Problemi da informatici

Università di Pisa, Dipartimento di Informatica

https://didattica.di.unipi.it/laurea-in-informatica/

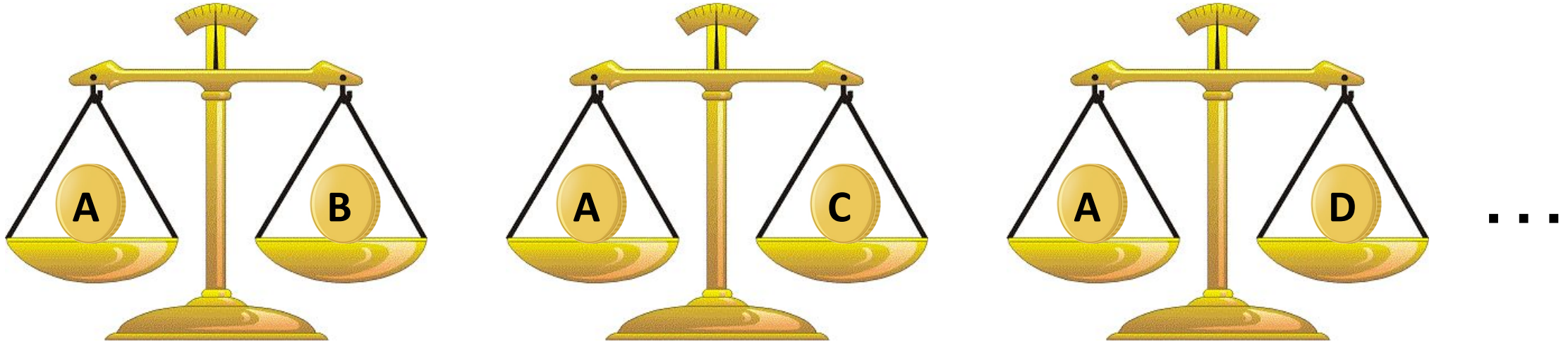
Problemi da informatici!

Se vi piacciono questi problemi, e vi interessa provare a risolverli, siete un po' informatici anche voi!

Il Problema delle 8 Monete

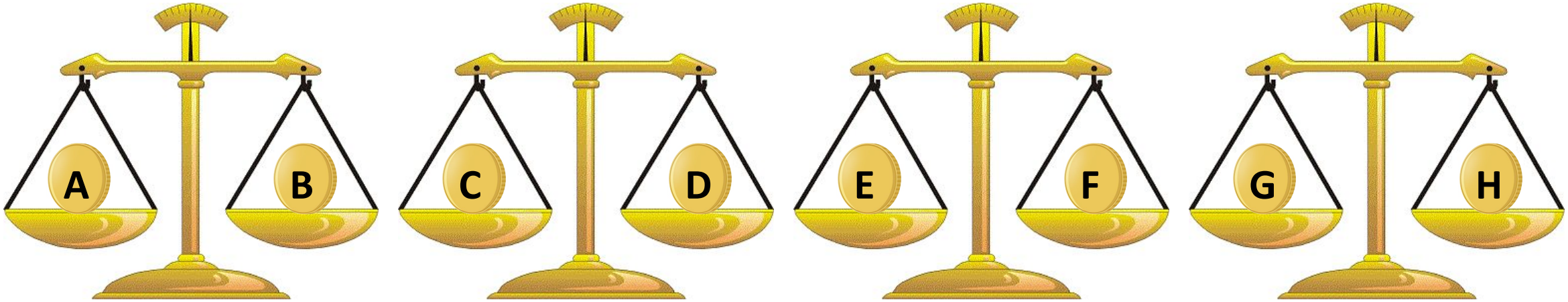
- **8** monete di cui 7 uguali e **1 falsa** con un peso minore
- Una bilancia classica con due piatti
- **Obiettivo**: trovare la moneta falsa con **meno pesate possibili**

Tante pesate



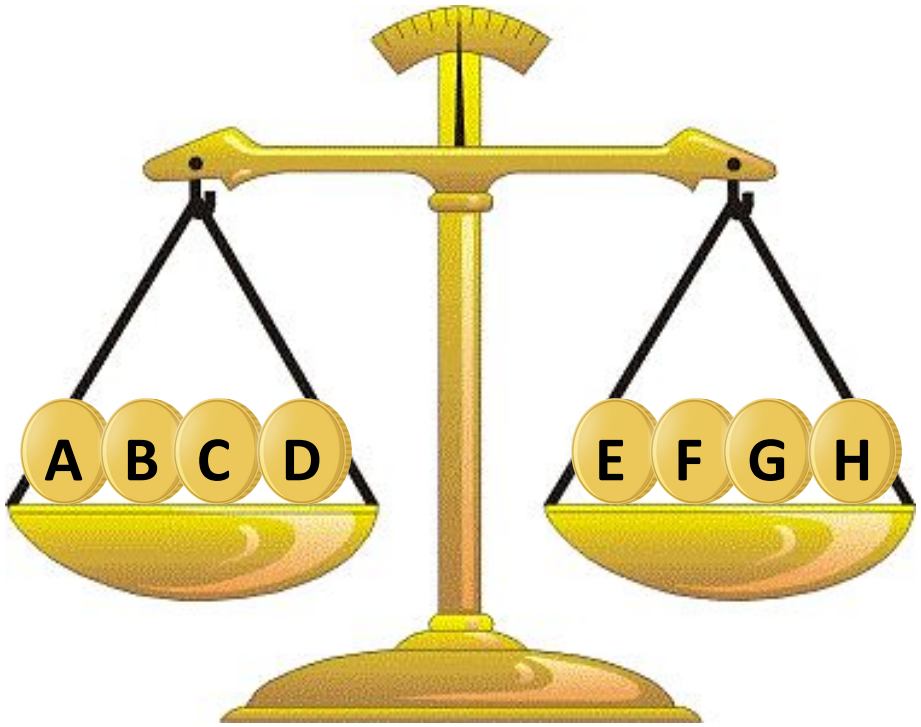
Peso tutte le coppie possibili

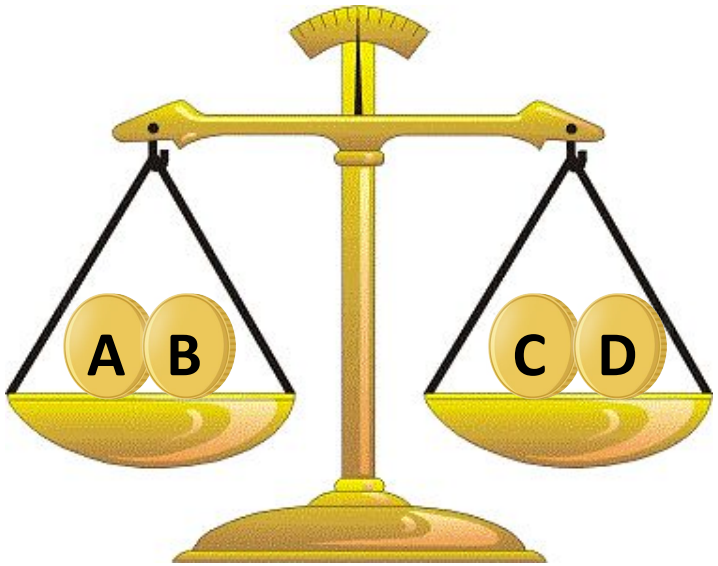
4 pesate

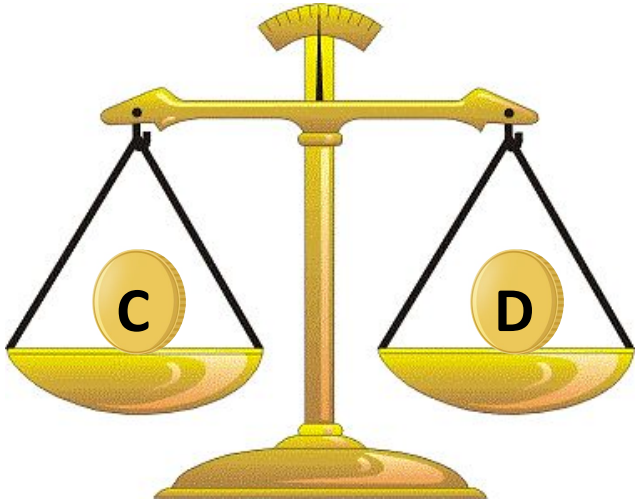


Peso ogni moneta una volta sola.

3 pesate







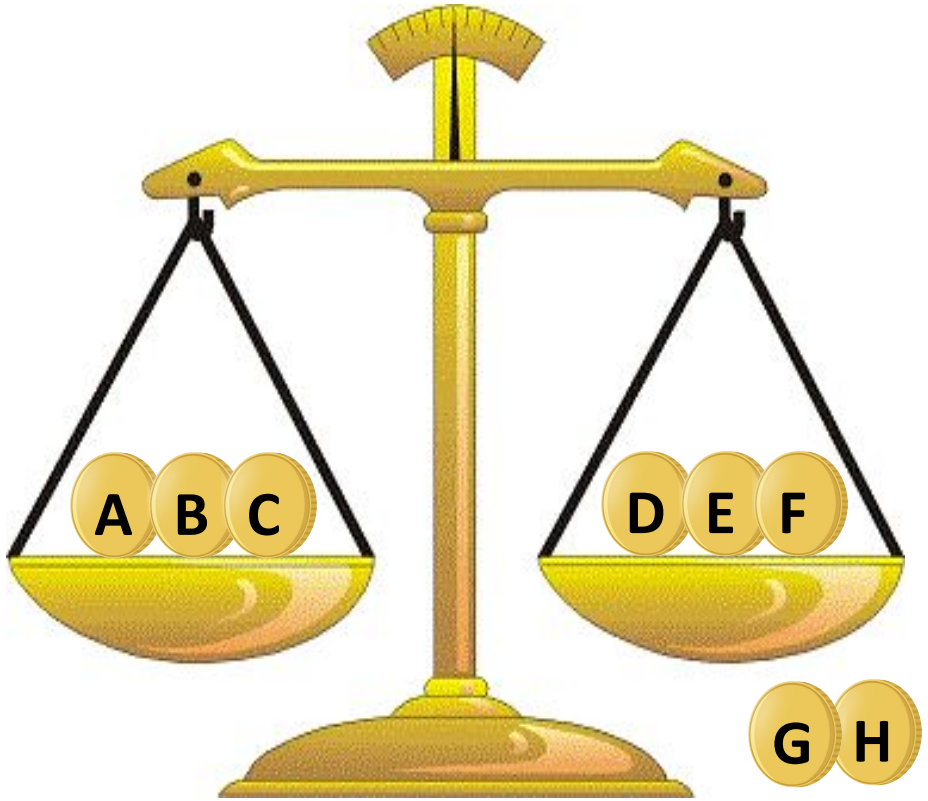
Peso a gruppi di 4, poi 2, poi 1

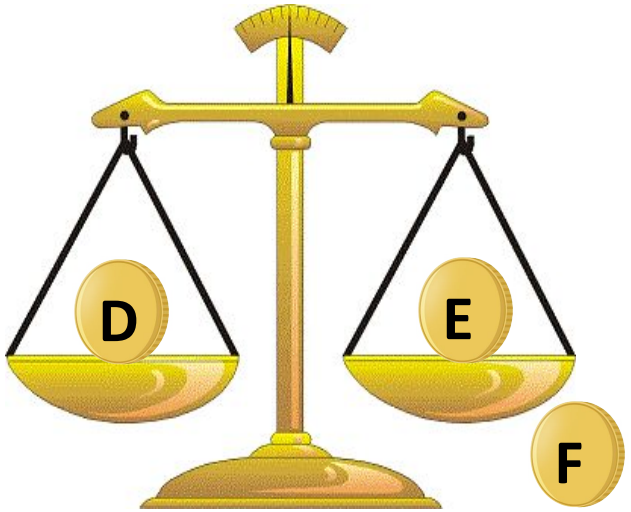
1 pesata?

No! 3 possibili risposte non bastano per distinguere 8 casi

2 pesate = 3x3 = 9 risposte diverse, ma devo dividere meglio i casi tra i tre risultati della prima pesata.

Soluzione migliore: 2 pesate





3 su ogni piatto, 2 da parte: individuo il gruppo (ABC, DEF, oppure GH)

Poi confronto due monete di quel gruppo

Cosa c’entra con l’informatica?

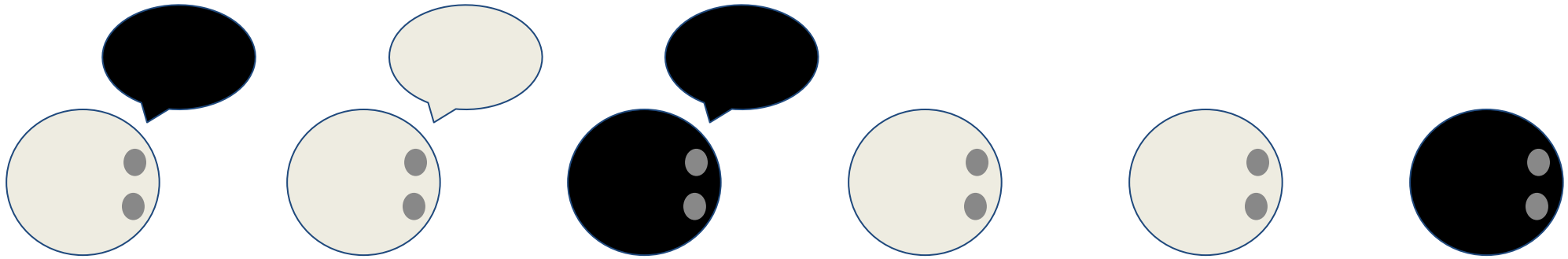
È molto simile all’analisi degli algoritmi di ordinamento che si studia al primo anno.

Trovare una soluzione ottimale è capire (e dimostrare) cosa **non** si può fare con gli strumenti a disposizione, e dalla comprensione del problema ricavare idee che portano a una soluzione.

Bonus: possiamo sempre trovare la falsa tra 9 monete anziché 8? Quante monete al massimo possiamo avere se le pesate a disposizione sono 3, 4, 5, … ?

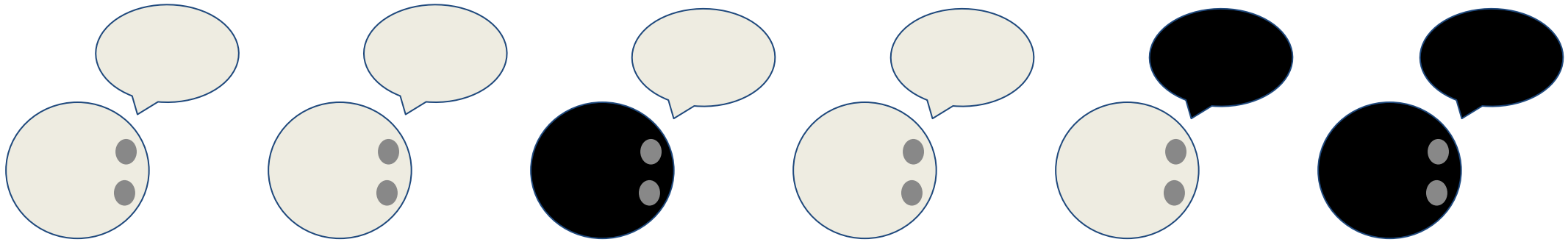
Cappelli bianchi e neri

Sei avventurieri vengono catturati da un mago che li vuole sottoporre a una prova di intelligenza. Domani verranno messi in fila indiana, e ognuno di loro indosserà un cappello di colore bianco oppure nero (scelto a caso). A turno, a partire da quello dietro a tutti, dovranno cercare di indovinare il colore del loro cappello, dicendo ad alta voce “bianco” oppure “nero”. Chi indovina avrà salva la vita. Questa notte, nella loro cella, gli avventurieri elaborano una strategia per permettere al maggior numero possibile di salvarsi. Qual è questa strategia?



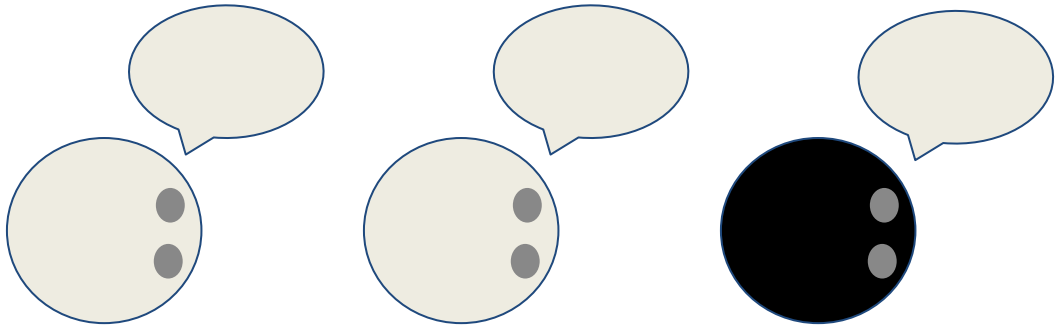
Salviamo sicuramente 3 persone

Il primo, il terzo e il quinto dicono il colore del cappello dell'avventuriero di fronte a loro. In questo modo lui può salvarsi!



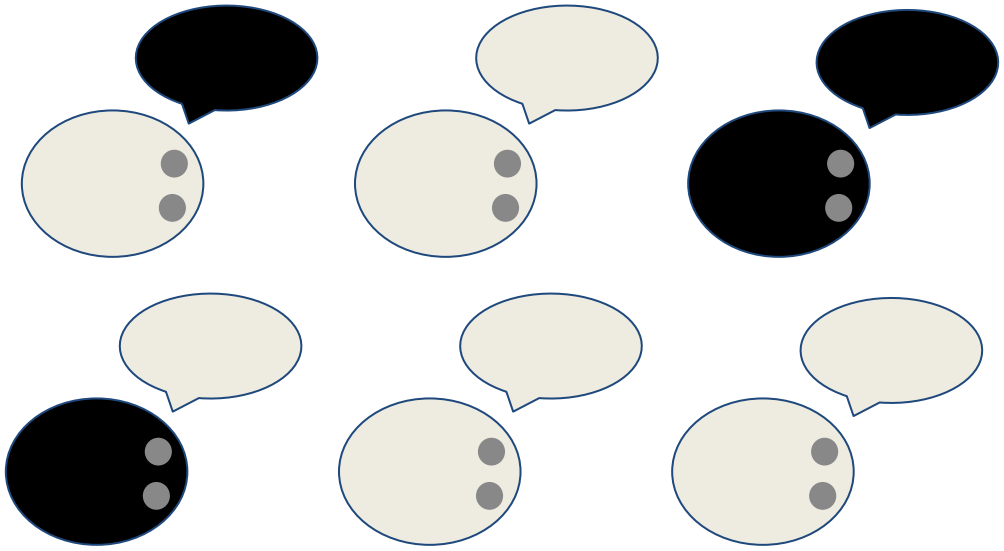
Un caso più semplice

Supponiamo di avere solo tre avventurieri. Quello più indietro non può certamente salvarsi se non tirando a caso, perché nessuno sa il colore del suo cappello. Può però dare un'informazione che permette di salvare gli altri due?



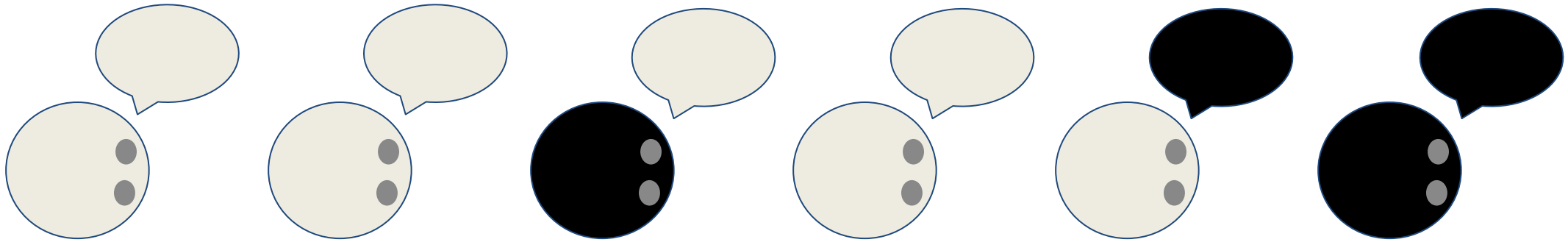
Un caso più semplice: soluzione

Sì! Per esempio può dire “bianco” se i due cappelli sono dello stesso colore, e “nero” altrimenti. In questo modo il secondo avventuriero **vede** il colore del cappello davanti a lui e sa cosa rispondere, mentre il terzo avventuriero **sente** cosa dice il secondo e sa cosa rispondere.



Soluzione migliore: 5 persone (e mezza)

Il primo dice “bianco” se vede un numero **pari** di cappelli neri, e “nero” se ne vede un numero **dispari**. In questo modo ogni avventuriero **vede** i cappelli di fronte a lui, **sente** le risposte precedenti, e sa cosa rispondere.



Cosa c’entra con l’informatica?

Cappelli bianchi e neri rappresentano 0 e 1 del codice binario. Il primo avventuriero calcola la funzione XOR (or esclusivo) dei cappelli di fronte a lui, una funzione che studierete.

La strategia generale è quella di fornire un **bit di parità** che permette di ricostruire un’informazione mancante in una stringa di bit. Questa idea è usata in molte situazioni in cui i dati possono venire **corrotti** ed è necessaria una ``cifra di controllo’’: nelle reti di calcolatori, nella memoria RAM (ECC) nei dischi SSD dei server (RAID).

Amore a Kleptonia

Sheldon e Amy sono innamorati. Amy vorrebbe inviare un anello per posta a Sheldon.

Sfortunatamente, loro vivono a Kleptonia dove qualunque cosa spedita per posta viene rubata a meno che non sia in una **scatola lucchettata**.

Sheldon e Amy hanno dei lucchetti, ma nessuno dei due ha le chiavi dei lucchetti dell'altro.
Come può Amy far arrivare l'anello a Sheldon?

Soluzione

Amy invia a Sheldon una scatola con un lucchetto, che chiamiamo A (e tiene la chiave A).

Sheldon applica alla scatola un altro lucchetto, che chiamiamo S (tenendone la chiave), e la rispedisce indietro con due lucchetti.

Amy rimuove il lucchetto A, e restituisce a Sheldon la scatola con il solo lucchetto S.

Quando Sheldon riceve la scatola, ha la chiave del lucchetto S e può aprirla!

Cosa c’entra con l’informatica?

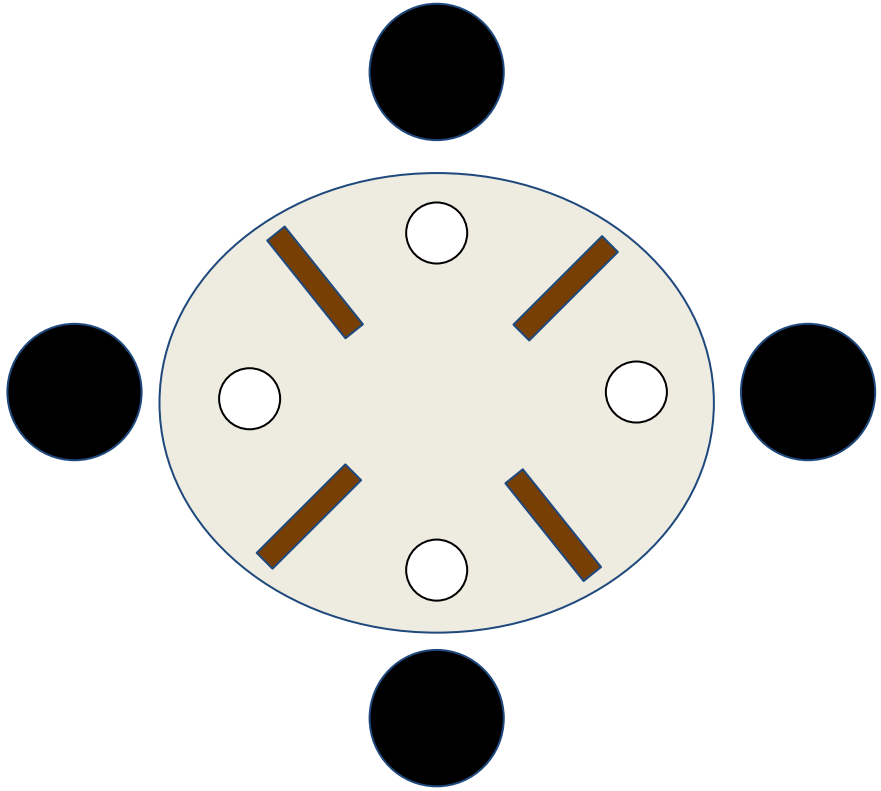
Questo puzzle simula il problema di spedire messaggi attraverso un canale di comunicazione sicuro.

Un famoso protocollo di crittografia (three-pass protocol) prevede di cifrare e decifrare messaggi proprio nell’ordine in cui Amy e Sheldon applicano e tolgono lucchetti alla scatola.

I monaci e le bacchette

In un monastero buddhista abitavano quattro saggi monaci. Essi venivano riforniti di cibo dai loro apprendisti, e spendevano le loro giornate seduti a un tavolo meditando con piccole pause per mangiare.
I monaci avevano davanti a loro ciotole con riso sufficiente per tutta la giornata, e quattro bacchette, ognuna condivisa con il vicino.

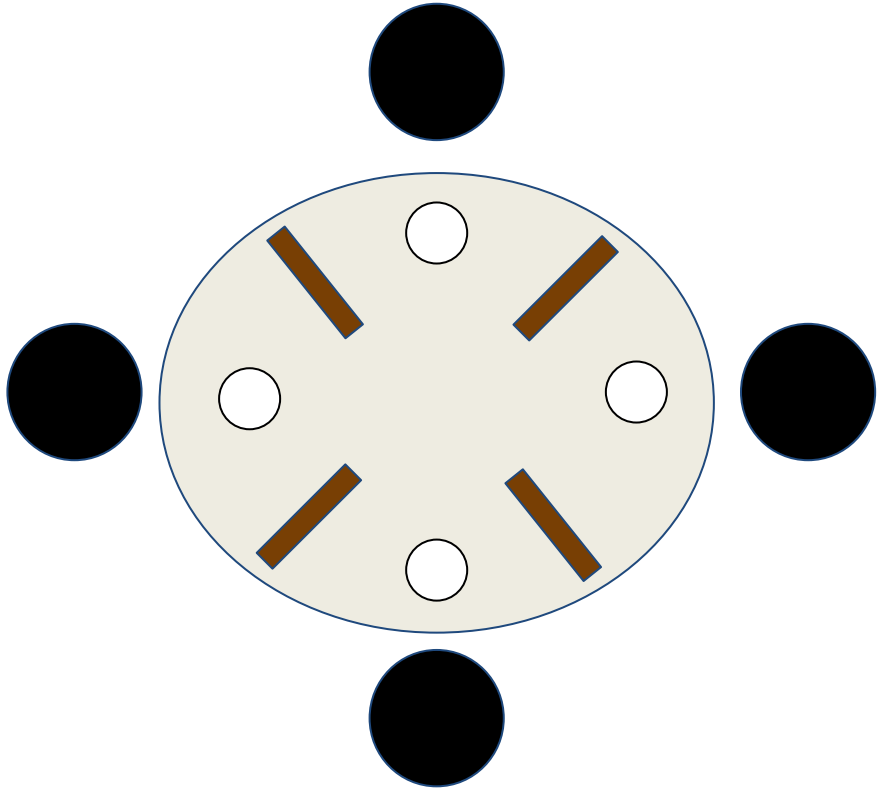
(Continua)



I monaci e le bacchette

Ogni volta che ha fame, un monaco fa questo: (1) prende la bacchetta alla sua destra. Se è occupata dall'altro monaco con cui la condivide, aspetta educatamente il suo turno. (2) similmente, prende la bacchetta alla sua sinistra appena è libera (3) mangia un poco di riso (4) posa le bacchette e torna a meditare.

A un certo punto tutti i monaci rimangono bloccati fermi immobili ad aspettare. Cosa è successo? Come possiamo evitarlo?



Soluzione: deadlock

Tutti i monaci hanno deciso di mettersi a mangiare nello stesso momento: hanno afferrato tutti contemporaneamente la bacchetta alla loro destra, e stanno pazientemente aspettando che si liberi la bacchetta alla loro sinistra.

Una possibile soluzione per evitare questo problema è cambiare l'ordine in cui scelgono le bacchette: è sufficiente che uno dei quattro monaci parta dalla bacchetta alla sua sinistra (ordinamento delle risorse).

Cosa c’entra con l’informatica?

Questo puzzle simula il comportamento dei sistemi con molti processori e accessi condivisi: quando più agenti eseguono istruzioni contemporaneamente e devono accedere a delle risorse (dischi, database, memoria…), bisogna fare attenzione al rischio che si blocchino tutti in attesa (deadlock).

Il puzzle, dovuto a E. Dijkstra, è noto come “the dining philosophers problem”.

L’intero mancante

Vi viene data una lista di 999 numeri, contenente (in un ordine qualsiasi) tutti i numeri interi da 1 a 1000 **tranne uno**. Come è possibile trovare l’intero mancante nel modo più efficiente possibile?

Se sapete programmare, pensate a quante istruzioni dovete eseguire nel vostro **linguaggio di programmazione** preferito.

- 344
- 253
- 2
- 76
- 467
- 1000
- 94
- 11
- …

Tanti controlli

Scorro i numeri della lista uno per uno e controllo se c'è l'1.
Poi li scorro di nuovo, e controllo se c'è il 2. E così via!

Problema: queste sono circa **1.000.000 istruzioni (n^2)**, visto che rischio di dover guardare 999 volte ognuna delle 999 posizioni.

Tanta carta

Mi scrivo su un altro foglio di carta i numeri da 1 a 1000. Faccio passare la lista, e tutte le volte che incontro un numero lo segno con un pallino sul mio foglio di carta.

Ora devo far passare la lista una volta sola, però ho bisogno di un foglio di carta aggiuntivo con lo spazio per scrivere 1000 numeri. Nel gergo degli informatici, mi serve **spazio O(n)**.

Soluzione: tante somme!

Calcolo la somma S di tutti i numeri nella lista. Se i numeri ci fossero tutti, questa somma farebbe 1000*1001/2 = 500500. Quindi l'intero mancante è 500500 - S.

Questo metodo esegue **circa 1000 istruzioni (O(n))** e richiede spazio aggiuntivo solo per tenere in memoria un numero di 6 cifre... a patto di saper fare le somme velocemente, come un computer. (In realtà ci basta tener traccia delle ultime 3 cifre di questa somma).

Cosa c’entra con l’informatica?

La complessità in **tempo** e **spazio** di un algoritmo è un fattore importante da tenere presente quando si programma.

Le tecniche per **lavorare con liste** in modo efficiente sono un tema che si affronta molte volte in un corso di laurea in informatica.