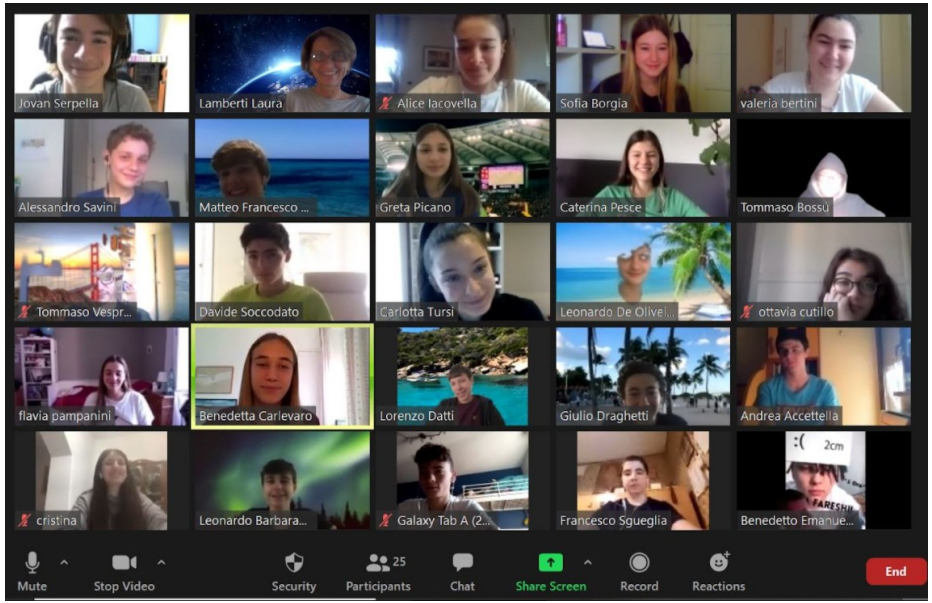
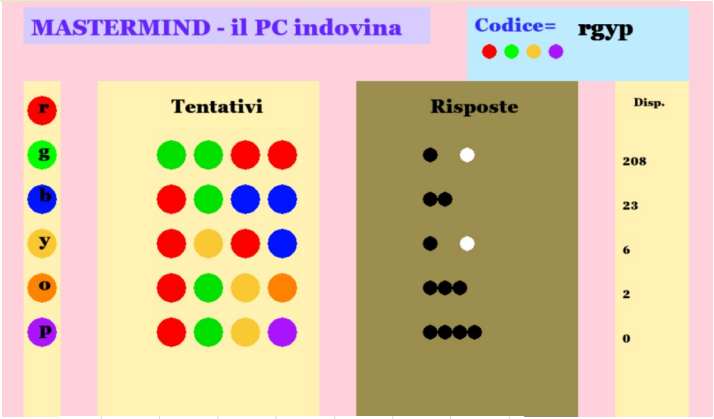
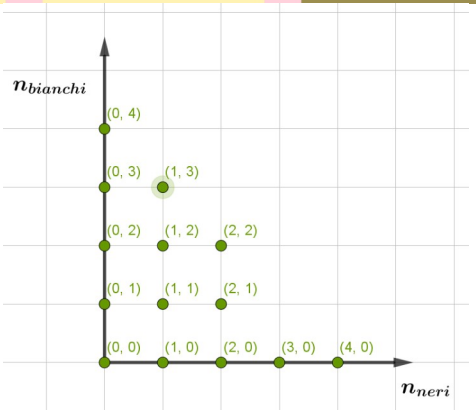
# Dalla costruzione di un videogioco agli algoritmi decisionali di scelta




1/

# Dalla costruzione di un videogioco agli algoritmi decisionali di scelta

Il progetto didattico è stato svolto tra marzo e maggio 2021 con una classe **seconda liceo scientifico**

E' stato svolto in parte in **DAD** e in parte in **presenza**. Non è semplice quantificare le ore che si sono rese necessarie perché molto del lavoro è stato svolto a casa dai ragazzi. Le sole ore curricolari spese sono state **almeno 12**.

Alcuni studenti hanno lavorato in **gruppi di lavoro**, anche nei periodi in cui la didattica era solo a distanza, collaborando via Zoom e suddividendosi i compiti

L'attività è stata di tipo **laboratoriale:** l'implementazione informatica è stata progettata nelle ore curricolari in presenza ed è stata codificata soprattutto nei momenti in cui le lezioni erano a distanza.

La fase di progettazione del codice ha previsto una fase di discussione e analisi
che è avvenuta in buona parte in presenza.

# Dalla costruzione di un videogioco agli algoritmi decisionali di scelta

Il progetto è  nato a conclusione del **percorso di informatica del primo biennio**.

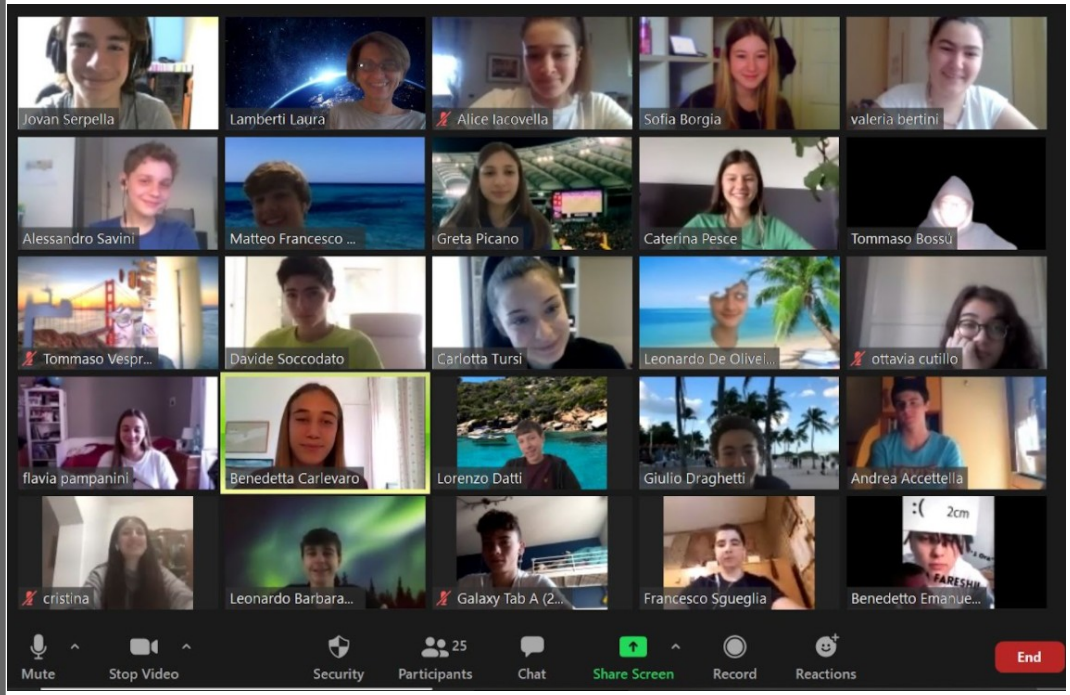
La classe a cui è stato rivolto è un'ottima classe che ha mostrato nei due anni un certo gradimento per i lavori di informatica.

Nei due anni del biennio gli alunni hanno imparato ad utilizzare le istruzioni di lettura-scrittura, scelta condizionale e i cicli; hanno familiarizzato con alcune strutture dati e con le funzioni.
E' stato sempre utilizzato il linguaggio **Python**.
I codici scritti sono stati spesso di supporto agli argomenti inerenti il programma di matematica.

Inoltre la classe aveva prodotto già due videogiochi: **Snake** e **The game of life** utilizzando per entrambi la libreria Turtle.

La scelta didattica di terminare l'anno scolastico con la progettazione e codifica di **Mastermind** è stata espressamente richiesta dagli alunni, decisamente provati dal lungo periodo di DAD che avevano affrontato.

# Dalla costruzione di un videogioco agli algoritmi decisionali di scelta

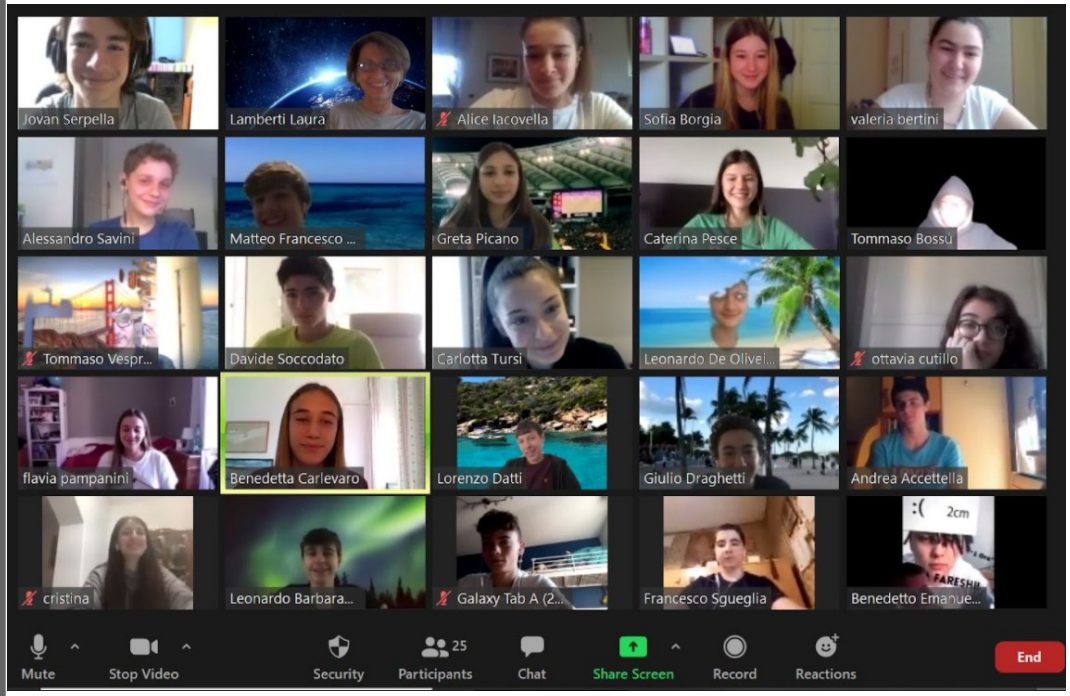



In classe,
gli studenti hanno prima analizzato e progettato un videogioco in cui la scelta iniziale è effettuata dal pc;

la codifica in Python ha fornito spunti per introdurre molti elementi di informatica
e rafforzare concetti di matematica.

Poi, in un secondo momento è stata implementata la versione reverse, in cui i ruoli tra giocatore e pc sono invertiti.

# Dalla costruzione di un videogioco agli algoritmi decisionali di scelta




Il **gioco** è uno **strumento didattico** che aiuta a coinvolgere e appassionare i ragazzi.
In ogni gioco ci sono regole che vanno comprese e applicate e poi c'è un obiettivo da raggiungere.

Gli studenti, catturati dal gioco, sono naturalmente interessati a riflettere su regole e procedure e sono disposti a mettere in moto le proprie competenze.

Il gioco Mastermind, in particolare, consente di introdurre **elementi di insiemistica**, **logica** e **combinatoria** e ne favorisce la sperimentazione;
La ricerca delle strategie vincenti permette di parlare di **algoritmi di scelta decisionali**.

# Mastermind: le regole del gioco

Mastermind è stato inventato da Mordechai Meirovitz nel **1970** e
commercializzato in tutto il mondo dalla Invicta Plastics.
Ha goduto di un'ottima fama durante gli anni settanta, diventando uno dei giochi più venduti.
In realtà è un gioco piuttosto antico poiché è una lieve modifica dell'originario Bulls and Cows.

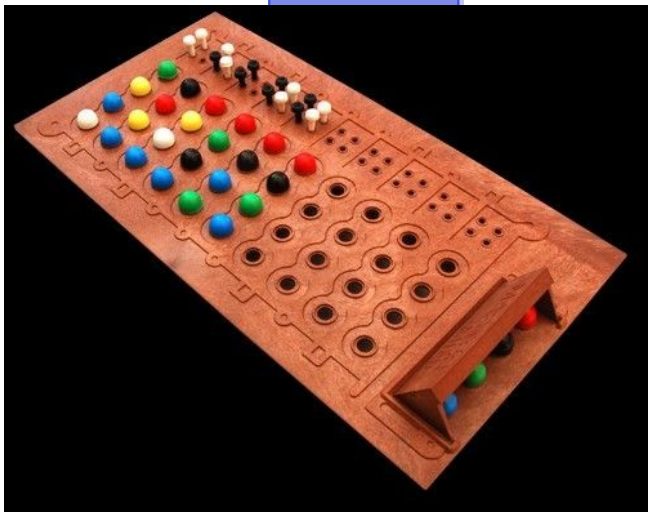
Si gioca essenzialmente tra due giocatori: un **codificatore (code-maker)**
che sceglie un **codice** che mantiene **segreto**,
mentre l'altro giocatore, il **decodificatore (code-breaker)**
deve indovinarlo eseguendo al massimo un certo numero di tentativi.


Il **codice segreto** è composto da una **sequenza di numeri o colori**

(nella versione commerciale e in quella qui studiata si utilizzano i colori,
ma nulla proibisce di associare ad ogni colore una particolare cifra
e quindi trasformare il codice in una sequenza numerica).

# Mastermind: le regole del gioco

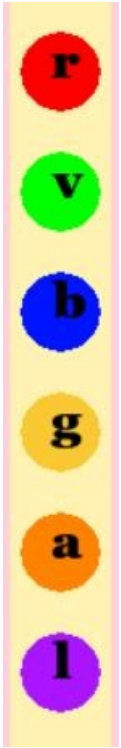
La versione commerciale del gioco ha una tabella forata su cui disporre dei chiodini colorati per elencare i tentativi svolti dal decodificatore.



A fianco di ciascun tentativo,
il giocatore che ha selezionato il codice segreto
annota una risposta
con dei **chiodini bianchi** e **neri,**
fornendo indizi sulla correttezza del tentativo.

Si vince se si indovina il codice segreto in un **numero di mosse inferiore a quello stabilito** come massimo (nella tabella fornita nella versione in scatola ci sono 10 linee forate e quindi si presuppone che 10 sia il numero massimo dei tentativi),

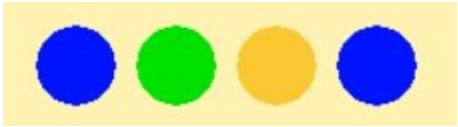
# Mastermind: le regole del gioco




Nella versione standard del gioco il codice è formato da **4** elementi, ciascuno dei quali selezionato da un insieme di **6** elementi diversi. Sono ammesse le ripetizioni

Per esempio:

**CODICE SEGRETO**



# Mastermind: le regole del gioco

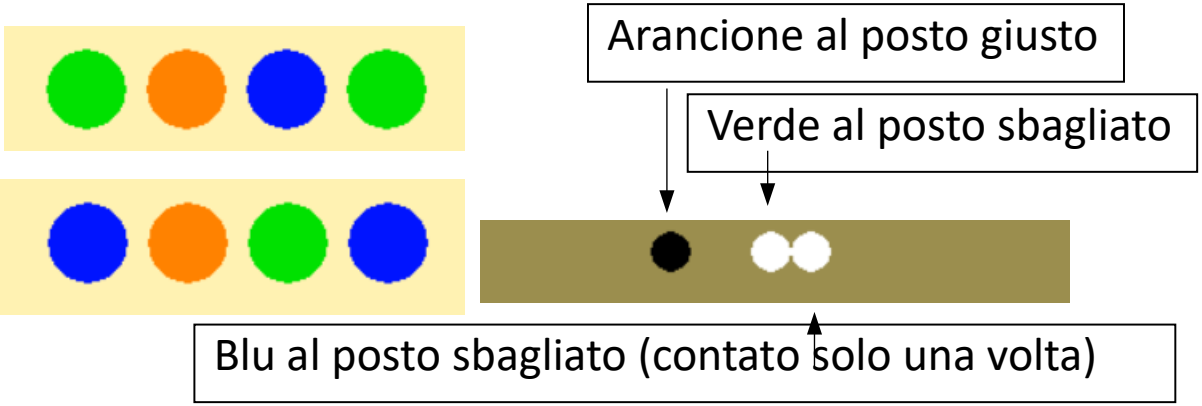
Ad ogni tentativo del giocatore che deve indovinare il codice viene corrisposta una risposta formata da una **sequenza di pallini neri e bianchi**.

Per ogni colore corretto al posto corretto nel tentativo, si inserisce nella risposta un pallino nero.

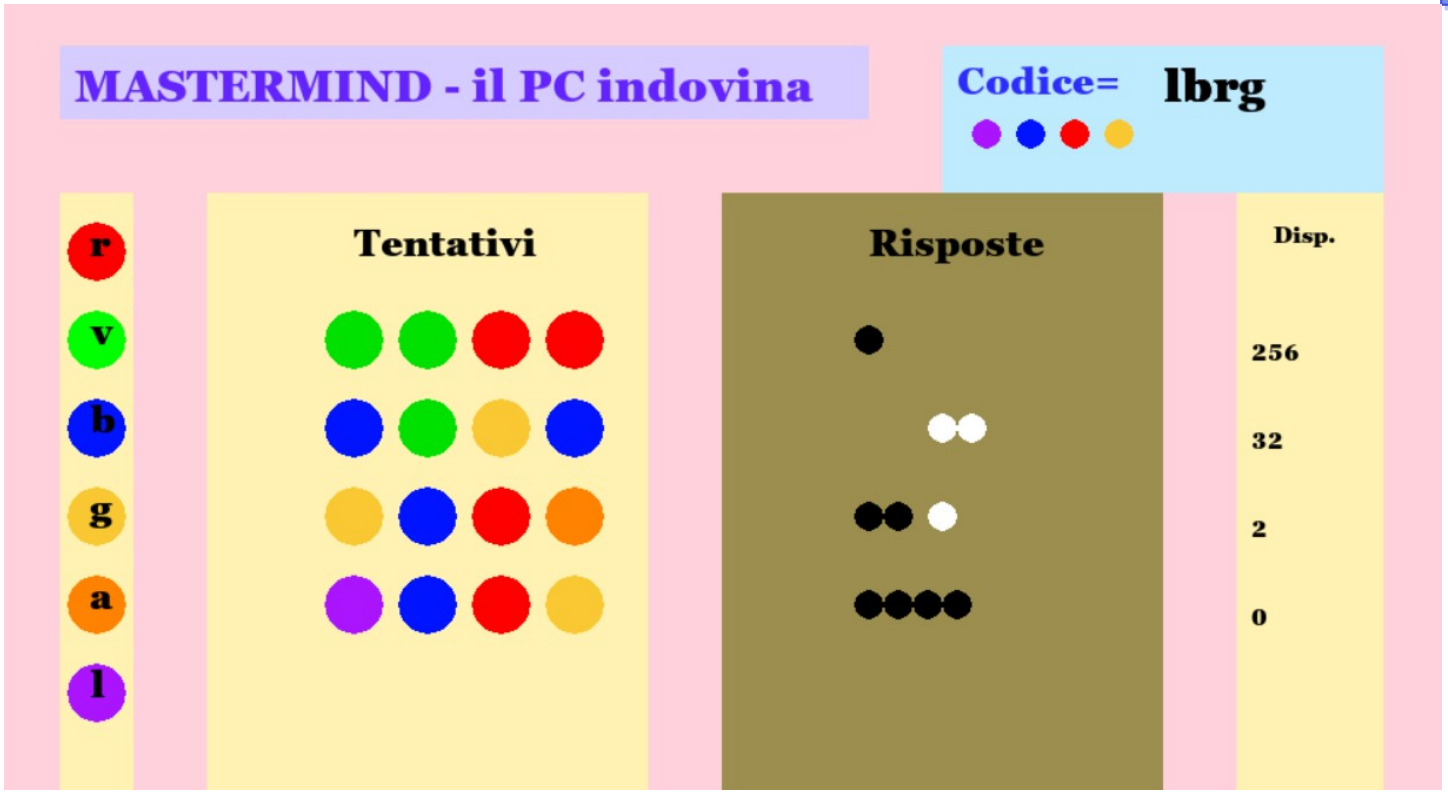
I pallini bianchi vengono utilizzati per comunicare la presenza di un elemento del colore giusto ma che non si trova al posto corretto (eliminando di volta in volta gli elementi già considerati).

**CODICE SEGRETO**

**TENTATIVO**



# Mastermind: la nostra versione “reverse”

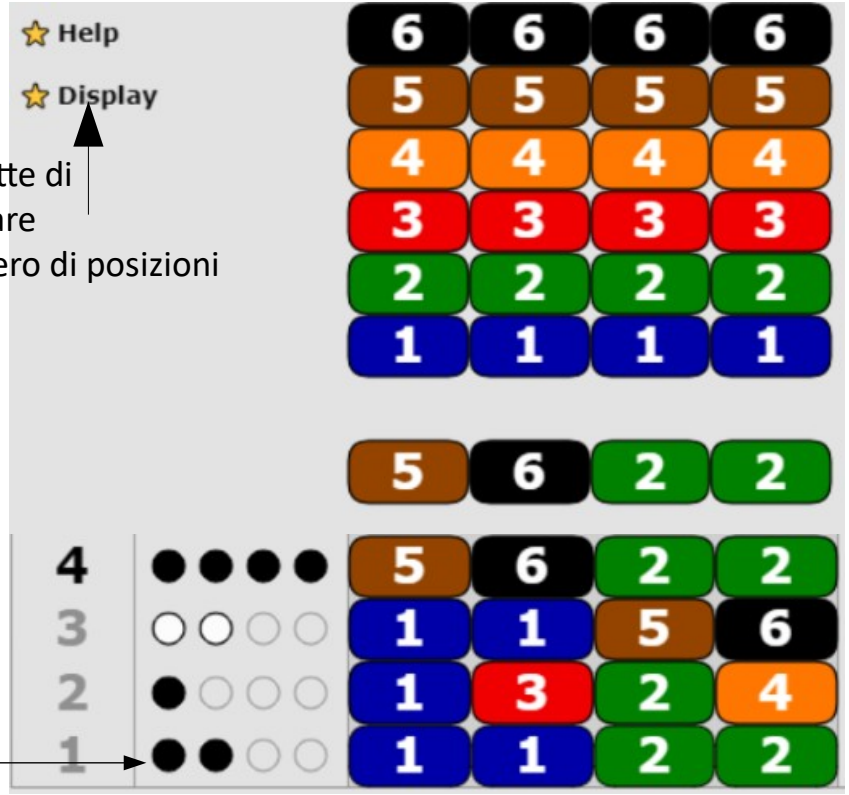


# Mastermind: una versione online

https://bit.ly/3orLOmc

# Mastermind: un esempio commentato

https://bit.ly/3orLOmc




Tasti da utilizzare per inserire ciascun tentativo

CODICE SEGRETO

Tentativi (dal basso verso l'alto)

Risposta al tentativo

# Mastermind: un esempio commentato

https://bit.ly/3orLOmc


**CODICE SEGRETO**
**scelto dal calcolatore**
**NON visibile da chi gioca**

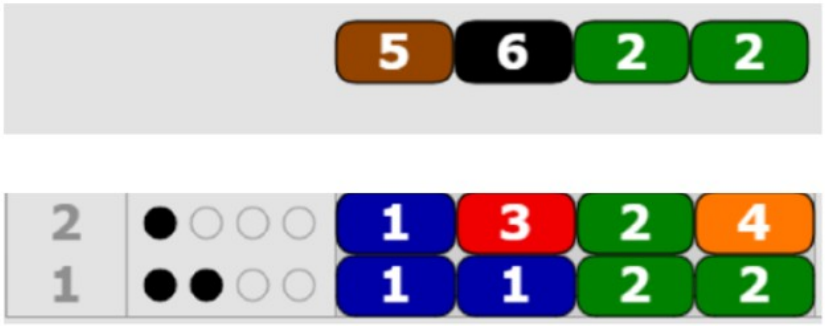

Tentativo

Due colori sono al posto giusto: noi sappiamo che sono i due verdi.
Il decodificatore può dedurre che i due elementi giusti possono essere

- un blu e un verde: in tal caso, blu e verde compaiono solo una volta nel codice, perché la risposta non contiene pallini bianchi
- due verdi: in tal caso, il blu non compare nel codice
- due blu: in tal caso, il verde non compare nel codice

# Mastermind: un esempio commentato – secondo tentativo

https://bit.ly/3orLOmc




**CODICE SEGRETO**
**scelto dal calcolatore**

Il decodificatore mantiene, nel secondo tentativo, un verde e un blu nella posizione precedente e inserisce rosso e arancio. Ora un solo elemento è corretto: noi sappiamo che è il verde.

Il decodificatore deduce che il codice contiene uno solo tra i colori blu, rosso, arancio e verde (perché non ci sono pallini bianchi) e quel colore è al posto giusto. Può pensare che l'elemento corretto sia uno qualsiasi tra i quattro proposti.

# Mastermind: un esempio commentato – terzo tentativo

https://bit.ly/3orLOmc

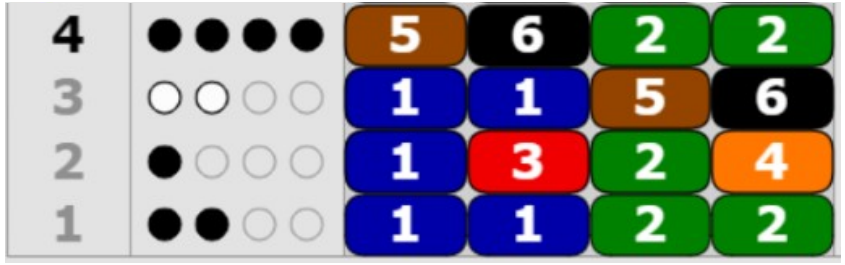

**CODICE SEGRETO**
**scelto dal calcolatore**

Il decodificatore inserisce nuovamente, nel terzo tentativo, due blu in prima e seconda posizione (sperando che corrispondano agli elementi corretti del primo tentativo) e inserisce i colori rimanenti, marrone e nero. Ora solo due colori sono corretti, ma stanno nella posizione sbagliata: noi sappiamo che sono il marrone e il nero.
Il decodificatore deduce che il codice non contiene il blu (perché per il secondo tentativo ci sarebbe dovuto essere un pallino nero), e dunque nel primo tentativo i verdi sono corretti.
Inoltre il codice contiene marrone e nero (ma in un'altra posizione)

# Mastermind: un esempio commentato – quarto tentativo

https://bit.ly/3orLOmc

**CODICE SEGRETO**
**scelto dal calcolatore**




Ora il decodificatore ha solo due possibilità:

- deve inserire il verde in terza e quarta posizione
- Deve inserire marrone e nero in prima o seconda posizione, ma non sa con quale ordine.
-

È fortunato e indovina al quarto tentativo

# Mastermind: un esempio da completare

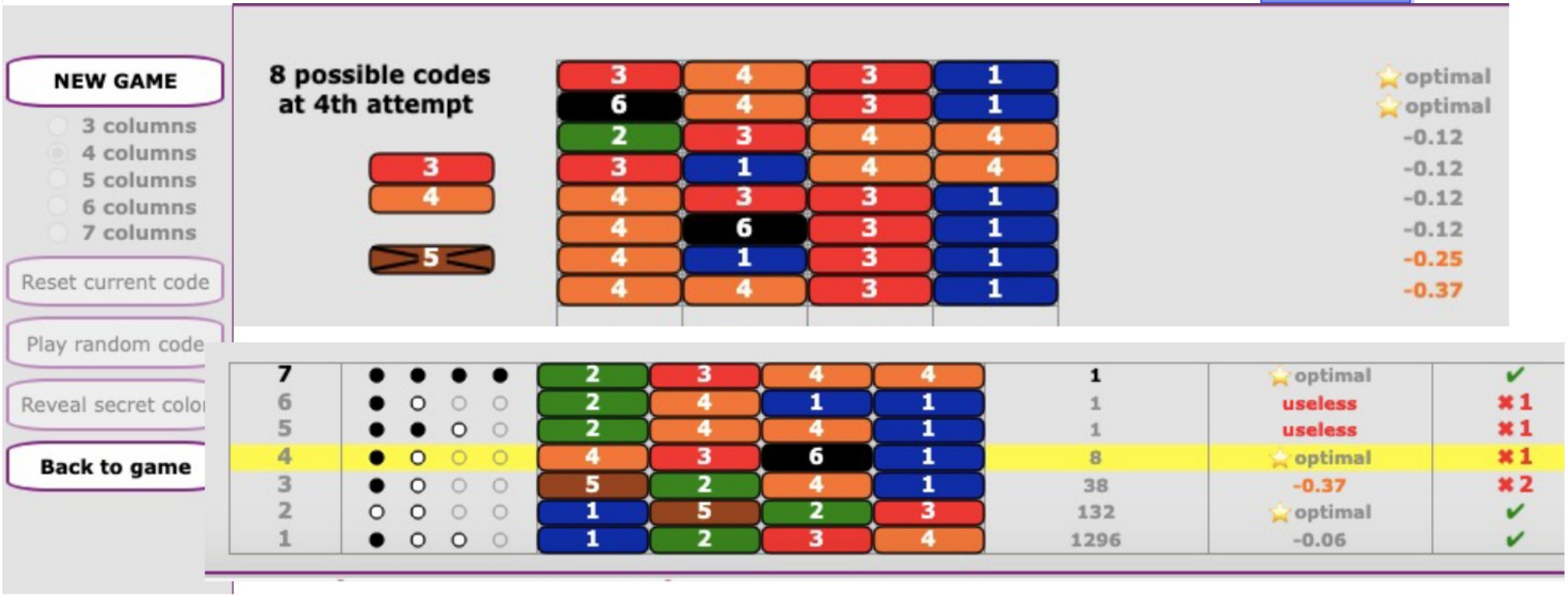

CODICE SEGRETO

Prova!

Ad ogni passo, il sistema segnala quanti codici sono compatibili con le risposte ricevute

Inoltre, a gioco terminato, fornisce l'elenco dei codici compatibili con le risposte

https://bit.ly/3orLOmc


NEW GAME
3 columns
4 columns
5 columns
6 columns
7 columns
Reset current code
Play random code
Reveal secret colo
Back to game
8 possible codes
at 4th attempt
3
4
5
3 4 3 1
6 4 3 1
2 3 4 4
3 1 4 4
4 3 3 1
4 6 3 1
4 1 3 1
4 4 3 1
optimal
optimal
-0.12
-0.12
-0.12
-0.12
-0.25
-0.37
7 2 3 4 4 1 optimal ✔
6 2 4 1 1 1 useless ✖1
5 2 4 4 1 1 useless ✖1
4 4 3 6 1 8 optimal ✖1
3 5 2 4 1 38 -0.37 ✖2
2 1 5 2 3 132 optimal ✔
1 1 2 3 4 1296 -0.06 ✔

# Mastermind: quale strategia?

Come tenere memoria e utilizzare al meglio le risposte ottenute?

Quale strategia utilizzare?

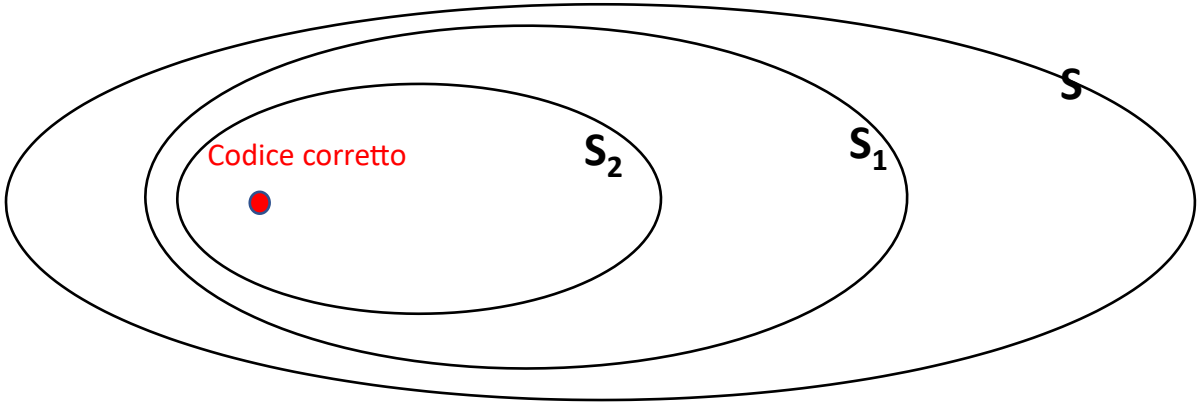
Supponiamo che il codice sia stato fissato e un tentativo sia stato sperimentato.
L'insieme dei codici compatibili con la risposta ottenuta è un sottoinsieme $\mathbf{S_1}$ dell'insieme **S** di tutti i codici : tale sottoinsieme contiene la risposta corretta.


Tutti i codici della lunghezza fissata

# Mastermind: quale strategia?

Provato un ulteriore tentativo, è sufficiente **cercare in $S_1$ i codici compatibili con la seconda risposta**, selezionando un sottoinsieme $S_2$ di codici compatibili con tutte le risposte.

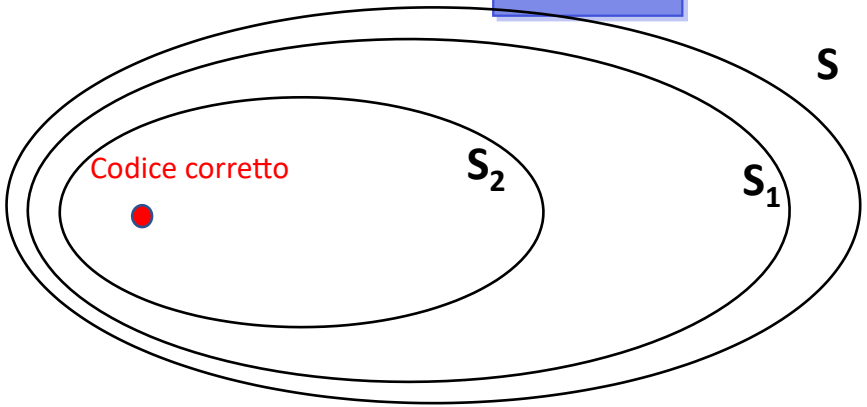
Se il tentativo è scelto in modo adeguato, il sottoinsieme $S_2$ è propriamente contenuto in $S_1$ .
**Un modo per assicurare che il tentativo permette di ridurre l'insieme dei codici compatibili è quello di selezionare il tentativo tra gli elementi dell'insieme dei codici compatibili**.



# Mastermind: come valutare l'efficacia di un tentativo?

Ad ogni passo, come selezionare il tentativo più efficace?

La scelta va compiuta PRIMA di conoscere la risposta....




Idea percorribile: scegliere il tentativo in modo tale che, *anche se si verifica il "peggiore dei casi"*, siamo sicuri di ridurre maggiormente il numero di codici compatibili.

Per stabilire quale sia il "peggiore dei casi" e valutare quale sia la riduzione sul numero dei codici compatibili devo formalizzare meglio il lavoro.

# Una minima formalizzazione

Dal punto di vista matematico, il codice segreto è rappresentato da una sequenza

$$x_1\ x_2\ x_3\ x_4$$

Analogamente, ogni tentativo può essere rappresentato da

$$y_1\ y_2\ y_3\ y_4$$

Gli elementi $x_i$ e $y_k$ del codice sono scelti in un insieme di 6 valori distinti

Devo poter prevedere l'elenco delle possibili risposte e calcolare rapidamente la compatibilità

# Una minima formalizzazione

$x_1\ x_2\ x_3\ x_4$ codice

$y_1\ y_2\ y_3\ y_4$ tentativo

La risposta è una coppia ordinata ($n_{neri}$, $n_{bianchi}$) ove

il primo elemento $n_{neri}$ è il numero di pallini neri, cioè il numero delle concordanze $x_i = y_i$

il secondo elemento $n_{bianchi}$ è il numero di pallini bianchi, cioè il numero di coppie (i,k) con $i \neq k$ ma $x_i = y_k$

calcolati senza tenere conto degli indici già utilizzati nel conteggio dei pallini neri o

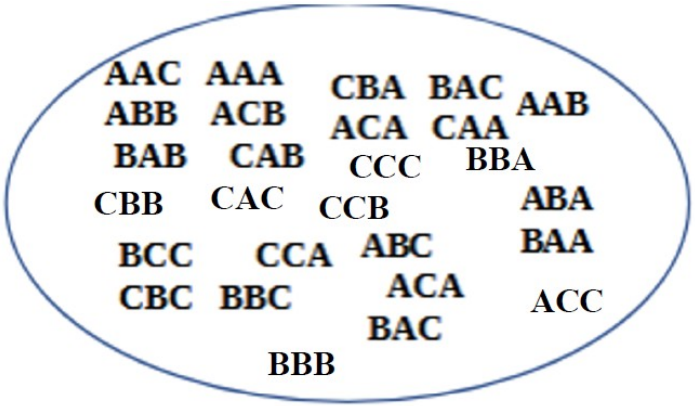
dei già assegnati pallini bianchi

# Elementi di combinatoria

Analizziamo un **caso semplice**, che viene presentato come lavoro agli studenti.
Supponiamo che il codice segreto sia composto da **tre elementi**,
quindi ci siano solo **tre posizioni,** e sia possibile utilizzare soltanto **tre colori**
per costruire la sequenza del codice segreto. Le ripetizioni sono comunque permesse.

Per semplicità i tre colori saranno
identificati dalle lettere A, B e C.

Le disposizioni possibili sono $3^3 = 27$ .



# Una minima formalizzazione – le risposte

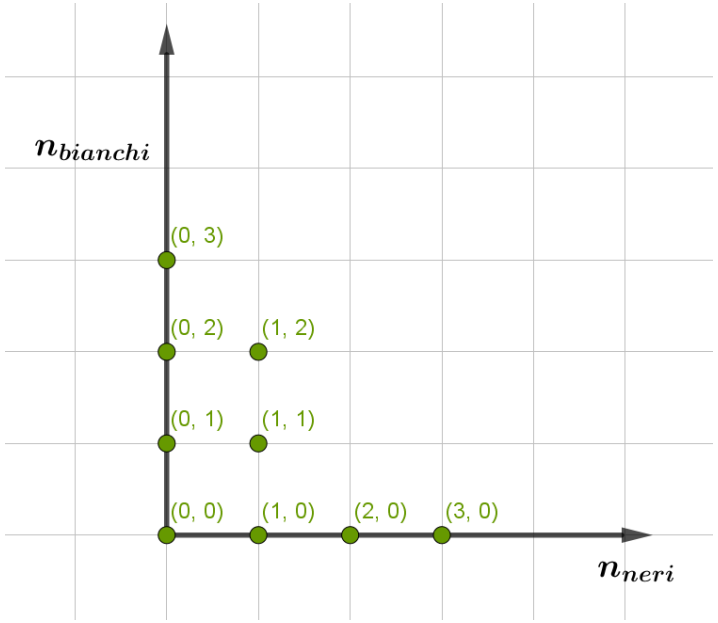
Al tentativo giocato il pc corrisponde una risposta in termini di pallini neri e pallini bianchi.
La risposta quindi è costituita da una coppia ordinata di numeri

$(n_{neri}, n_{bianchi})$

Quante sono le possibili risposte?

La somma degli indici è limitata

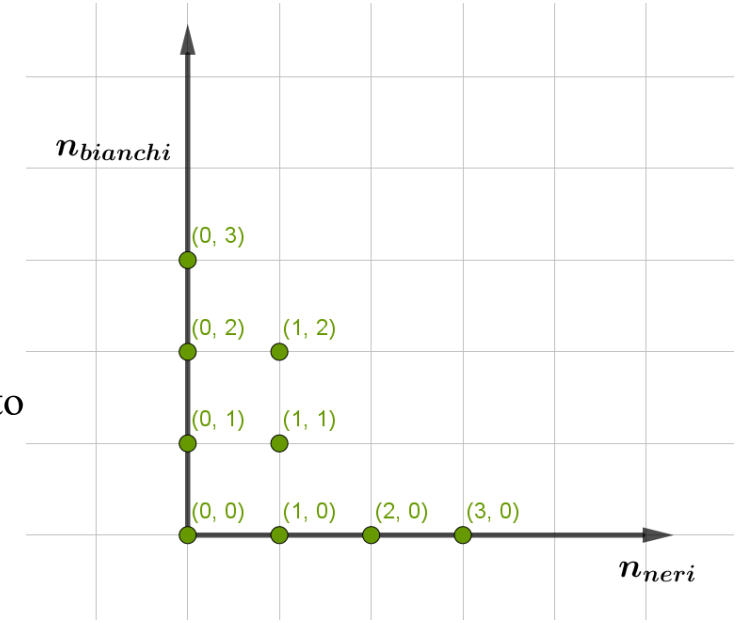
$$n_{bianchi} + n_{neri} \leq 3$$




In figura sono rappresentate le 9 possibili risposte
Nel caso in cui si giochi con 3 colori e 3 posizioni

# Una minima formalizzazione – le risposte

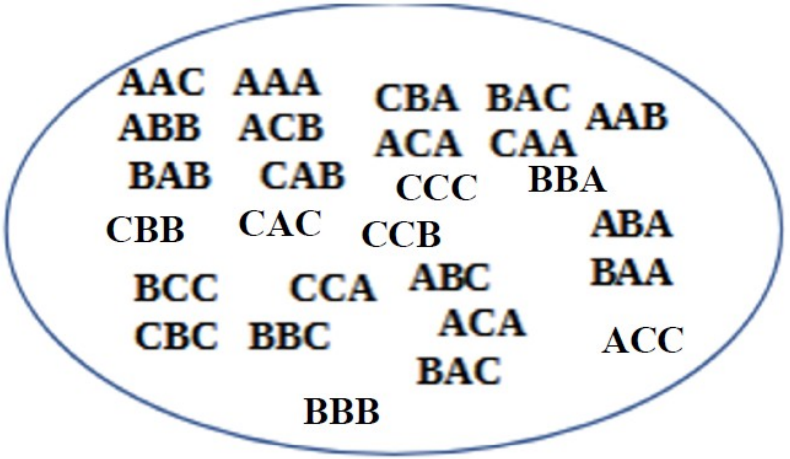
1. Esistono 9 possibili risposte ad un tentativo giocato.
   l'insieme delle disposizioni può essere partizionato
   al massimo in 9 sottoinsiemi disgiunti tra loro.
2. Ogni tentativo scelto tra le disposizioni produce
   una partizione differente.
3. Ciascun sottoinsieme di una stessa partizione è formato
   da un certo numero di disposizioni che hanno
   in comune la stessa risposta
   rispetto al tentativo che ha generato la partizione




In figura sono rappresentate le 9 possibili risposte
Nel caso in cui si giochi con 3 colori e 3 posizioni

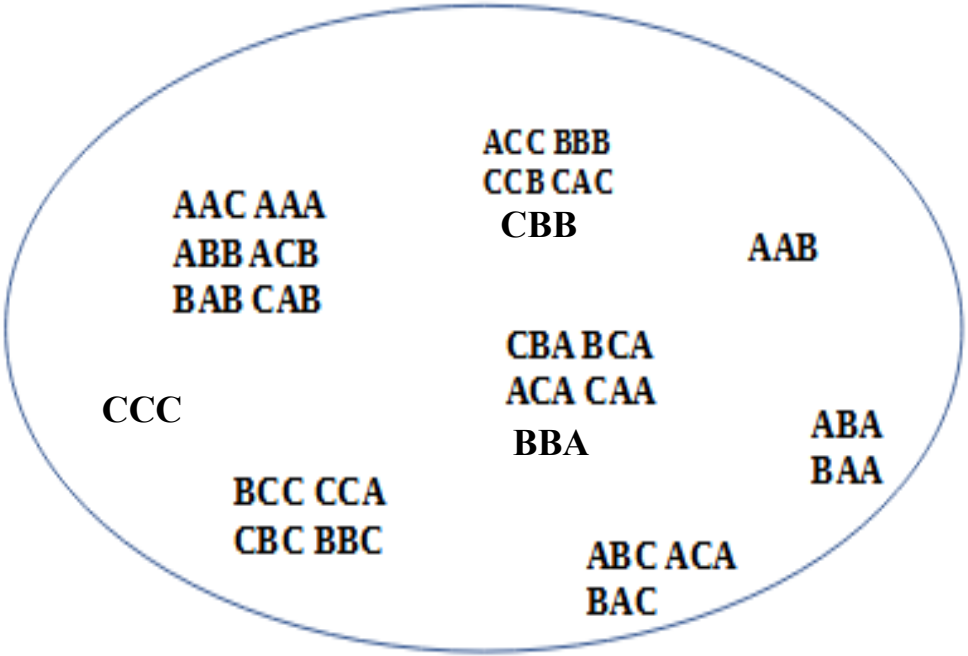
# Elementi di combinatoria

Per studiare un caso di esempio possiamo supporre che il codice segreto scelto sia **BCA**.

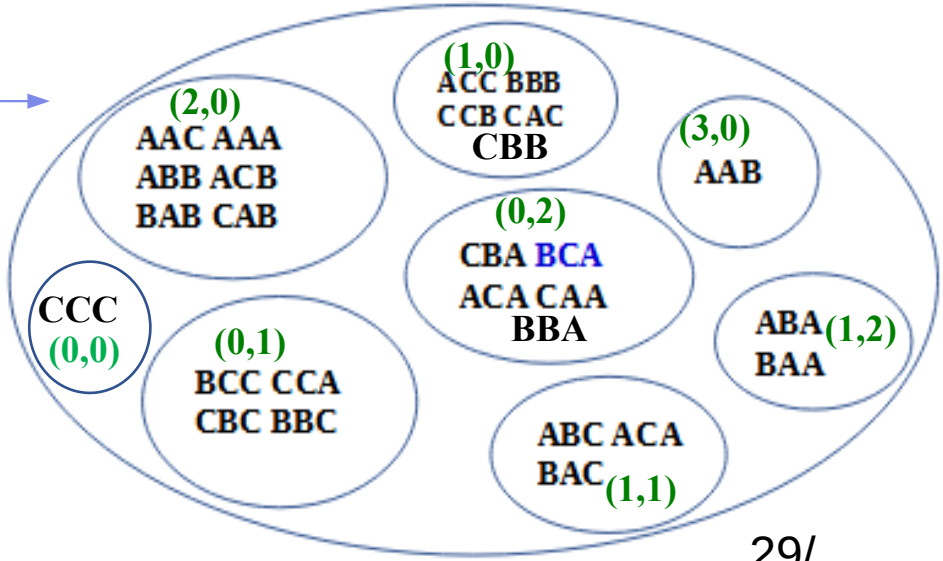



Supponiamo per esempio di tentare il codice **AAB**.
La scelta **AAB** per il tentativo fornisce una partizione dell'insieme delle possibili disposizioni in sottoinsiemi, ciascuno dei quali contiene elementi che sono compatibili con AAB relativamente ad una (e una sola) delle possibili risposte .
Ogni scelta di un tentativo individua una specifica partizione.

# Costruzione dei sottoinsiemi




Partizione prodotta
Scegliendo come
Primo tentativo **AAB**



# Elementi di combinatoria

1. Le operazioni che portano alla scoperta del codice segreto
si basano tutte sulla costruzione e individuazione di **sottoinsiemi**
**dell'insieme delle disposizioni possibili.**

2. La definizione di questi sottoinsiemi determina una **partizione** dell'insieme iniziale
e dipende dal campione di partenza e dalla risposta ricevuta.

3. Il tentativo seguente permette una **nuova classificazione** degli elementi
che erano compatibili con la risposta seguente.

4. L'iterazione del procedimento così progettato
porta certamente alla individuazione del **codice segreto in un numero finito di passaggi**.

# Costruzione  dei sottoinsiemi

**Insieme delle disposizioni**

**Terne**: AAA, BBB, CCC

**Coppie:** AAB, AAC, ABA, ACA, BAA, CAA, ABB, BAB, BBA,
BBC, BCB, CBB, ACC, CAC, CCA, BCC, CBC, CCB

**No ripetizioni:** ABC, BCA, CBA, BAC, ACB, CAB

# Costruzione dei sottoinsiemi

**Insieme delle disposizioni**

terne: AAA, BBB , CCC,
coppie: AAB, AAC, ABA, ACA, BAA, CAA, ABB, BAB, BBA, CCA, CAC, CAC, BBC, CCB, CBC, BCB, CCB, CBB
no ripetizioni: ABC, BCA, CBA, CAB, BAC, ACB

Primo Tentativo **AAB**

| Risposta | Num disposizioni consentite | disposizioni |
|---|---|---|
| (3,0) | 1 | AAB |
| (2,0) | 6 | AAC, AAA, ABB, ACB, BAB, CAB |
| (1,0) | 5 | ACC, BBB, CCB, CAC, CBB |
| (1,1) | 3 | ABC, ACA, BAC |
| (1,2) | 2 | ABA, BAA |
| (0,2) | 5 | CBA, BBA BCA, ACA, CAA |
| (0,3) | no | no |
| (0,1) | 4 | BCC, CCA, CBC, BBC |
| (2,1) | no | no |
| (0,0) | 1 | CCC |
| Totale: | 27 | |

# Costruzione dei sottoinsiemi

**Insieme delle disposizioni**

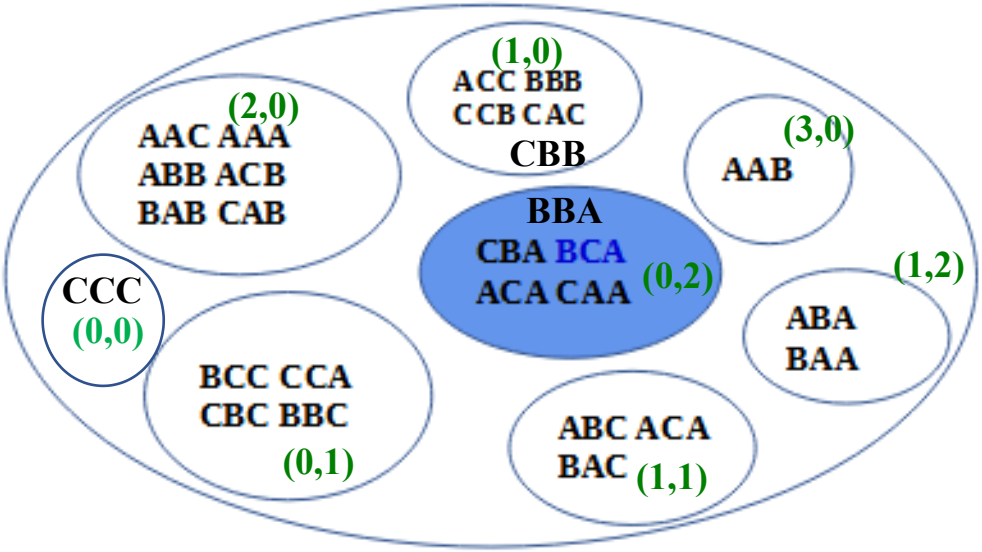
terne: AAA, BBB , CCC,
coppie: AAB, AAC, ABA, ACA, BAA, CAA, ABB, BAB, BBA, CCA, CAC, CAC, BBC, CCB, CBC, BCB, CCB, CBB
no ripetizioni: ABC, BCA, CBA, CAB, BAC, ACB

| Risposta | Primo Tentativo **ABC** | | Primo Tentativo **AAB** | | Primo Tentativo **AAA** | |
|---|---|---|---|---|---|---|
| | Num disposizioni consentite | disposizioni | Num disposizioni consentite | disposizioni | Num disposizioni consentite | disposizioni |
| (3,0) | 1 | ABC | 1 | AAB | 1 | AAA |
| (2,0) | 6 | ABA, ABB, AAC, ABC, CBC, BBC | 6 | AAC, AAA, ABB, ACB, BAB, CAB, | 6 | AAB, AAC, ABA, ACA, CAA, BAA, |
| (1,0) | 3 | AAA, BBB, CCC | 5 | ACC, BBB, CCB, CAC, CBB | 12 | ABB, ABC, ACB, ACC, BAB, CAB, BAC, CCA, CBA, CAC, BCA, BBA |
| (1,1) | 6 | ACA, ABB, BBA, CBB, CAC, BCC | 3 | ABC, ACA, BAC | no | no |
| (1,2) | 3 | ACB, CBA, BAC | 2 | ABA, BAA | no | no |
| (0,2) | 6 | BAB, BAA, CCA, CAA, CCB, BCB | 5 | CBA, BCA, ACA, CAA, BBA | no | no |
| (0,3) | 2 | CAB, CBA | no | no | no | no |
| (0,1) | no | no | 4 | BCC, CCA, CBC, BBC | no | no |
| (2,1) | no | no | no | no | no | no |
| (0,0) | no | no | 1 | CCC | 8 | BBB, CCC, BCB. CBC, CCB, BBC, CBC, CBB |
| Totale: | 27 | | 27 | | 27 | |

# Costruzione dei sottoinsiemi

Partizione prodotta
Scegliendo come
Primo tentativo **AAB**

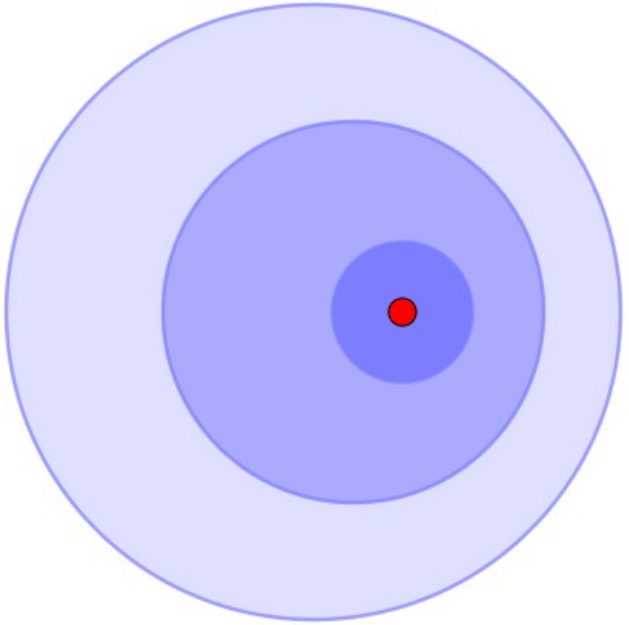
La risposta **(0,2)** seleziona
un sottoinsieme



# Costruzione dei sottoinsiemi

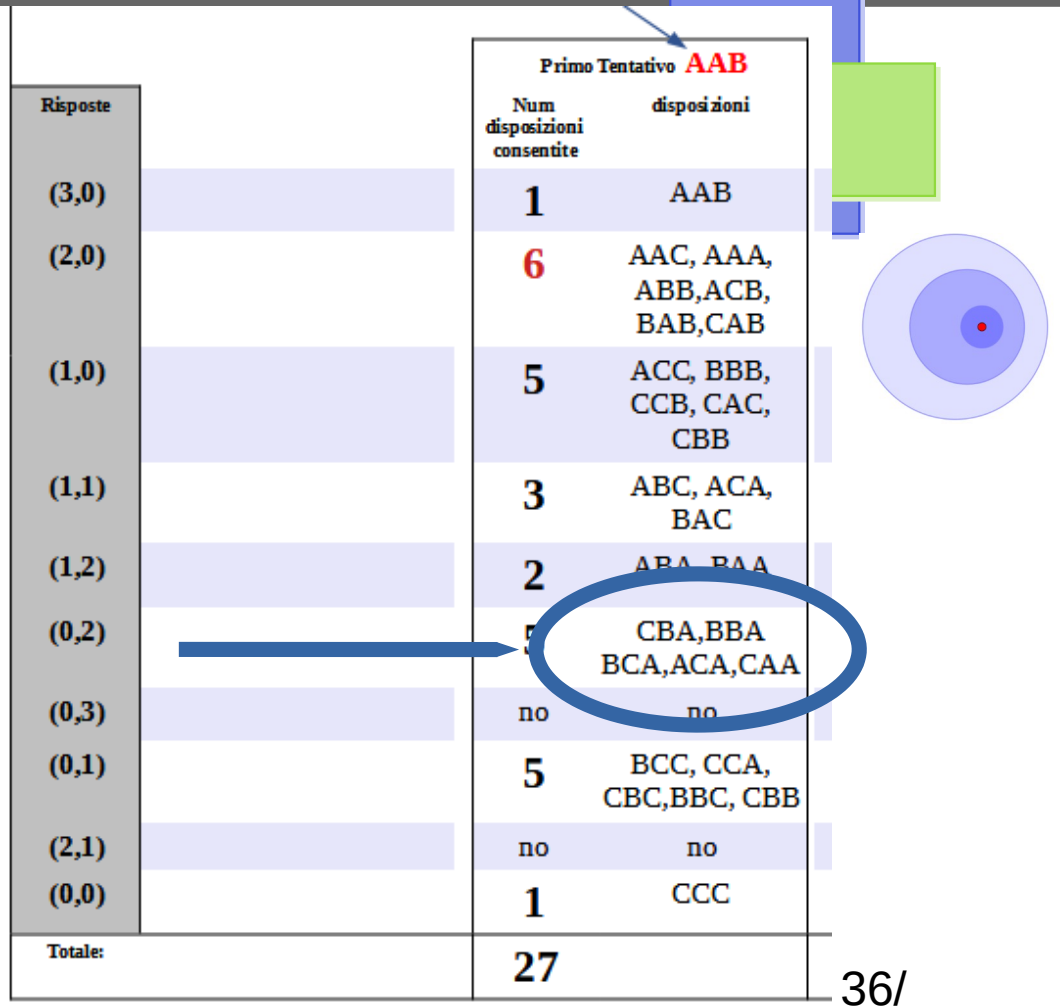
**Itero** il procedimento
Scegliendo un nuovo tentativo
tra gli elementi del sottoinsieme
generato dalla prima risposta

Il codice segreto appartiene anche
al nuovo sottoinsieme



# Costruzione di sottoinsiemi

Supponiamo che
al primo tentativo
si abbia la risposta **(0,2)**



Primo Tentativo **AAB**

| Risposte | Num disposizioni consentite | disposizioni |
|---|---|---|
| (3,0) | 1 | AAB |
| (2,0) | 6 | AAC, AAA, ABB,ACB, BAB,CAB |
| (1,0) | 5 | ACC, BBB, CCB, CAC, CBB |
| (1,1) | 3 | ABC, ACA, BAC |
| (1,2) | 2 | ABA, BAA |
| (0,2) | 5 | CBA,BBA BCA,ACA,CAA |
| (0,3) | no | no |
| (0,1) | 5 | BCC, CCA, CBC,BBC, CBB |
| (2,1) | no | no |
| (0,0) | 1 | CCC |
| Totale: | 27 | |

# Costruzione di sottoinsiemi

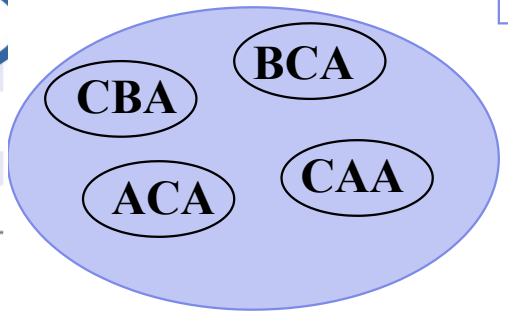
| Risposte | Primo Tentativo AAB<br>Num disposizioni consentite | disposizioni |
|---|---|---|
| (3,0) | 1 | AAB |
| (2,0) | 6 | AAC, AAA, ABB,ACB, BAB,CAB |
| (1,0) | 5 | ACC, BBB, CCB, CAC, CBB |
| (1,1) | 3 | ABC, ACA, BAC |
| (1,2) | 2 | ABA, BAA |
| (0,2) | 5 | CBA,BBA BCA,ACA,CAA |
| (0,3) | no | no |
| (0,1) | 5 | BCC, CCA, CBC,BBC, CBB |
| (2,1) | no | no |
| (0,0) | 1 | CCC |
| Totale: | 27 | |

Insieme delle disposizioni rimaste
terne:
coppie: ACA, CAA,
no ripetizioni: BCA, CBA,

| Risposte | Secondo Tentativo BCA<br>Num disposizioni consentite | disposizioni | Secondo Tentativo CAA<br>Num disposizioni consentite | disposizioni |
|---|---|---|---|---|
| (3,0) | 1 | BCA | 1 | CAA |
| (2,0) | 1 | ACA | 1 | CBA |
| (1,0) | | | 1 | BCA |
| (1,1) | 1 | CAA | | |
| (1,2) | 1 | CBA | 1 | ACA |
| (0,2) | | | | |
| (0,3) | | | | |
| (0,1) | | | | |
| (2,1) | | | | |
| (0,0) | 4 | | | |

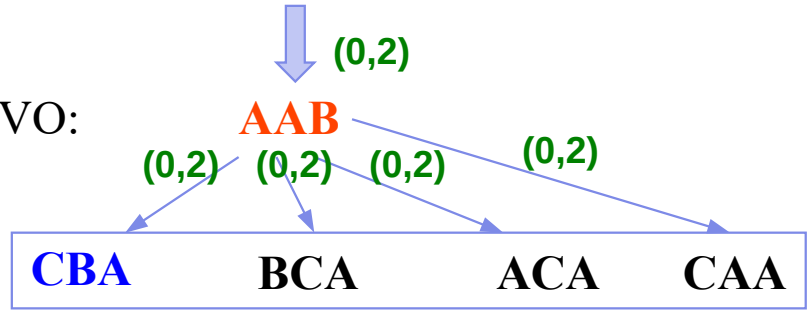
# Simmetria e informazione nei sottoinsiemi

| Risposte | Primo Tentativo AAB: Num disposizioni consentite | disposizioni |
|---|---|---|
| (3,0) | 1 | AAB |
| (2,0) | 6 | AAC, AAA, ABB,ACB, BAB,CAB |
| (1,0) | 5 | ACC, BBB, CCB, CAC, CBB |
| (1,1) | 3 | ABC, ACA, BAC |
| (1,2) | 2 | ABA, BAA |
| (0,2) | 5 | CBA,BBA BCA,ACA,CAA |
| (0,3) | no | no |
| (0,1) | 5 | BCC, CCA, CBC,BBC, CBB |
| (2,1) | no | no |
| (0,0) | 1 | CCC |




CODICE SEGRETO: **CBA**

TENTATIVO:



# Strategia risolutiva- metodo minmax

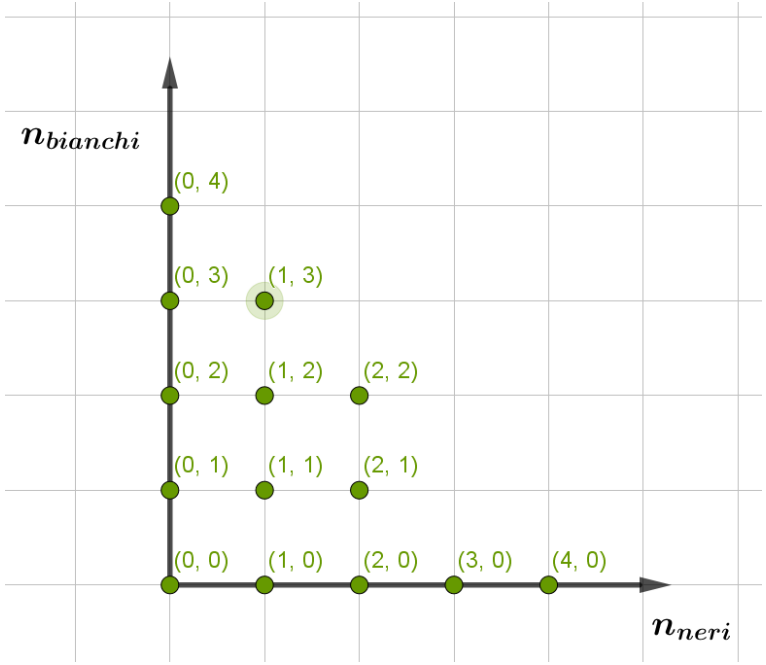
**Per ogni partizione così ottenuta del mio insieme delle scelte**
**valuto la cardinalità K del sottoinsieme più numeroso (caso peggiore)**
**E scelgo quella partizione che produce sottoinsiemi con il minimo valore di K**

In questo modo riduco il numero delle disposizioni possibili e raggiungo
la soluzione con un numero finito di passi, anche se può non coincidere con il minimo numero di passi (ottimo)

# classi di tentativi e partizioni del sistema 4x6

Il numero totale di disposizioni è $6^4 = 1296$

1. Esistono 14 possibili risposte ad un tentativo giocato.
   l'insieme delle disposizioni può essere partizionato
   al massimo in 14 sottoinsiemi disgiunti tra loro.
2. Ogni tentativo scelto tra le disposizioni produce
   una partizione differente.
3. Ciascun sottoinsieme di una stessa partizione è formato
   da un certo numero di disposizioni che hanno
   in comune la stessa risposta
   rispetto al tentativo che ha generato la partizione



# Algoritmo di knuth

Il tentativo iniziale è formato da una **doppia coppia** ( es **AABB**)
In accordo con il metodo min-max

Costruisco i sottoinsiemi corrispondenti alla partizione generata
e itero il procedimento scegliendo per i tentativi successivi
quelli che producono partizioni con le cardinalità minime.

Nel caso di parità vado in ordine lessicografico, ma è ininfluente la scelta

L'algoritmo risolve il problema in al più **5 mosse**.
Il numero medio delle mosse è 4.4761

# Perché scegliere come tentativo iniziale la doppia coppia

Possiamo classificare i tentativi iniziali in base alla loro tipologia:

| | |
|---|---|
| classe 0 : tutti elementi diversi | **ABCD** |
| classe 1: una coppia | **AABC** |
| classe 2 due coppie | **AABB** |
| classe 3: una terna | **AAAB** |
| classe 4 : tutti uguali | **AAAA** |

# Perché scegliere come tentativo iniziale la doppia coppia

classe 0 : tutti elementi diversi **ABCD**

| n \ b | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | 16 | 152 | 312 | 136 | 9 |
| 1 | 108 | 252 | 132 | 8 | 0 |
| 2 | 96 | 48 | 6 | 0 | 0 |
| 3 | 20 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 0 | 0 |

# Perché scegliere come tentativo iniziale la doppia coppia

classe 1: una coppia

**AABC**

| n \ b | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | 81 | 276 | 222 | 44 | 2 |
| 1 | 182 | 230 | 84 | 4 | 0 |
| 2 | 105 | 40 | 5 | 0 | 0 |
| 3 | 20 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 0 | 0 |

# Perché scegliere come tentativo iniziale la doppia coppia

classe 2 due coppie **AABB**

| n \ b | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | 256 | 256 | 96 | 16 | 1 |
| 1 | 256 | 208 | 36 | 0 | 0 |
| 2 | 114 | 32 | 4 | 0 | 0 |
| 3 | 20 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 0 | 0 |

# Perché scegliere come tentativo iniziale la doppia coppia

classe 3: una terna **AAAB**

| n \ b | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | 256 | 308 | 61 | 0 | 0 |
| 1 | 317 | 156 | 27 | 0 | 0 |
| 2 | 123 | 24 | 3 | 0 | 0 |
| 3 | 20 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 0 | 0 |

# Perché scegliere come tentativo iniziale la doppia coppia

classe 4 : tutti uguali **AAAA**

| n \ b | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | 625 | 0 | 0 | 0 | 0 |
| 1 | 500 | 0 | 0 | 0 | 0 |
| 2 | 150 | 0 | 0 | 0 | 0 |
| 3 | 20 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 0 | 0 |

**CASI PEGGIORI**

classe0: 312
classe1: 276
classe2: **256**
classe3: 308
classe4: 625

# La progettazione del codice

**attività didattica in classe**

1. Parte Indovina TU

discussione sulle strutture dati che servono per l'implementazione in python
scrittura delle singole funzioni che si devono interfacciare tra di loro
scrittura del modulo grafico

2. Parte Indovina PC

discussione sulle strategie vincenti
implementazione del codice per il caso Indovina PC

**implementazione in python**
progettazione del codice
strutture dati
costruzione di funzioni, moduli
design dell'interfaccia grafica

# La progettazione del codice

**DI COSA ABBIAMO BISOGNO?** Scelta degli “ingredienti”

1. una struttura dati che contenga l’elenco dei colori
2. una struttura dati che contenga il codice segreto
3. una struttura dati che contenga le disposizioni

4. una sezione di (input) e una struttura dati che contenga l’input
5. una funzione che scelga il codice segreto
6. una funzione che conti le concordanze tra il codice segreto e il tentativo
7. una funzione che elimini le disposizioni non compatibili con le risposte
8. una funzione calcoli la cardinalità massima dei sottoinsiemi

9. un contatore che conti il numero di tentativi effettuati

10. una libreria per l’interfaccia grafica

# La progettazione del codice

**DI COSA ABBIAMO BISOGNO?**

1. una struttura dati che contenga l'elenco dei colori **colours[ ]**
2. una struttura dati che contenga il codice segreto **code[ ]**
3. una struttura dati che contenga le disposizioni **prova[ ]**

4. una sezione di (input) e una struttura dati che contenga l'input **guess[ ]**
5. una funzione che scelga il codice segreto **choise( )**
6. una funzione che conti le concordanze tra il codice segreto e il tentativo **confronta()**
7. una funzione che elimini le disposizioni non compatibili con le risposte **pulisci()**
8. una funzione calcoli la cardinalità massima dei sottoinsiemi **punteggio()**

9. un contatore che conti il numero di tentativi effettuati **g**

10. una libreria per l'interfaccia grafica **pygame**

# La progettazione del codice

...motore del programma è la funzione **confronta**

```
def confronta(guess, code):
        n_neri = 0
        n_bianchi = 0
        for i in range(4):
            if guess[i] == code[i]:
                n_neri += 1    #aumento i chiodini neri
                guess[i] += "PEG!"
                code[i] += "PEG!"
        for i in range(4):
            if guess[i] in code and guess[i] != code[i]:
                n_bianchi += 1    #aumento i chiodini bianchi
                code[code.index(guess[i])] += "PEG!"

        for i in range(4):
            if len(code[i]) > 1:
                code[i] = (code[i])[0]
                guess[i]=(guess[i])[0]


        return n_neri, n_bianchi
```

# La progettazione del codice

...motore del programma è la funzione **pulisci**

```
def pulisci(prova, tentativo, codice):
      #list(prova)
      elimino=[]
      riparto=[]
      n_neri_cod=confronta(tentativo,codice)[0]
      n_bianchi_cod=confronta(tentativo,codice)[1]

      if (n_neri_cod==0 and n_bianchi_cod==0):
          for j in range (0,len(prova)):
            for i in range(4):
              if (prova[j][i] in tentativo):
                  if (prova[j] not in elimino):
                                          elimino.append(prova[j])
                                          break
```

# La progettazione del codice

...motore del programma è la funzione **pulisci**

```
for j in range (0,len(prova)):
        for i in range(4):
                if (confronta(prova[j],tentativo)[0]!= n_neri_cod or confronta(prova[j],tentativo)[
                        if (prova[j] not in elimino):
                                        elimino.append(prova[j])
                                        break

for item in elimino:

  prova.remove(item)
if tentativo in prova:
  prova.remove(tentativo)
#for i in range(len(prova)):
   #     print(i, "prova \n",prova[i] )
elimino=[]
return
```

# La progettazione del codice

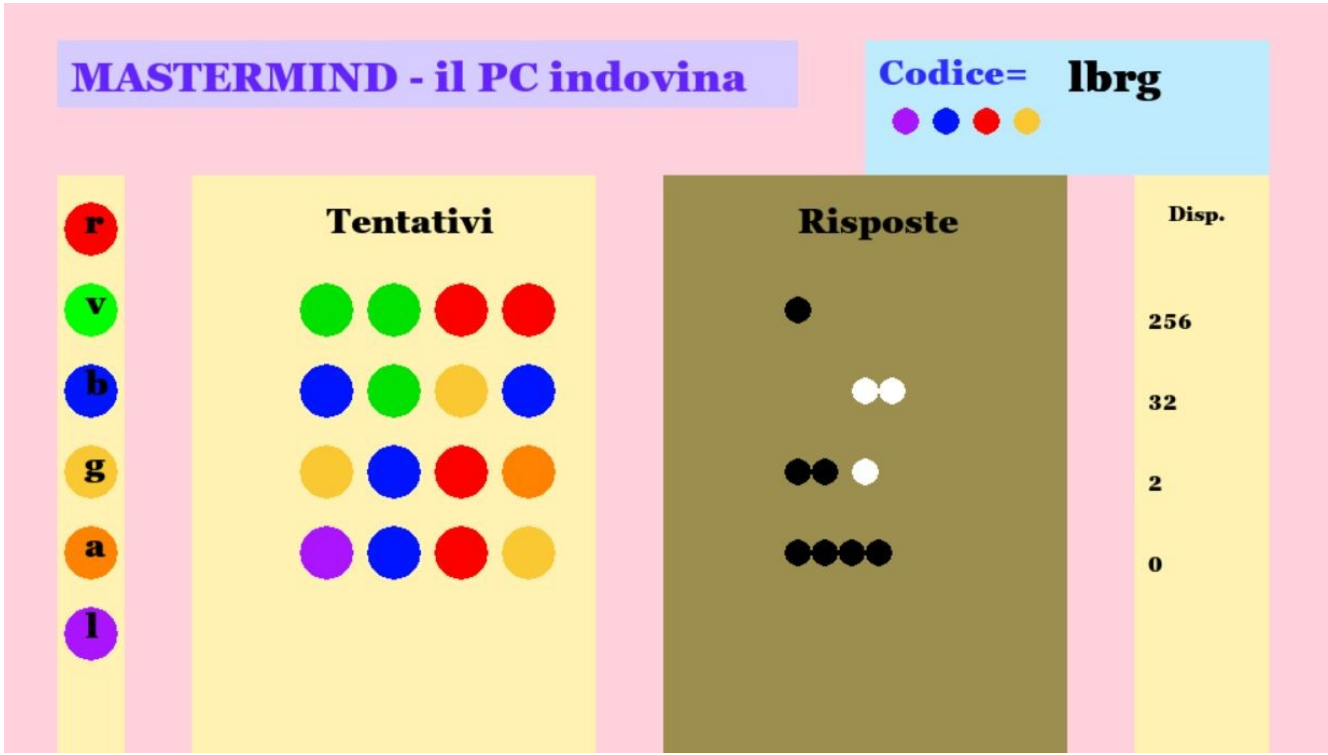
...motore del programma è la funzione **punteggio** sulla base del quale viene scelto il tentativo successiv

```
def punteggio (item, S):

  punti=0
  occurrences=[0]*50
  max_occur=0
  t=0
  for j in range (len(S)):

          n_bianchi=confronta(item,S[j])[0]
          n_neri=confronta(item,S[j])[1]
          t=n_bianchi+10*n_neri
          occurrences[t] +=1
  for t in range(len(occurrences)):
   if (occurrences[t]>max_occur):
          max_occur=occurrences[t]
  return max_occur
```

# La progettazione del codice



# Conclusioni

L’attività proposta è risultata efficace sotto vari aspetti:

**Logico-matematico**: Mastermind permette di allenare la mente migliorandone le abilità logiche, abitua al ragionamento critico, insegna a formulare ipotesi e dedurne conseguenze, costringe a mettere in discussione le ipotesi qualora risultino in disaccordo con i risultati conseguiti; mostra come talora risultati apparentemente negativo possono risultare di grande aiuto
Fornisce un valido strumento per sviluppare competenze di combinatoria e insiemistica

**Argomentativo:** soprattutto nel gioco tra squadre sviluppa la capacità a sostenere le proprie ipotesi nel confronto con gli altri, a ricercare strategie risolutive, favorisce il dibattito

**Coding :** la progettazione di un videogioco ha stimolato gli alunni a migliorare le proprie competenze informatiche

**Socializzazione:** ha permesso di rendere più divertente un periodo molto difficile per i ragazzi

# Grazie per l'attenzione

Francesca Tovena (Università di Tor Vergata) tovena@axp.mat.uniroma2.it
Laura Lamberti (Liceo Scientifico A. Righi, Roma) lamberti.laura@gmail.com

**Bibliografia /Sitografia:**

D.E. Knuth, The computer as Mastermind, J. Recreational Mathematics, vol. 9(1), 1976-77
A.R.Strom, S.Barolo, Using the Game of Mastermind to Teach, Practice and Discuss Scientific Reasoning Skills, PLoS Biology | www.plosbiology.org, January 2011 | Volume 9 | Issue 1 | e1000578 (Open Access)
https://journals.plos.org/plosbiology/article?id=10.1371/journal.pbio.1000578
https://supermastermind.github.io/playonline/game.html