo, non ci sono scelte da fare!

**Contesto informatico e riferimenti.** Conoscete la leggenda degli scacchi? Se no potete trovare notizie nella benemerita *Wikipedia*. Comunque, sembra che all'inventore del gioco fosse stato promesso 1 chicco di grano per la prima casella della scacchiera, 2 per la seconda, 4 per la terza, e cosí via. Per la sessantaquattresima casella ci sarebbero voluti $2^{63}$ chicchi di grano, ossia un numero pari al prodotto di 2 per se stesso 62 volte.

Che cosa c'entra tutto questo con le oche e il relativo gioco? Un attimo di pazienza: intanto contate i chicchi... e riflettete. Per calcolare $2^{63}$ saranno proprio necessarie 62 moltiplicazioni? Non sarebbe bello poterlo calcolare in fretta e mentalmente? Possiamo farlo in maniera approssimata, chicco piú chicco meno, sfruttando... la fortuna.

La fortuna è che $2^{10}$ vale circa 1000, per la precisione 1024 (e questo bisogna ricordarselo: è facile pensando all'esponente 10 e al risultato 1000, circa). Ma allora $2^{30} = 2^{10} \times 2^{10} \times 2^{10} \cong 1000 \times 1000 \times 1000 = 1$ miliardo, e poi $2^{60} = 2^{30} \times 2^{30} \cong 1$ miliardo di miliardi, e manca ancora un fattore $2^3 = 8$, che alla fine darà un po' piú di 8 miliardi di miliardi (sommando i chicchi di tutte le caselle e facendo i conti un po' piú precisi si aggiungono altri 10 miliardi di miliardi di chicchi).

Non siete soddisfatti: volete sapere delle oche e anche calcolare il valore preciso di $2^{63}$? Ma avete notato che per calcolare $2^{60}$ bastava conoscere $2^{30}$ e per ottenere $2^{30}$ bastavano due moltiplicazioni a partire da $2^{10}$? E se facciamo il gioco dell'oca su 63 caselle? Partiamo... all'incontrario: $63 \leftarrow 62 \leftarrow 31 \leftarrow 30 \leftarrow 15 \leftarrow 14 \leftarrow 7 \leftarrow 6 \leftarrow 3 \leftarrow 2 \leftarrow 1$.

Come sfruttiamo il nostro strano gioco dell'oca per calcolare le potenze? Osserviamo che $2 \times 2 = 2^2$, $2 \times 2^2 = 2^3$, $2^3 \times 2^3 = 2^6$, $2 \times 2^6 = 2^7 \ldots$ e quindi possiamo costruirci tutte le *potenze* i cui *esponenti* compaiono nella sequenza riportata sopra facendo tante moltiplicazioni quante sono le *frecce* nella sequenza (cioè 10) e arriviamo cosí a calcolare $2^{63}$ con 10 moltiplicazioni invece di 62, un bel progresso! Incidentalmente, il valore 63 è uno di quelli per cui abbiamo il *caso peggiore*, in cui dobbiamo alternare un passo all'indietro e un salto: il *caso migliore* è quello con soli salti: $64 \leftarrow 32 \leftarrow 16 \leftarrow 8 \leftarrow 4 \leftarrow 2 \leftarrow 1$. E allora potremmo calcolare $2^{63}$ facendo solo le 6 moltiplicazioni corrispondenti alle 6 frecce della sequenza qui sopra e poi dividendo $2^{64}$ per 2 (senza fare le 10 moltiplicazioni della sequenza per il 63).

Senza usare le divisioni, possiamo in questo modo comunque calcolare una potenza come $2^n$ effettuando non $n - 1$ moltiplicazioni ma soltanto all'incirca un numero di moltiplicazioni pari al *logaritmo* (in base 2) di $n$, dove il logaritmo è una *funzione* il cui valore cresce *molto* meno rapidamente di $n$: per esempio, il logaritmo di $2^{64}$ è proprio 64, e abbiamo visto che, per calcolare questa potenza, bastavano 6 moltiplicazioni, e 6 è proprio il logaritmo di 64, perché $2^6 = 64$.

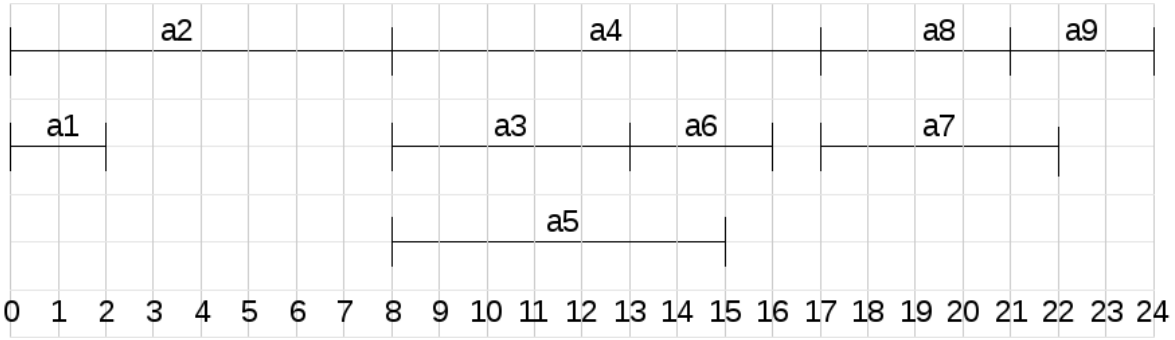
Conclusione: abbiamo scoperto, oltre che un modo per vincere al nostro gioco dell'oca, una precisa procedura di calcolo, un *algoritmo* per calcolare le potenze col *minimo* numero di moltiplicazioni. Scusate, non proprio, bisogna sempre stare attenti a non fare affermazioni azzardate: chi ha detto che il numero è *minimo*? Per esempio, $15 \leftarrow 14 \leftarrow 7 \leftarrow 6 \leftarrow 3 \leftarrow 2 \leftarrow 1$ dà 6 moltiplicazioni, mentre $15 \leftarrow 12 \leftarrow 6 \leftarrow 3 \leftarrow 2 \leftarrow 1$ ne richiede 5 (l'ultima moltiplicazione dà $2^{15} = 2^{12} \times 2^3$). Ma la sequenza viola le regole del nostro gioco dell'oca (dalla casella 12 non si può saltare alla 15). Quali sono allora le regole di questo nuovo gioco dell'oca? Scopritele voi: chi ha detto che dobbiamo rispondere a tutto?

**Parole chiave:** algoritmi, caso peggiore e caso migliore, potenze di 2, calcolo delle potenze, logaritmi, catene di addizioni.



# Soluzione del quesito “Lavori di manutenzione”

**Soluzione.** Le attività previste potranno essere completate in 24 ore come minimo. C’è infatti un *percorso critico*, che stabilisce un *limite inferiore* alla durata dei lavori: in questo caso è costituito dalle attività a2, a4, a8 e a9, che complessivamente occupano 24 ore. Essendo tre i ragazzi, ce la possono fare in questo tempo, come mostra il seguente *diagramma di Gantt*, che riporta la collocazione temporale nel caso in cui ciascuna attività sia fatta iniziare il piú presto possibile:




Si noti che questo diagramma ha il minimo numero di righe, ma non rispecchia necessariamente una ripartizione delle attività tra i ragazzi, ossia non è detto che a ciascuna riga corrisponda un nome: qui si riesce a distribuire equamente il lavoro, nel modo migliore senza “spezzare” le attività, assegnando ad esempio ad Aldo a1, a4 e a7 (per un totale di 16 ore); a Bruno a2 e a5 (15 ore); a Carlo a3, a6, a8 e a9 (15 ore)... ma questo sarebbe stato un problema piú difficile!

**Difficoltà e errori frequenti.** Il problema specifico che è stato proposto non è particolarmente arduo, anche se comporta un po’ di calcoli: qui, infatti, bastava trovare il percorso piú “costoso” (in termini di tempo) da una

delle attività che non sono precedute da altre (a1 e a2) a una di quelle che non sono seguite da altre (a7 e a9), ed è facile accorgersi che tale percorso è costituito da a2, a4, a8 e a9. Dunque, il periodo di tempo minimo non può essere inferiore a 24 ore. Nella versione per la categoria "Medie", questa osservazione portava ad escludere due delle cinque risposte possibili. Provando poi a disporre le altre attività in modo da rispettare i vincoli di precedenza ma anche da sfruttare ed evidenziare il parallelismo (almeno di a2 con a1 e di a4 con a3 e a5), si arriva senza eccessiva difficoltà a un diagramma di tre righe come quello sopra riportato.

**Contesto informatico e riferimenti.** Il quesito proposto è un problema di pianificazione (della produzione o di un progetto: in inglese è chiamato *project scheduling*) che consiste nell'ordinare in senso temporale un insieme di attività, in modo da minimizzare il tempo di completamento rispettando la relazione di precedenza tra le attività stesse. Tale relazione è rappresentata dagli archi del grafo orientato (che, chiaramente, deve essere privo di cicli) dato nella formulazione del problema.

Tecniche, dette *reticolari*, per affrontare problemi di questo tipo — di particolare interesse aziendale — furono sviluppate a partire dalla seconda metà degli anni '50. Se si ipotizza che le *risorse* siano *illimitate*, allora la questione è *trattabile* (dal punto di vista computazionale). Questa ipotesi non rientra nel problema proposto, dove i ragazzi sono soltanto tre; tuttavia, anche supponendo di poter reclutare in aiuto quanti ragazzi si vogliono, il tempo totale resterebbe di 24 ore — per cui si vede che tre ragazzi bastano per eseguire tutti i lavori in questo lasso di tempo. (Una piccola ammenda: nella realtà dei fatti, trattandosi di lavori di manutenzione, non è affatto plausibile ipotizzare che la durata di un'attività non cambi a fronte di un aiuto; tuttavia qui si voleva proporre un problema con determinate caratteristiche, e con tempi prefissati e certi.)

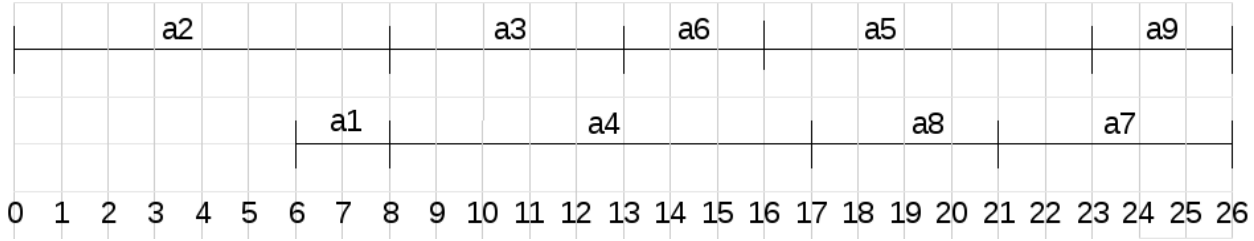
Si possono fare alcune interessanti osservazioni, rimanendo nel limite inferiore di tempo (ammesso che certe attività non si prolunghino):

- a1 può slittare in avanti (se ci fosse un quarto ragazzo, potrebbe addirittura sovrapporsi ad a3: l'importante, infatti, è che finisca entro l'inizio di a6);
- d'altra parte, anche la coppia a3–a6 o soltanto a6 possono slittare in avanti di un'ora;



- e infine a5 può slittare in avanti fino a sei ore, o chi la svolge se la potrebbe prendere piú comoda!

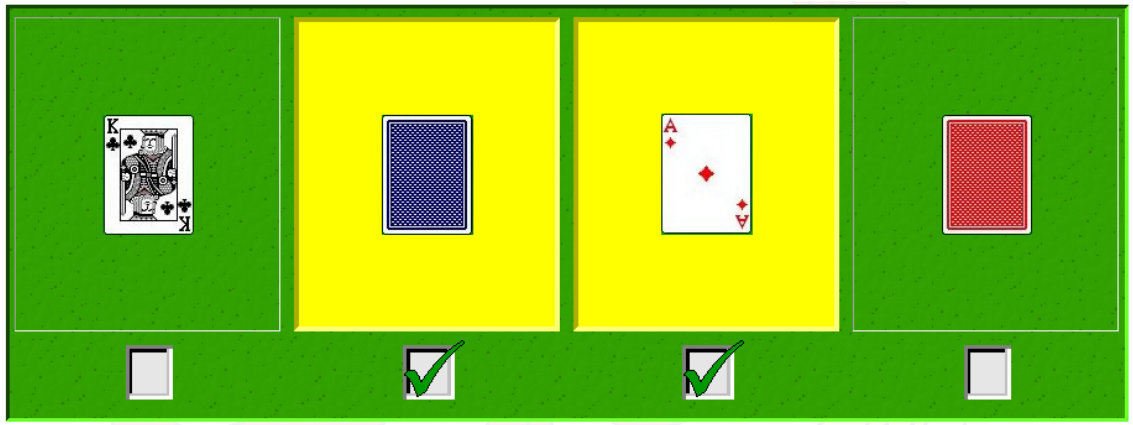
Assai piú difficile è l'analogo problema nell'ipotesi di *risorse limitate*, quando è davvero restrittiva. Si possono seguire criteri *euristici*, ma sostanzialmente — se si vuole la soluzione ottima — bisogna cercarla tra tutte le possibili. Ad esempio, se i ragazzi fossero soltanto due, Aldo e Bruno, di quanto si allungherebbe la durata dei lavori? Di sole *due* ore, ed ecco la soluzione ottima (dove a1, oltre ad anticipare, può scambiarsi con a2):




Qui una riga del diagramma dovrà riferirsi ad Aldo, l'altra a Bruno. Questo diagramma si può ottenere partendo dall'istante finale (sconosciuto) e assegnando a ritroso l'attività piú lunga, compatibilmente con i vincoli di precedenza, al primo ragazzo libero, che se ne prenderà carico. Ovviamente questo criterio non assicura la soluzione ottima! Adottando invece l'algoritmo che parte dall'istante iniziale e, procedendo in avanti, assegna ancora l'attività piú lunga (come sopra), per questo specifico problema si troverebbe una soluzione peggiore: 30 ore complessive; altri criteri portano a durate intermedie. Per ogni regola che può essere stabilita nell'assegnare i vari compiti, si possono trovare dei casi in cui darà il risultato migliore, ma pure altri casi in cui darà il peggiore! E, quando la dimensione del problema aumenta, la ricerca della soluzione ottima diviene presto intrattabile: ricordate i problemi NP-completi che abbiamo incontrato lo scorso anno, come quello del commesso viaggiatore?

**Parole chiave:** *project scheduling*, percorso critico, diagramma di Gantt, risorse, criteri euristici.

# Soluzione del quesito “Carte rosse e carte blu”



**Soluzione.** Le carte che contribuiscono a stabilire il *valore di verità* (vero o falso) dell'affermazione del quesito sono essenzialmente due: la seconda (che ha il retro blu) e la terza (l'asso di quadri). Infatti soltanto queste due carte potrebbero smentire l'affermazione: se la seconda carta non fosse un re, allora avremmo un esempio di carta col retro blu che non è un re, quindi l'affermazione in questione sarebbe smentita; analogamente, se il retro della terza carta fosse blu, avremmo un esempio di carta col retro blu che non è un re, e anche in questo caso l'affermazione sarebbe smentita.

**Difficoltà e errori frequenti.** Di primo acchito si può essere indotti a pensare che il re di fiori abbia importanza ai fini della richiesta... Invece no, perché il colore del suo retro non influenza il valore di verità dell'affermazione fatta: si vuole stabilire se le carte col retro *blu* sono *re*, per cui il colore del retro di un re (non importa di quale seme) non aggiunge alcuna informazione a quanto già si sa. Anche il valore della carta col retro rosso è del tutto indifferente per la questione posta.

Lo stesso quesito potrebbe essere riformulato in un contesto piú usuale, in cui, avendo maggiore esperienza, la soluzione del quesito appare piú intuitiva. Considerate l'affermazione “ogni minorenne beve un analcolico” e immaginate di vedere quattro persone con un bicchiere in mano: una ragazza con un bicchier d'acqua, un bambino, un ragazzo con una birra e un vecchio. In questo caso è evidente che basta verificare che cosa sta bevendo il bambino e che età ha il ragazzo con la birra! Vedete l'analogia col quesito originale delle quattro carte?

**Contesto informatico e riferimenti.** Il quesito si ispira ad uno dei piú famosi paradigmi sperimentali della psicologia del ragionamento, proposto da Peter Wason nel 1966 e noto come "test delle quattro carte".

La domanda ha a che fare con un *connettivo logico* che si chiama *condizionale* o *implicazione materiale*: $A \to B$, che si legge "se $A$ allora $B$" o "$A$ implica $B$", è una *proposizione*, dove $A$ (*antecedente*) e $B$ (*conseguente*) sono a lora volta due proposizioni. $A \to B$ equivale a "(non $A$) oppure $B$"; quindi è vera quando $A$ è falsa oppure $B$ è vera ("oppure" si deve intendere in senso inclusivo, come il *vel* latino, cioè $A \to B$ è vera anche quando si ha contemporanemente che $A$ è falsa e $B$ è vera). Viceversa, $A \to B$ è falsa se e soltanto se $A$ è vera e $B$ è falsa. Piú semplicemente, si può dire che $A \to B$ abbia questo significato: "non può succedere che $A$ sia vera e $B$ sia falsa". Nel quesito proposto:

$$A = \text{ la carta ha il retro blu} \qquad\qquad B = \text{ la carta è un re}$$

e l'affermazione fatta nel testo sarà vera se e soltanto se l'implicazione $A \to B$ è vera *per ognuna* delle quattro carte. In parole piú semplici, per ogni carta si deve vedere se non ha il retro blu oppure è un re. A questo punto, è facile constatare che l'implicazione è vera sia per la prima carta (perché $B$ è vera) sia per la quarta carta (perché $A$ è falsa), senza bisogno di voltarle! Invece, la seconda carta ha il retro blu: quindi $A$ è vera ed è $B$ che decide il valore di verità dell'implicazione; e la terza carta non è un re: quindi $B$ è falsa ed è $A$ che decide.

Bisogna fare molta attenzione al linguaggio della logica, su cui pure l'informatica si fonda; nei linguaggi naturali le cose stanno un po' diversamente: spesso, in italiano, con "se ... allora ..." si intende esprimere un nesso *causale* o una relazione *temporale* tra i *contenuti* dell'antecedente e del conseguente, sicché talvolta può non essere intuitiva la verità di un'implicazione logica quando l'antecedente è falso.

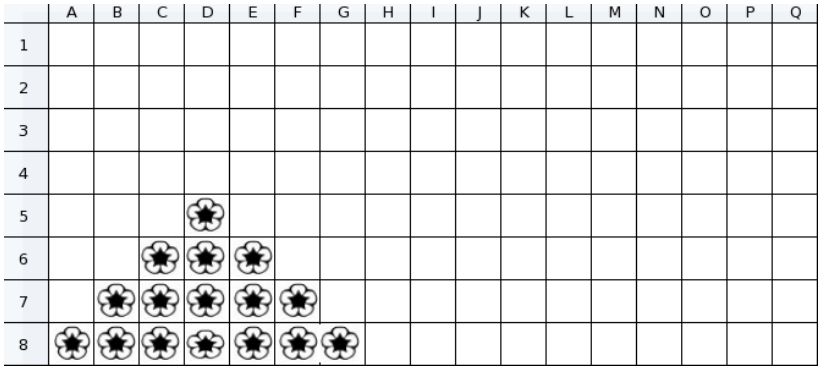
Questa forma "essenziale" di implicazione, che è anche la piú conveniente per le scienze matematiche, fu adottata da Crisippo di Soli (vissuto nel III secolo a.C.), che la riprese da Filone di Mégara. A Crisippo si devono anche alcune importanti *regole di inferenza* (ad esempio il *modus ponens*) e il classico "paradosso del mentitore", a cui dedicò ben 28 dei suoi settecento "libri" perduti...

**Parole chiave:** logica, valore di verità, proposizione, implicazione materiale.

# Soluzione dei quesiti “Vasi di fiori” e “Acchiappa bug”

Il quesito “Acchiappa bug” per la categoria “Biennio” include il quesito “Vasi di fiori” della categoria “Medie”. La soluzione è quindi divisa in due parti: la prima riguarda il quesito comune alle due categorie, la seconda riguarda solo la categoria “Biennio”.

**Soluzione del quesito comune alle due categorie.** La risposta corretta è quella illustrata in figura, dove abbiamo aggiunto lettere e numeri per individuare univocamente la posizione di ciascuna cella.




Per rispondere al quesito occorre eseguire passo passo le *istruzioni* della procedura, posizionandosi correttamente nella piastrella di partenza, calcolando di quante piastrelle spostarsi e disponendo i vasi secondo le indicazioni.

Le prime cinque istruzioni sono semplici da eseguire, e corrispondono a quella che si chiama fase di *inizializzazione* di una procedura.

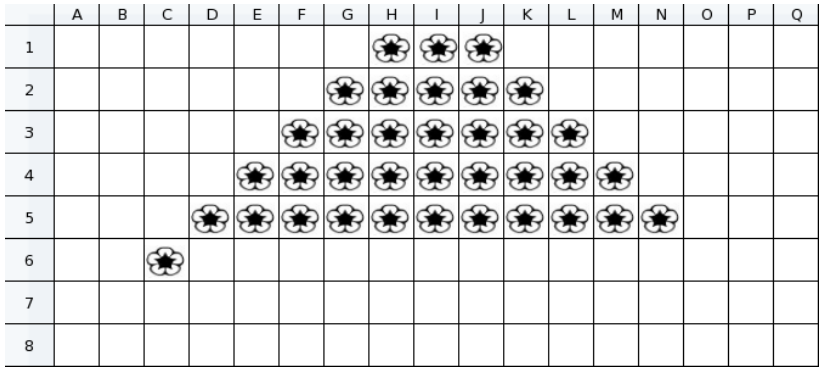
1. Il punto di partenza è la piastrella $(A,1)$.
2. $P$ è il numero di piastrelle sul lato corto del cortile, quindi $P = 8$.
3. $V$ è il numero di vasi a disposizione, quindi $V = 16$.
4. Inizialmente il valore da segnare su un foglio è il numero 1; per comodità, lo indicheremo con $f$.
5. $N$ è il risultato dell'operazione $1/4 * V$, quindi $N = 16/4 = 4$.

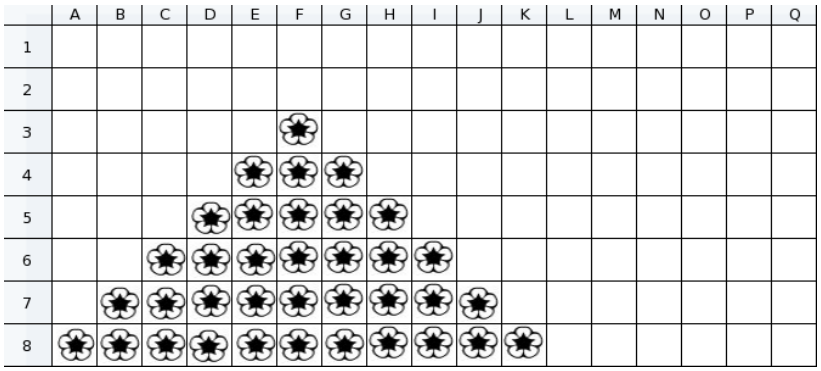
L'istruzione 6 prescrive di eseguire $P$ volte (quindi 8 volte) le istruzioni dalla a. alla f. Ciascuna di queste 8 ripetizioni è chiamata *iterazione*. Le istruzioni chiedono di calcolare alcuni valori e indicano al geometra come spostarsi nel cortile e dove disporre i vasi. Segniamo nella seguente tabella la sequenza dei risultati delle operazioni eseguite: ogni riga corrisponde ad una iterazione; le colonne corrispondono alle istruzioni. Per ogni iterazione e

per ogni istruzione viene riportato il risultato dell'operazione eseguita (il valore calcolato o la posizione raggiunta, come indicato nell'intestazione della tabella).

| iterazione | valore di $P-f$ dopo l'istruz. a. | posizione raggiunta dopo l'istruz. b. | valore di $M$ dopo l'istruz. c. | posizione raggiunta dopo l'istruz. d. | valore di $f$ dopo l'istruz. e. | posizione raggiunta dopo l'istruz. f. |
|---|---|---|---|---|---|---|
| 1 | 7 | $(H,1)$ | $-7$ | $(H,1)$ | 2 | $(A,2)$ |
| 2 | 6 | $(G,2)$ | $-5$ | $(G,2)$ | 3 | $(A,3)$ |
| 3 | 5 | $(F,3)$ | $-3$ | $(F,3)$ | 4 | $(A,4)$ |
| 4 | 4 | $(E,4)$ | $-1$ | $(E,4)$ | 5 | $(A,5)$ |
| 5 | 3 | $(D,5)$ | 1 | $(D,5)$ | 6 | $(A,6)$ |
| 6 | 2 | $(C,6)$ | 3 | $(E,6)$ | 7 | $(A,7)$ |
| 7 | 1 | $(B,7)$ | 5 | $(F,7)$ | 8 | $(A,8)$ |
| 8 | 0 | $(A,8)$ | 7 | $(G,8)$ | 9 | $(G,8)$ |

**Soluzione del quesito aggiuntivo per la categoria "Biennio".** Nel caso in cui il numero di vasi è $V = 36$, eseguendo la procedura progettata da Dueceli si otterrebbe la disposizione rappresentata nella figura a sinistra. Si nota chiaramente che la figura ottenuta non è un triangolo: c'è qualcosa che non va! La disposizione voluta è invece quella rappresentata a destra.



Per ottenerla, la procedura va modificata sostituendo la formula "$1/4 * V$" con "$\sqrt{V}$". Notate che le righe del triangolo sono fatte da numeri dispari consecutivi di vasi (1, 3, 5...). Ciò spiega la presenza della radice quadrata: infatti la somma dei primi $x$ numeri dispari è pari a $x^2$. Per 16 vasi è possibile sostituire $\sqrt{16}$ con $16/4$ perché danno lo stesso valore, ma cosí facendo l'algoritmo non è utilizzabile quando il numero di vasi è diverso da 16.

**Difficoltà e errori frequenti.** La procedura non è difficile in sé, tuttavia è abbastanza lunga da eseguire e non è facile intuire che cosa prescrive senza eseguirla passo passo. Questo probabilmente è dovuto al fatto che il lavoro da svolgere è scomposto e quindi definito in termini di operazioni molto elementari rispetto a quelle che useremmo comunemente per descrivere un lavoro di questo tipo. Inoltre le operazioni aritmetiche richieste, benché molto semplici, coinvolgono molte *variabili*: $V$, $P$, $N$, $M$, $f$. È importante tenere traccia di tutti i loro *valori*, che naturalmente possono cambiare (o, meglio, *variare*) durante l'esecuzione della procedura stessa (in realtà, alcune di queste, come $P$, $V$ e $N$, hanno un valore *costante* per tutta l'esecuzione della procedura). Insomma: bisogna mantenere altissima la concentrazione! Il secondo quesito proposto per la categoria "Biennio" era invece piú difficile poiché non richiedeva la mera esecuzione di una procedura. Il compito era però semplificato, grazie alla presenza di alcune opzioni tra cui scegliere.

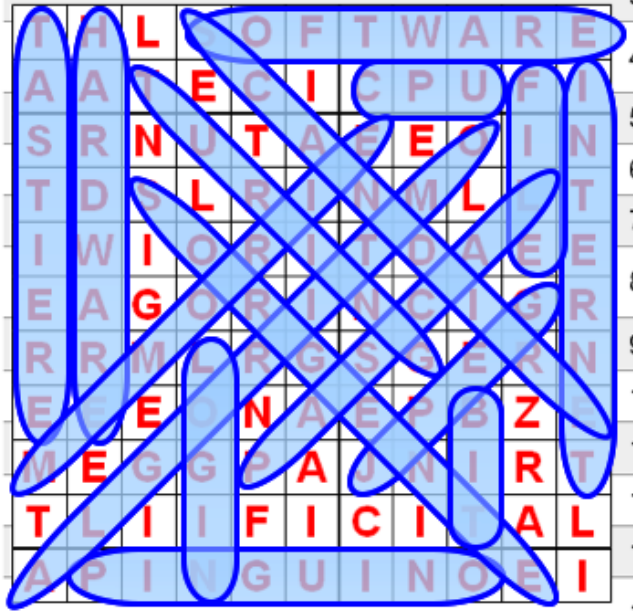
**Contesto informatico e riferimenti.** La procedura presentata nel quesito fornisce un esempio di *algoritmo*. In informatica, col termine algoritmo si intende un metodo per la soluzione di un problema, descritto attraverso passi elementari. Naturalmente chi progetta algoritmi, o —a maggior ragione— chi li traduce in *programmi* usando un *linguaggio di programmazione*, deve preoccuparsi di controllare la *correttezza* dei propri algoritmi e/o programmi. Spesso si possono verificare malfunzionamenti in un programma; ad esempio ci si accorge che con certi dati in ingresso (*input*) si ottiene un risultato (*output*) diverso da quello atteso. In questo caso, è necessario analizzare il programma alla ricerca dell'errore che ne ha determinato il malfunzionamento. In gergo, un errore di programmazione è chiamato *bug* (in italiano si usa a volte la parola *baco*) e l'attività che consiste nel ricercare ed eliminare i *bug* da un programma è detta *debugging*. Da qui il titolo del quesito nella versione per la categoria "Biennio", cioè "Acchiappa bug".

**Parole chiave:** algoritmo, esecuzione e correttezza, variabile, iterazione, debugging.



# Soluzione del quesito “Crucipuzzle informatico”

**Soluzione.** La soluzione corretta è presentata nella figura:



| | | |
|---|---|---|
| 1- | ALGORITMO | procedimento, specificato rigorosamente e senza ambiguità, per la risoluzione di un problema |
| 2- | BIT | cifra binaria |
| 3- | CPU | acronimo inglese della "unità centrale di elaborazione" di un computer |
| 4- | FILE | unità, costituita da una sequenza di byte e individuata da un nome, secondo cui sono organizzati i dati su un hard disk o altro dispositivo di memoria permanente |
| 5- | HARDWARE | insieme delle parti fisiche di un computer |
| 6- | INTERNET | la rete delle reti |
| 7- | JPEG | un formato standard per rappresentare immagini |
| 8- | LOGIN | procedura di accesso ad un sistema o un'applicazione informatica, in cui a volte si richiede anche l'inserimento di una password |
| 9- | MEMORIA | Io sono ad esempio la RAM, l'hard disk, una chiavetta USB |
| 10- | PASCAL | linguaggio di programmazione così chiamato in onore di un grande matematico e filosofo francese |
| 11- | PINGUINO | mascotte del sistema operativo libero Linux |
| 12- | SCANDIRE | acquisire un'immagine usando uno scanner |
| 13- | SOFTWARE | la parte che non si tocca dell'informatica... |
| 14- | SORGENTE | si chiama così il codice di un programma formato da istruzioni appartenenti ad un determinato linguaggio di programmazione |
| 15- | TASTIERA | ce ne sono QWERTY e DVORAK |
| 16- | TURING | matematico inglese, considerato uno dei padri dell'informatica, al quale è dedicato un premio internazionale |

Le definizioni dei termini informatici sono concise ma speriamo sufficientemente chiare. Per approfondire (per esempio per informazioni sulla *tastiera DVORAK* o sul *premio Turing*) è possibile usare la rete e in particolare Wikipedia. Cancellando le parole nella griglia si evidenzia la frase misteriosa "LE INTELLIGENZE ARTIFICIALI": i metodi e gli scopi delle ricerche sull'intelligenza artificiale possono essere alquanto diversi da quelli comunemente associati all'intelligenza "naturale", cosí come un sommergibile ha funzionamento e finalità diverse dal nuoto.

**Difficoltà e errori frequenti.** Individuare le parole nella griglia è piú facile se si ha esperienza con analoghi giochi enigmistici, e naturalmente è d'aiuto conoscere i termini cui si riferivano le definizioni, ma è possibile comunque individuare le parole e abbinarle con buon senso alle definizioni.

**Contesto informatico e riferimenti.** Il contesto informatico è abbastanza ovvio: si tratta di apprendere un po' di terminologia, con qualche spunto magari per ulteriori approfondimenti.

**Parole chiave:** terminologia, storia dell'informatica, sistemi operativi, architettura degli elaboratori.