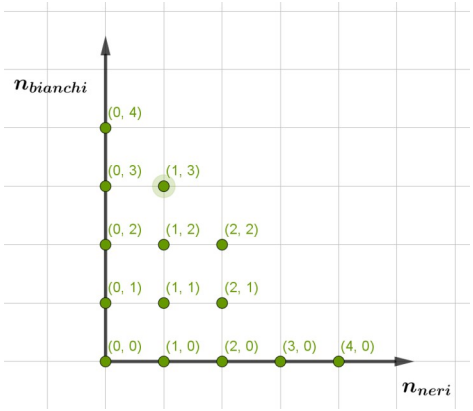
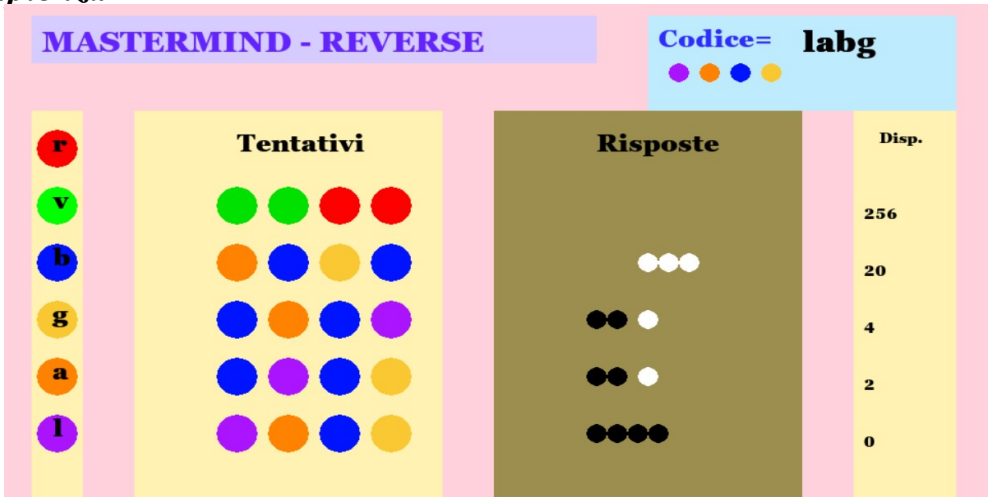
Dalla costruzione di un videogioco agli algoritmi decisionali di scelta

Laura Lamberti, Liceo Augusto Righi, Roma
Francesca Tovena, Università di Roma Tor Vergata

Seminari per il Liceo Matematico, 18 marzo 2022, La Sapienza





Dalla costruzione di un videogioco agli algoritmi decisionali di scelta

Laura Lamberti, Liceo Augusto Righi, Roma
Francesca Tovena, Università di Roma Tor Vergata

Seminari per il Liceo Matematico, 18 marzo 2022, La Sapienza

- Presentazione di un'attività didattica di gamification:
  progettazione e scrittura del codice di un **videogioco**

- Discussione degli obiettivi didattici matematici e informatici:
  Combinatorica, teoria degli insiemi,
  algoritmi decisionali di scelta per la parte di matematica
  Uso delle **strutture dati**, delle funzioni e delle librerie grafiche per la parte di informatica

Dalla costruzione di un videogioco agli algoritmi decisionali di scelta

Gamifying math

Con il termine **gamification**, si intende l'uso del gioco come tecnica didattica: gli elementi di progetto del gioco, quali le regole del gioco, la realizzazione grafica, la componente strategica del gioco diventano strumenti didattici ed educativi.

L'intento è quello di modificare il contesto in cui avviene l'apprendimento in modo da perseguire due obiettivi principali:

gli **obiettivi di apprendimento** che corrispondono ai contenuti didattici e

gli obiettivi relativi all'esperienza di gioco, cioè il divertimento e la soddisfazione del giocatore.

Dalla costruzione di un videogioco agli algoritmi decisionali di scelta

Gamifying math

Le due classi di obiettivi non sono né distinte né disgiunte:

scopo primario della gamification è quello di tipo motivazionale;
la tecnica didattica della gamification è utilizzata per **motivare gli studenti** (Kapp).

Migliorare la motivazione degli studenti ha però un **effetto estremamente positivo sui processi e sui risultati di apprendimento**

Ultimamente, numerose applicazioni della gamification hanno riguardato l'uso degli strumenti digitali, cercando di coinvolgere gli studenti, tramite l'uso di piattaforme o applicazioni con dispositivi digitali come tablet, smartphone o computer .

Il progetto qui presentato e la conseguente realizzazione di un **videogioco** è un esempio di gamification.

Dalla costruzione di un videogioco agli algoritmi decisionali di scelta

Gamifying math

Il **gioco** è da sempre 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.

In relazione agli obiettivi didattici il gioco Mastermind, in particolare, consente di introdurre **elementi di insiemistica**, **logica, combinatoria e teoria dell'informazione** e ne favorisce la sperimentazione;

La ricerca delle strategie vincenti permette di parlare di **algoritmi di scelta decisionali**.

Per quanto riguarda gli obiettivi motivazionali se ne possono elencare moltissimi: *elaborazione di una strategia vincente… stimolazione dei processi di collaborazione tra pari nella progettazione, partecipazione al gioco di squadra, soddisfazione nel raggiungimento di un prodotto finito e funzionante che può essere utilizzato anche da altre persone e così via….*

Dalla costruzione di un videogioco agli algoritmi decisionali di scelta

Attività didattica

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

Attività didattica

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 impegno 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

Attività didattica

Il progetto didattico ha riguardato inizialmente la progettazione e scrittura del codice del **videogioco Mastermind**.

L'attività si è sviluppata in due fasi: la prima parte ha riguardato la scrittura del codice che ha prodotto la versione digitale del gioco standard (il codice segreto è scelto dal pc). Il linguaggio utilizzato è **Python**. Successivamente è stata elaborata l'**interfaccia grafica**.

Queste due fasi, entrambe dedicate alla scrittura del codice hanno permesso il **rafforzamento di molteplici competenze informatiche e matematiche.** Il lavoro è stato suddiviso in gruppi e ciascun alunno ha potuto contribuire per sua misura al confezionamento del videogioco.

Poi, in un secondo momento è stata implementata
la **versione REVERSE**, in cui i ruoli tra giocatore e pc sono invertiti. La progettazione di questo nuovo videogioco ha necessitato l'elaborazione e l'implementazione degli **algoritmi decisionali di scelta**.

https://github.com/lauralamberti/Mastermind-game

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).

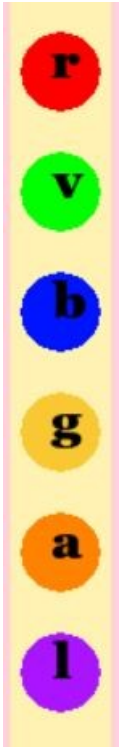
Mastermind: le regole del gioco

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:

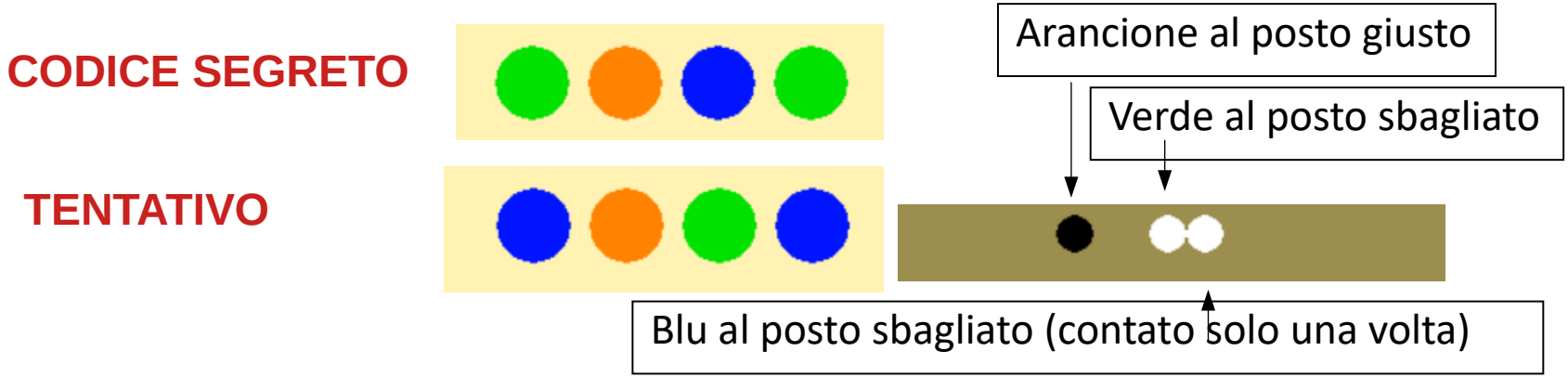
CODICE SEGRETO

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.

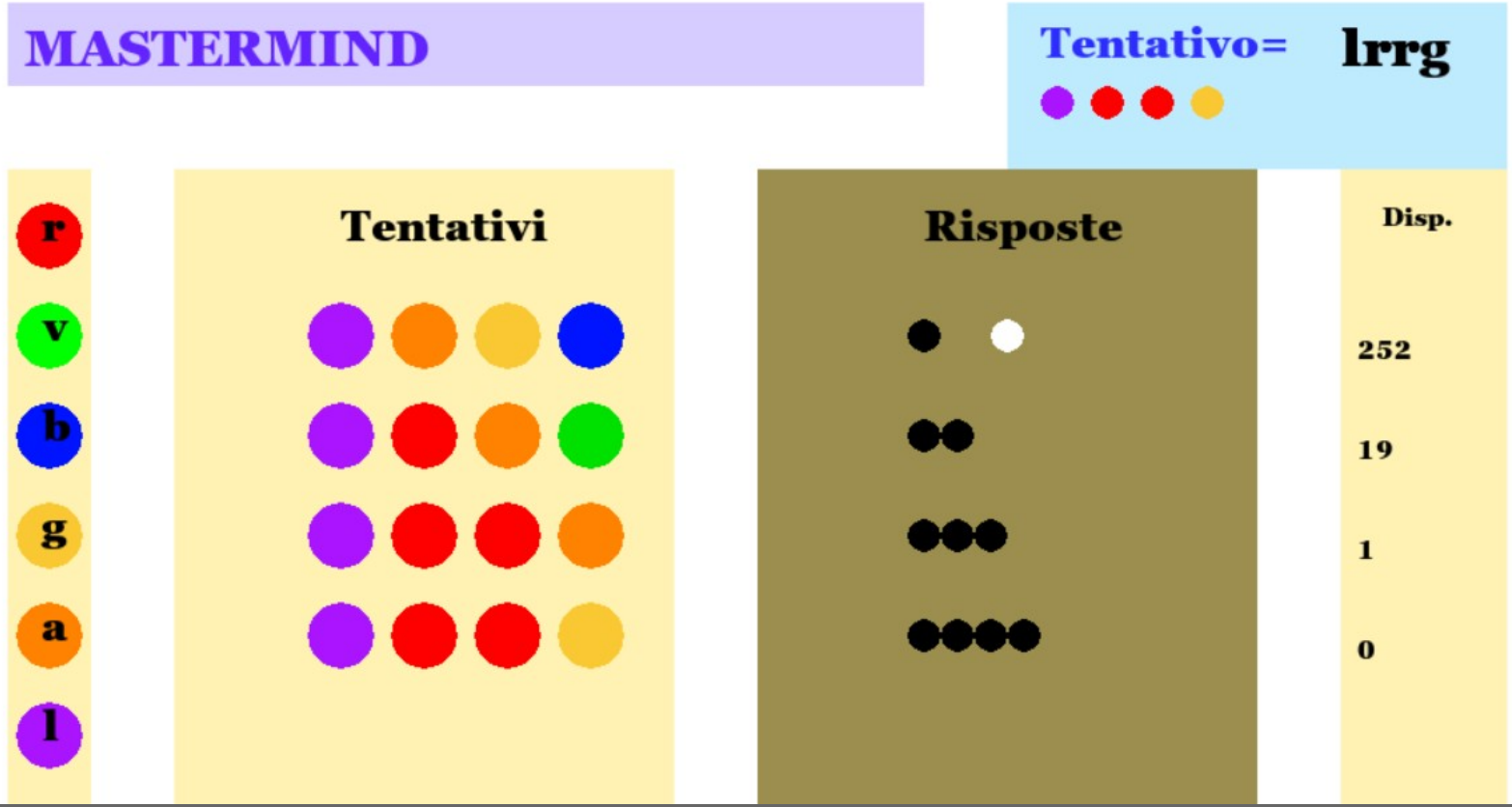
I pallini bianchi vengono utilizzati per comunicare la presenza di un elemento del colore giusto ma che non si trova al posto corretto (eliminando di volta in volta gli elementi già considerati).



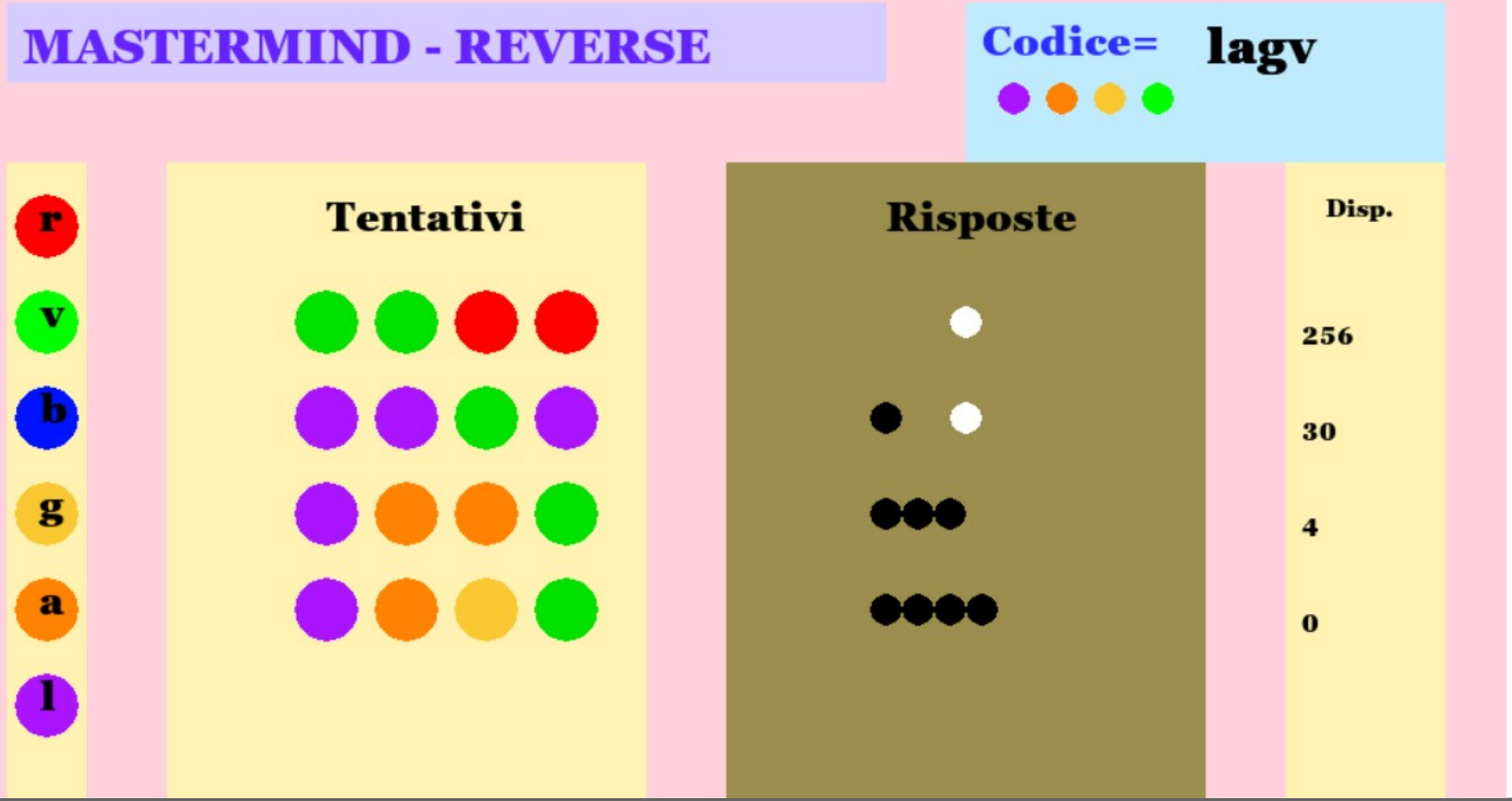
Mastermind: una versione online

https://bit.ly/3orLOmc

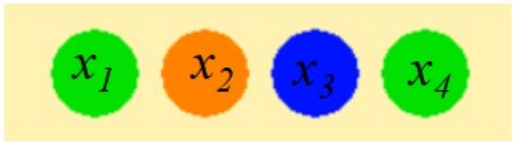
Mastermind: la nostra versione standard



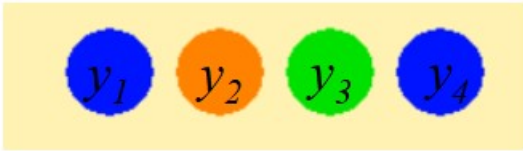
Mastermind: la nostra versione “reverse”



Mastermind: **Formalizzazione del problema**



Il codice segreto può essere rappresentato da una sequenza x_1,x_2,x_3, x_4
mentre il tentativo è y_1,y_2,y_3,y_4



Gli elementi x_i, y_k che compongono il codice e il tentativo sono scelti da un insieme di 6 elementi (i colori).

Il numero dei codici possibili (le disposizioni of 6 elementi di classe 4) è $\mathbf{6^4=1296}$.

Il code-breaker vince quando riesce a mettere i colori giusti al posto giusto.
Ciò significa che il tentativo vincente deve soddisfare la condizione:

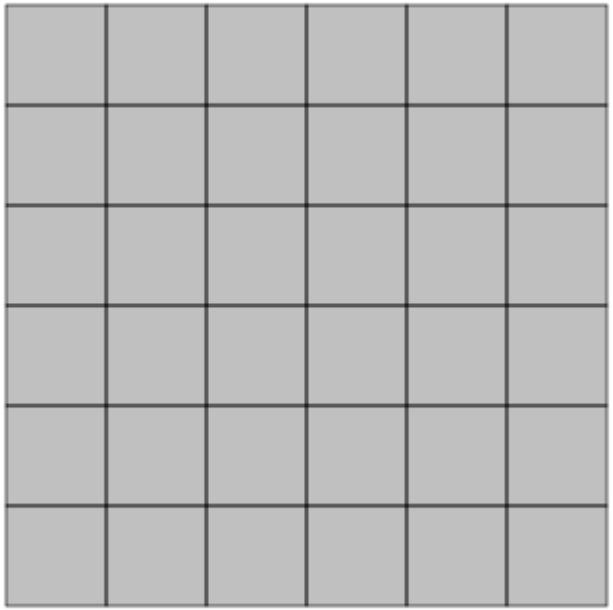
$x_i=y_i$ per i=1...4.

Ad ogni tentativo il giocatore riceve una **risposta (informazione)** che può essere rappresentata da una coppia ordinata

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

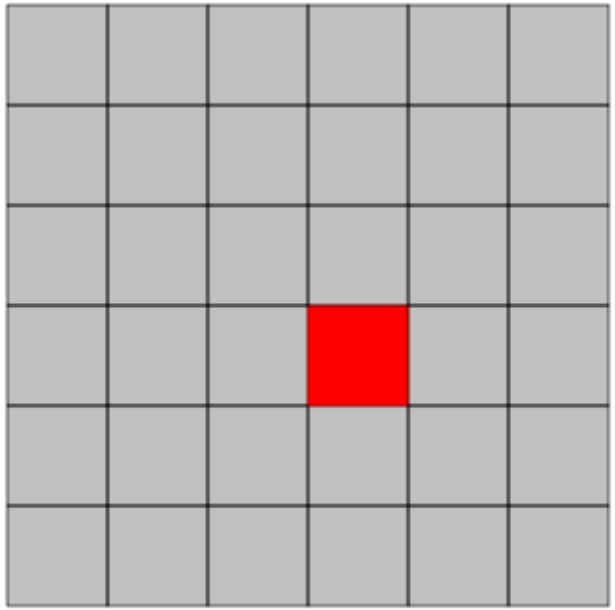
determinata dal confronto tra il tentativo provato e il codice segreto.

Mastermind: **aspetti di combinatorica**



S è l'insieme di tutte le possibili disposizioni.
Ogni disposizione è rappresentata da un quadretto.

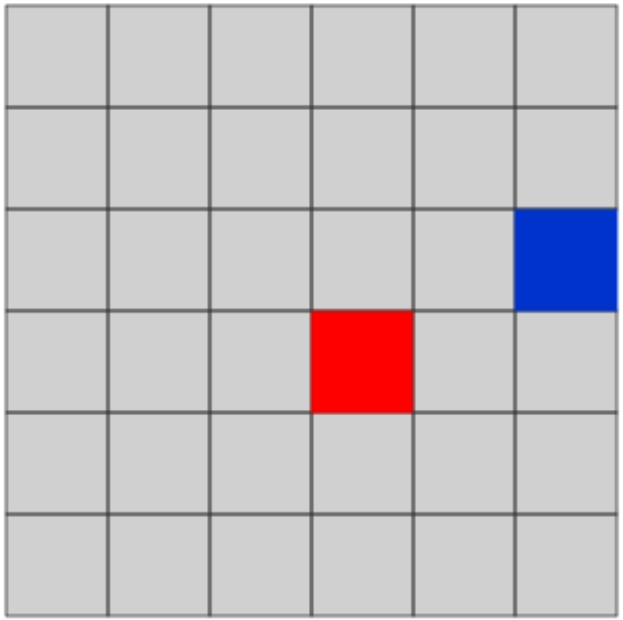
Mastermind: **aspetti di combinatorica**



S è l'insieme di tutte le possibili disposizioni.
Ogni disposizione è rappresentata da un quadretto.

Tra queste disposizioni c'è quella scelta
come **codice segreto**

Mastermind: **aspetti di combinatorica**



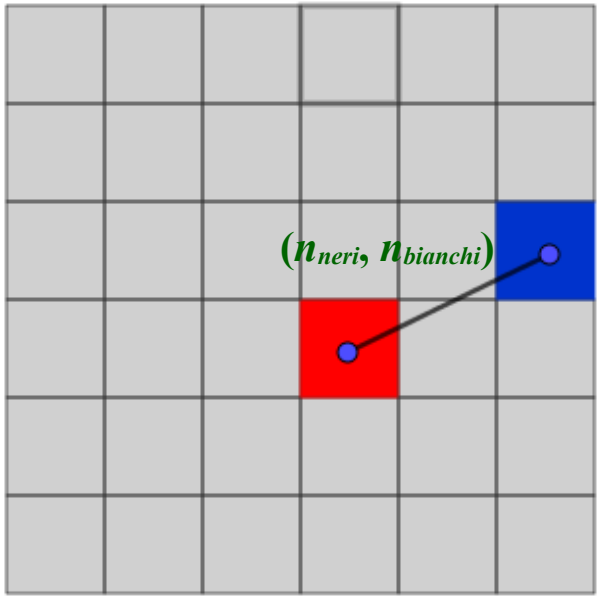
S è l'insieme di tutte le possibili disposizioni.
Ogni disposizione è rappresentata da un quadretto.

Tra queste disposizioni c'è quella scelta
come **codice segreto**

Il **(primo) tentativo** è in generale un'altra disposizione.

A questo tentativo il code-maker fornisce una risposta
$(n_{neri}, n_{bianchi})$ che viene sfruttata per ridurre la cardinalità dell'insieme in cui cercare il codice segreto

Mastermind: **aspetti di combinatorica**



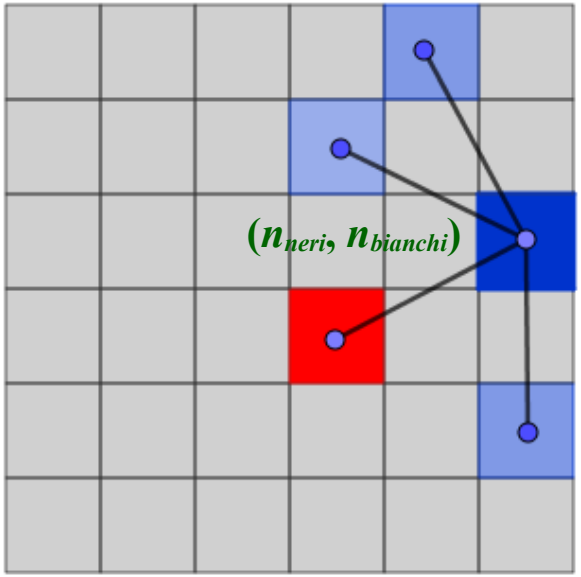
S è l'insieme di tutte le possibili disposizioni.
Ogni disposizione è rappresentata da un quadretto.

Tra queste disposizioni c'è quella scelta
come **codice segreto**

Il **(primo) tentativo** è in generale un'altra disposizione.

A questo tentativo il code-maker fornisce una risposta (informazione) $(n_{neri}, n_{bianchi})$ che permette di **ridurre la cardinalità** dell'insieme in cui cercare il codice segreto.

Mastermind: aspetti di combinatorica



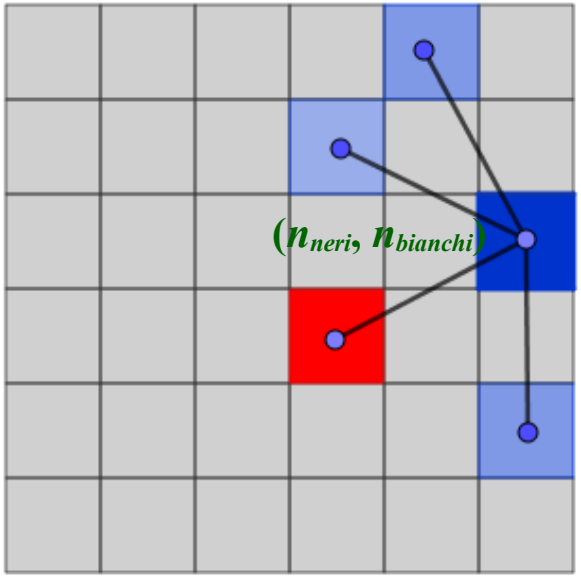
S è l'insieme di tutte le possibili disposizioni.
Ogni disposizione è rappresentata da un quadretto.

Tra queste disposizioni c'è quella scelta
come **codice segreto**

Il **(primo) tentativo** è in generale un'altra disposizione.

A questo tentativo il code-maker fornisce una risposta
$(n_{neri}, n_{bianchi})$ che permette di **ridurre la cardinalità**
dell'insieme in cui cercare il codice segreto

Mastermind: **aspetti di combinatorica**

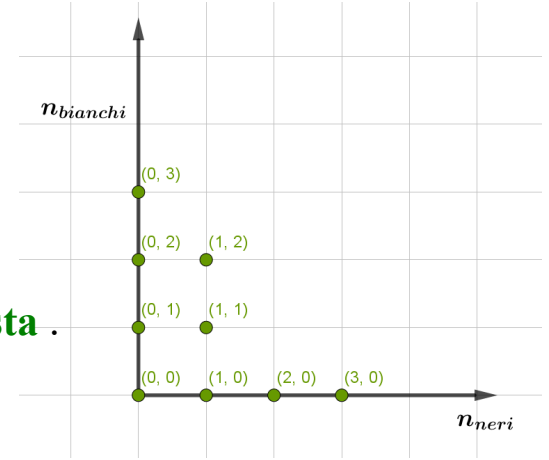


Per simmetria tutte le disposizioni che, confrontate con il tentativo ottengono la stessa risposta $(n_{neri}, n_{bianchi})$

appartengono ad un unico sottoinsieme
di cardinalità ridotta rispetto all'insieme iniziale,
ma che comunque che contiene il codice segreto.

Mastermind: aspetti di combinatorica

Consideriamo un caso semplificato: **il codice segreto è composto da 3 elementi** scelti da un insieme di tre colori **{A,B,C}**.
Le ripetizioni sono permesse.



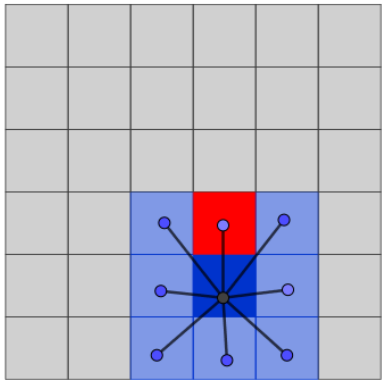
Le dispositioni sono $\mathbf{3^3=27}$.
Questa è la cardinalità dell'insieme S di partenza.

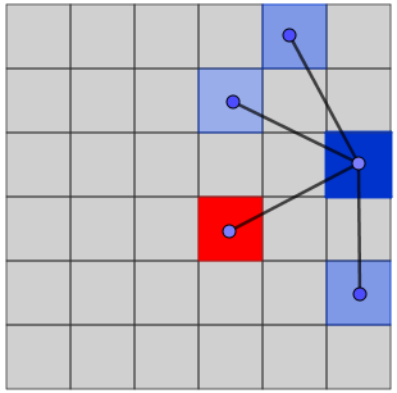
Il primo **tentativo** è confrontato con il **codice segreto** e riceve una **risposta** .
In questa versione semplificata ci sono solo **9** possibili **risposte**.

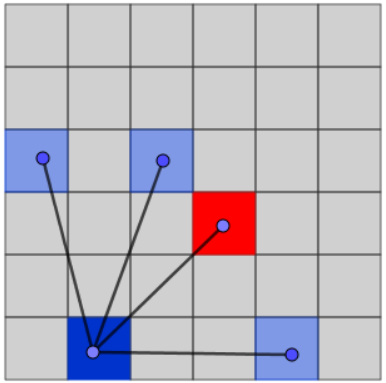
$$n_{bianchi} + n_{neri} \leq 3$$

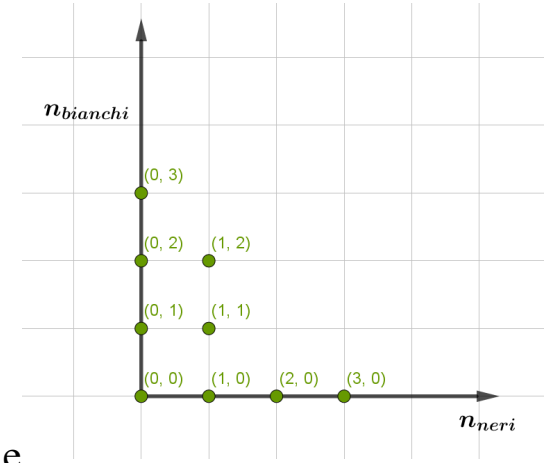
A seconda della risposta ricevuta si determina un sottoinsieme differente, con cardinalità differenti.
L'insieme S può essere partizionato in funzione del tentativo di partenza e della risposta ricevuta.

Mastermind: **aspetti di combinatorica**



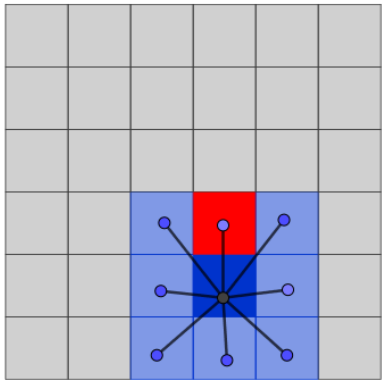


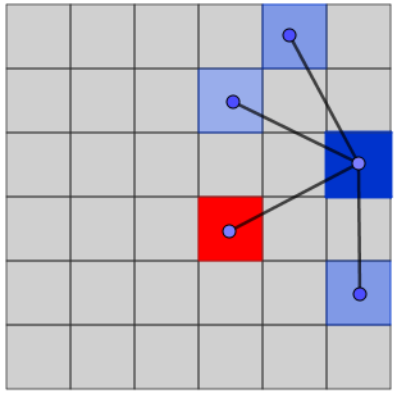


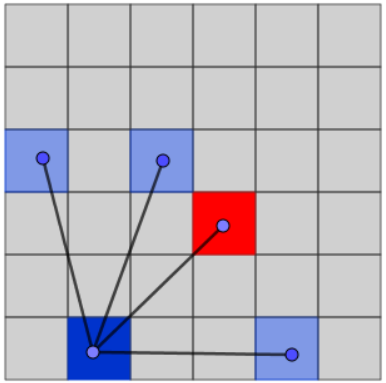


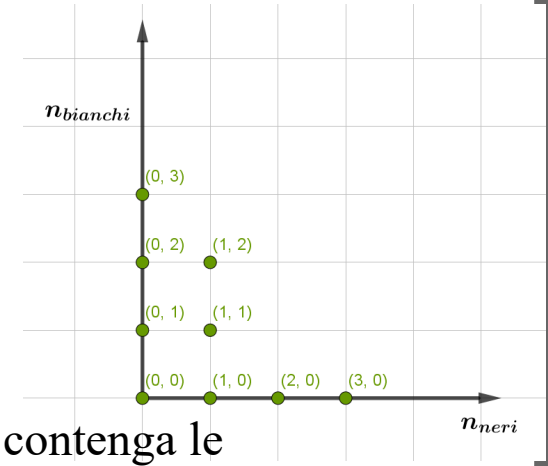
Il tentativo abbinato alla risposta ricevuta determina un sottoinsieme

Mastermind: aspetti di combinatorica









Dal punto di vista informatico occorre predisporre una **struttura dati** che contenga le disposizioni compatibili con le risposte ottenute e che venga aggiornata ad ogni tentativo provato.

E' necessaria anche una **funzione che confronti** il tentativo con il codice segreto fornendo la risposta e che poi confronti tutte le disposizioni con lo stesso tentativo per costruire il nuovo sottoinsieme in cui cercare il codice segreto.

Esempio 3x3

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

Codice segreto: AAB
Primo tentativo: BCA

Costruzione dei sottoinsiemi nel caso 3x3

Primo tentativo: BCA

Risposte	**Cardinalità dei sottoinsiemi**
(3,0)	1
(2,0)	6
(1,0)	3
(1,1)	6
(1,2)	3
(0,2)	6
(0,3)	2
(0,1)	
(0,0)	

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

Primo tentativo: BCA

Risposte	Cardinalità dei sottoinsiemi
(3,0)	1
(2,0)	6
(1,0)	3
(1,1)	6
(1,2)	3
(0,2)	6
(0,3)	2
(0,1)	5
(0,0)	1

Primo tentativo: AAB

Risposte	Cardinalità dei sottoinsiemi
(3,0)	1
(2,0)	6
(1,0)	5
(1,1)	3
(1,2)	2
(0,2)	5
(0,3)	
(0,1)	4
(0,0)	1

Costruzione dei sottoinsiemi

Primo tentativo: BCA

Risposte	Cardinalità dei sottoinsiemi
(3,0)	1
(2,0)	6
(1,0)	3
(1,1)	6
(1,2)	3
(0,2)	6
(0,3)	2
(0,1)	5
(0,0)	1

Primo tentativo: AAB

Risposte	Cardinalità dei sottoinsiemi
(3,0)	1
(2,0)	6
(1,0)	5
(1,1)	3
(1,2)	2
(0,2)	5
(0,3)	
(0,1)	4
(0,0)	1

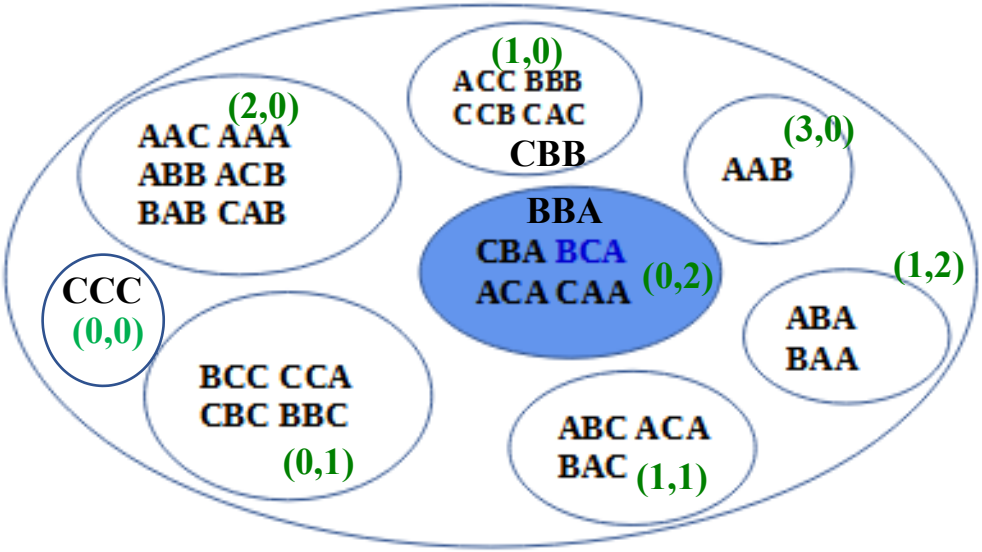
Primo tentativo: AAA

Risposte	Cardinalità dei sottoinsiemi
(3,0)	1
(2,0)	6
(1,0)	
(1,1)	12
(1,2)	
(0,2)	
(0,3)	
(0,1)	
(0,0)	8

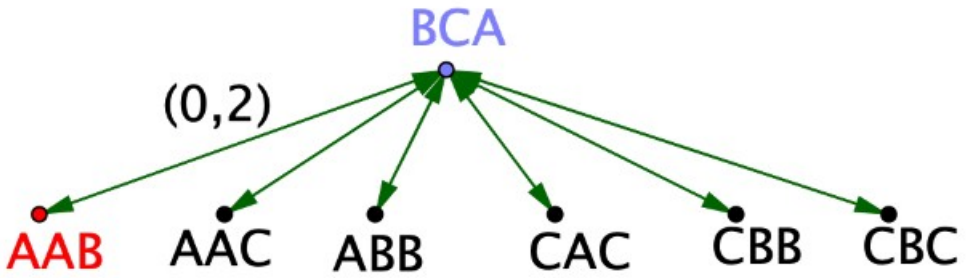
Costruzione dei sottoinsiemi

Partizione prodotta
Scegliendo come
Primo tentativo **BCA**

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

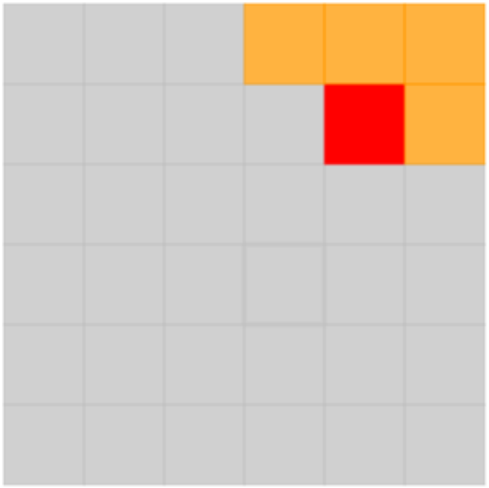


Simmetria e informazione nei sottoinsiemi

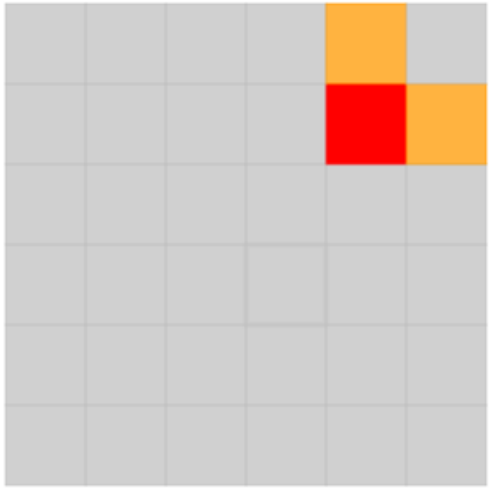


Costruzione dei sottoinsiemi - **tentativi successivi**

La probabilità di trovare il codice è 1/27



Scegliendo come
Primo tentativo **BCA**
La risposta **(0,2)** seleziona un sottoinsieme di 5 elementi.
La probabilità di trovare il codice in questo sottoinsieme è 1/5
p(**BCA**, **(0,2)**)= 5/27



Scegliendo come
Secondo tentativo **BBA**
La risposta **(2,0)** seleziona un sottoinsieme di 3 elementi.
La probabilità di trovare il codice in questo sottoinsieme è 1/3
…..

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.

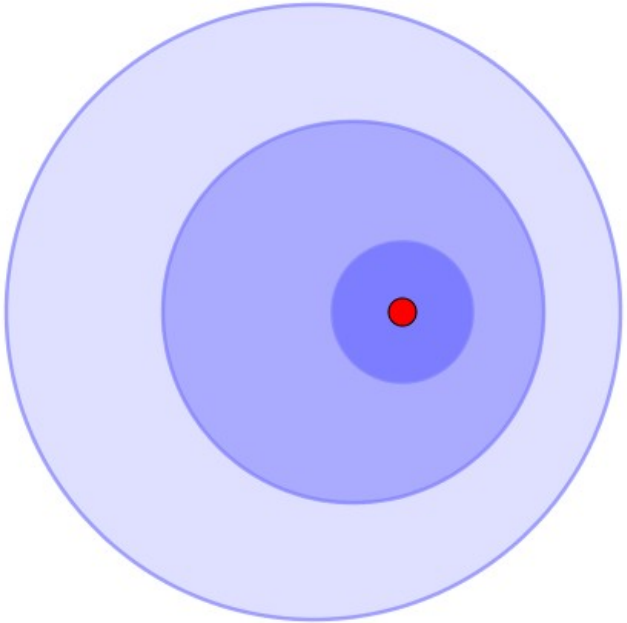
3. Il tentativo seguente permette una **nuova classificazione** degli elementi
che erano compatibili con la risposta seguente.

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

Costruzione dei sottoinsiemi

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

Il codice segreto appartiene anche
al nuovo sottoinsieme

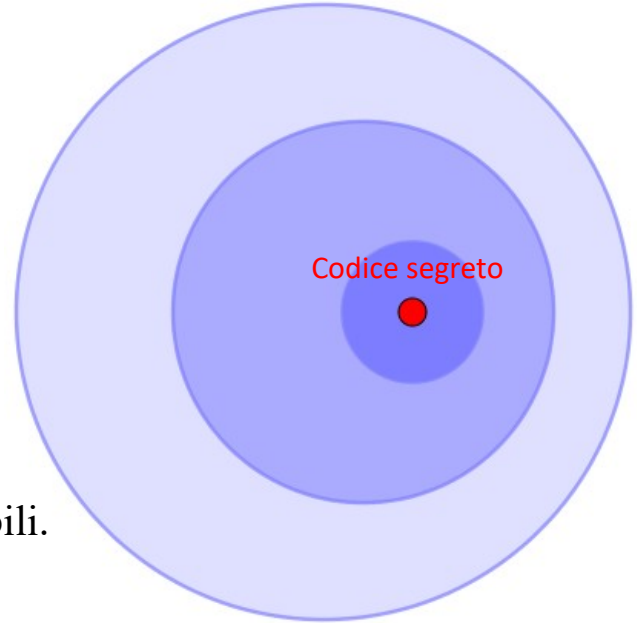


Mastermind: come valutare l'efficacia di un tentativo?

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.



Costruzione dei sottoinsiemi

E(I) Entropia dell'Informazione

$$\mathbf{I}=-\log_2 (p(t;r))$$

$$\mathbf{E(I)}=\Sigma_r [-p(t,r) \log_2 (p(t;r)]$$

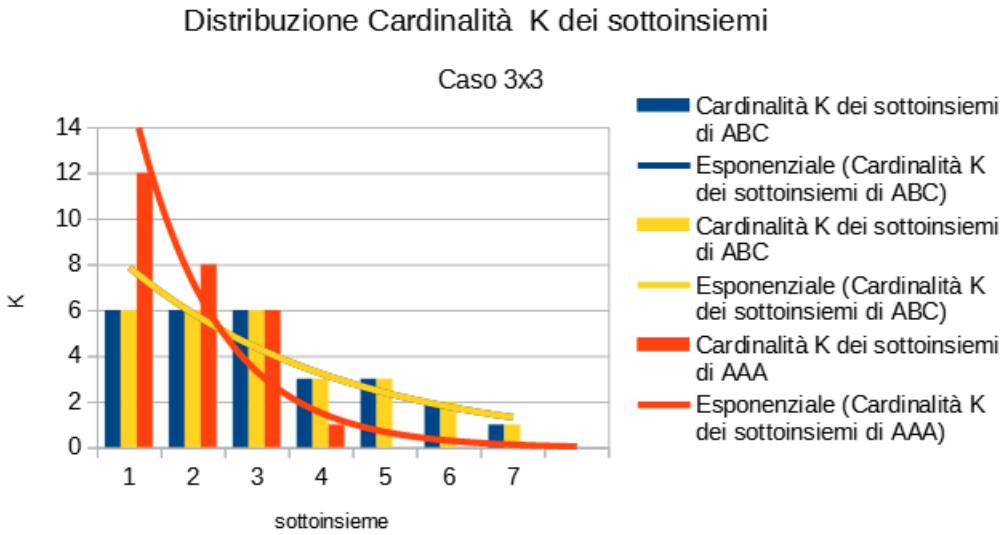
Tentativo ABC			**Tentativo AAB**			**Tentativo AAA**		
Cardinalità K dei sottoinsiemi di ABC	p=K/N di ABC	p· log₂(1/p)	Cardinalità K dei sottoinsiemi di AAB	p=K/N di AAB	p· log₂(1/p)	Cardinalità K dei sottoinsiemi di AAA	p=K/N di AAA	p· log₂(1/p)
6	0,22	0,48	**6**	0,22	0,48	**12**	0,44	0,52
6	0,22	0,48	**5**	0,19	0,45	**8**	0,30	0,52
6	0,22	0,48	**5**	0,19	0,45	**6**	0,22	0,48
3	0,11	0,35	**4**	0,15	0,41	**1**	0,04	0,18
3	0,11	0,35	**3**	0,11	0,35		0,00	
2	0,07	0,28	**2**	0,07	0,28		0,00	
1	0,04	0,18	**1**	0,04	0,18		0,00	
			1	0,04	0,18		0,00	
		E(I)			**E(I)**			**E(I)**
27		**2,61**	**27**		**2,60**	**27**		**1,70**

Costruzione dei sottoinsiemi

E(I) Entropia dell'Informazione

$$\mathbf{I}=-\log_2 (p(t;r))$$

$$\mathbf{E}(\mathbf{I})=\Sigma_r [-p(t,r) \log_2 (p(t;r)]$$



Strategia risolutiva- metodo min-max

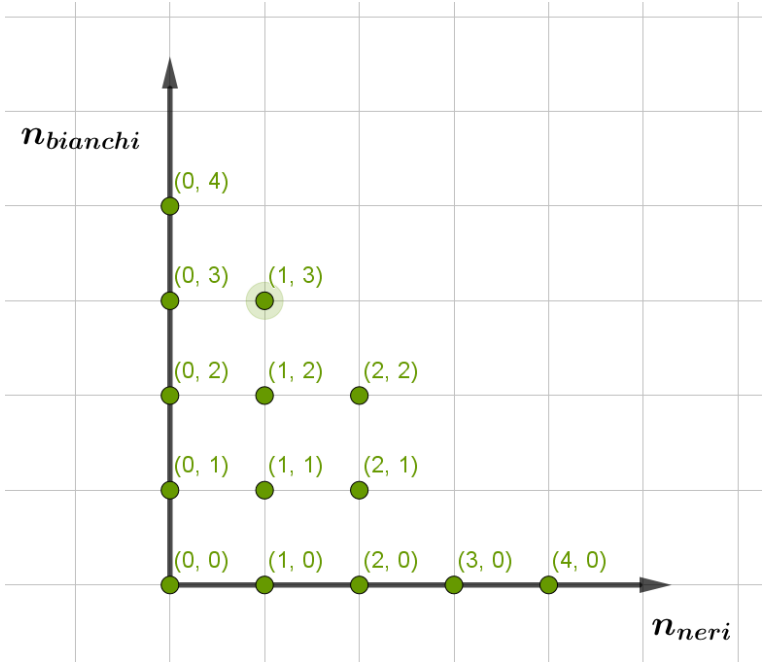
Per ogni partizione così ottenuta del mio insieme delle scelte
valuto la cardinalità K del sottoinsieme più numeroso (caso peggiore)
E scelgo quella partizione che produce sottoinsiemi con il minimo valore di K

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 $6^4 = 1296$

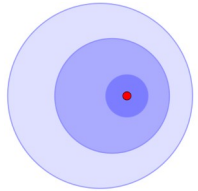
1. Esistono 14 possibili risposte ad un tentativo giocato.
   l'insieme delle disposizioni può essere partizionato
   al massimo in 14 sottoinsiemi disgiunti tra loro.
2. Ogni tentativo scelto tra le disposizioni produce
   una partizione differente.
3. Ciascun sottoinsieme di una stessa partizione è 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 per la versione REVERSE

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

Algoritmo di Knuth per la versione REVERSE

Il tentativo iniziale formato da una **doppia coppia** (es **AABB**) è quello che ha il più piccolo caso peggiore, in accordo con il metodo min-max.

Per arrivare a questo risultato costruisco le partizioni per ogni tipologia di tentativo e ogni conseguente tipo di risposta e misuro la cardinalità del caso peggiore.

CASI PEGGIORI

ABCD 312
AABC 276
AABB 256
AAAB 308
AAAA 625

La progettazione del codice

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

attività didattica in classe

1. Mastermind (versione Standard)

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. Mastermind (versione Reverse)

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

Il codice della versione Standard e della versione Reverse
si trova a questo link:
https://github.com/lauralamberti/Mastermind-game

La progettazione del codice

L'attività di coding è iniziata con la selezione delle strutture dati necessarie

DI COSA ABBIAMO BISOGNO?

una struttura dati che contenga l'elenco dei colori `colours[ ]`
una struttura dati che contenga il codice segreto `code[ ]`
una struttura dati che contenga le disposizioni `set_disp[ ]`
una sezione di input e una struttura dati che contenga l'input `guess[ ]`
una funzione che scelga il codice segreto `choice` (dalla libreria `random`)
una funzione che conti le concordanze tra il codice segreto e il tentativo `confronta`
una struttura che contenga la coppia risposta `(n_neri, n_bianchi)`
un contatore che conti il numero di tentativi effettuati `conta_tentativi`
una libreria per l'interfaccia grafica `pygame`

Il codice della versione Standard e della versione Reverse si trova a questo link:
https://github.com/lauralamberti/Mastermind-game

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(set_disp, tentativo, codice):
      elimino=[]
      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(set_disp)):
                  for i in range(4):
                  if (set_disp[j][i] in tentativo):
                              if (set_disp[j] not in elimino):
                                elimino.append(set_disp[j])
                                break

     for j in range (0,len(set_disp)):
                for i in range(4):
                  if (confronta(set_disp[j],tentativo)[0]!= n_neri_cod \
                  or confronta(set_disp[j],tentativo)[1]!= n_bianchi_cod):
                      if (set_disp[j] not in elimino):
                           elimino.append(set_disp[j])
                           break

      for item in elimino:
                set_disp.remove(item)
      if tentativo in set_disp:
                set_disp.remove(tentativo)
      elimino=[]
      return
```

La progettazione del codice

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

```
def punteggio (item, set_disp):
     max_occur=0
     occurrences=[0]
     for j in range (len(set_disp)):
          n_bianchi=confronta(item,set_disp[j])[0]
          n_neri=confronta(item,set_disp[j])[1]
          t=tuple(n_bianchi, n_neri)
          occurrences[t] +=1
     max_occur= max(occurrences.values())
     return max_occur
```

La progettazione del codice

L'algoritmo minimax di Knuth per la versione REVERSE

```
while not win :
    if (conta_tentativi==1):
        tentativo=["A", "A", "B", "B"]
    else:
        set_disp0.remove(tentativo)     #elimino il tentativo che sto per utilizzare
        Min_scores=1296
        best_guesses = []
        for j in range(len(set_disp0)):
                scores[j]=int(punteggio(set_disp0 [j],set_disp))
                                                    # la punteggio trova i casi peggiori
                if (scores[j]<Min_scores):
                   Min_scores=scores[j]  #trova punteggio minore tra i casi peggiori
        for j in range(len(set_disp0)):
            if (scores[j]==Min_scores):
                    best_guesses.append(set_disp0 [j])
                                #inserisco in best_guesses i migliori tra i peggiori

        tentativo = None
        for item in set_disp:
            if item in best_guesses:
                tentativo = item
                break
```

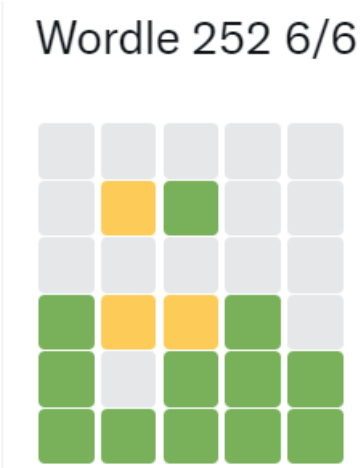
Possibili direzioni future della ricerca; proposte di nuove attività da svolgere in classe

Molto simile per certi versi è il gioco **Wordle** che sta spopolando in tutto il mondo. Se ne contano numerose versioni: in tutte le lingue e dedicate ad argomenti particolari come la storia, la geografia, la letteratura.

ITALIANO: https://pietroppeter.github.io/wordle-it/

INGLESE: https://www.nytimes.com/games/wordle/index.html

Commento di 3blu1brown
https://www.youtube.com/watch?v=v68zYyaEmEA



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

Dalla costruzione di un videogioco agli algoritmi decisionali di scelta

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

M. Kalogiannakis, S. Papadakis and A. -I. Zourmpakis, Gamification in Science Education. A Systematic Review of the Literature, Educ. Sci. 2021, 11, 22.
https://doi.org/10.3390/educsci11010022

https://supermastermind.github.io/playonline/game.html

Dalla costruzione di un videogioco agli algoritmi decisionali di scelta

Grazie!

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

CREATE MATH

NOT WAR