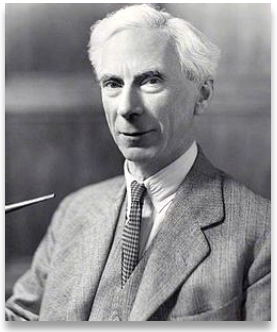
Università Roma Tre – Dipartimento di Matematica e Fisica
IN440 – Ottimizzazione Combinatoria
Prof. Marco Liverani

Insiemi e calcolo combinatorio

Insiemi



Bertrand Russel
(1872 – 1970)

- Un **insieme** A è una collezione di elementi di qualsiasi natura, *purché non contenga lo stesso* A

- **Paradosso di Russel** (Bertrand Russel, 1902):

  Un insieme è *«straordinario» se contiene anche se stesso. È* «ordinario» se non contiene se stesso.

  Sia $S = \{$collezione di tutti gli insiemi ordinari$\}$

  Problema: S è ordinario? Se lo fosse, allora $S \in S$, collezione di tutti gli insiemi ordinari; ma allora S sarebbe straordinario e questo ci porta ad una contraddizione! Se fosse straordinario, allora dovrebbe risultare $S \in S$, ma per definizione S contiene insiemi ordinari, per cui anche in questo caso abbiamo una contraddizione

- Per evitare situazioni paradossali evitiamo di trattare insiemi straordinari: d'ora in avanti consideriamo come insieme solo una collezione di oggetti che non contengono come elemento anche la collezione stessa

- In generale possiamo rappresentare un insieme con un computer attraverso una lista o un array. In Python esiste la struttura dati `set` (liste prive di elementi duplicati) su cui possiamo anche utilizzare gli operatori di unione «`|`», intersezione «`&`», differenza «`-`» e differenza simmetrica «`^`»

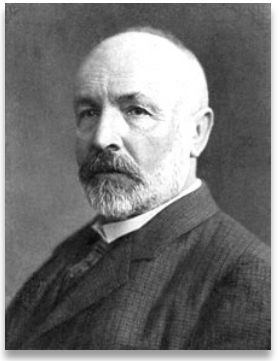
Insiemi

- L'**insieme delle parti** (o *insieme potenza*) di un insieme A è l'insieme di tutti i sottoinsiemi di A e lo indichiamo con $\mathcal{P}(A)$

- Una **partizione** di un insieme A è una collezione di sottoinsiemi di A, $\{A_1, ..., A_k\} \in \mathcal{P}(A)$, tale che $\bigcup_{i=1,...,k} A_i = A$ e $A_i \cap A_j = \emptyset$ per ogni $i \neq j$

- Una **relazione di equivalenza** ρ sugli elementi di un insieme A è una relazione *riflessiva* ($a \rho a$), *simmetrica* ($a \rho b \Rightarrow b \rho a$) e *transitiva* ($a \rho b$ e $b \rho c \Rightarrow a \rho c$)
- Dato un elemento $a \in A$, una **classe di equivalenza** su A definita dalla relazione d'equivalenza ρ è l'insieme $[a]_\rho = \{b \in A: a \rho b\}$
- Le classi di equivalenza su A definite mediante una relazione ρ formano una partizione di A

Cardinalità



Georg Cantor
(1845 – 1918)

- Insieme **finito**: gli elementi possono essere messi in corrispondenza biunivoca con l'insieme $\{1, 2, ..., n\}$. La **cardinalità** dell'insieme è n ($n = |A|$)
- Insieme **numerabile**: è possibile mettere gli elementi dell'insieme in corrispondenza biunivoca con $\mathbb{N}$. L'insieme ha la cardinalità del numerabile $\aleph_0$
- Insieme **continuo**: non è numerabile, non è possibile creare una corrispondenza biunivoca con $\mathbb{N}$. L'insieme ha la cardinalità del continuo $\aleph_1$

- Sia A un insieme finito, con $|A| = n$. Allora $|\mathcal{P}(A)| = 2^n$

  Dimostrazione per induzione su $|A|$: se $A = \emptyset$ allora $\mathcal{P}(A) = \{\emptyset\}$ e quindi $|\mathcal{P}(A)| = 1 = 2^0$. Si mostra poi che se $|\mathcal{P}(A)| = 2^{n-1}$ per $|A| = n-1$, allora per $A' = A \cup \{x\}$, $|A'| = n$ e $|\mathcal{P}(A')| = 2^n$; infatti avremo il doppio dei sottoinsiemi di A: tutti i sottoinsiemi di A e tutti i sottoinsiemi di A uniti a $\{x\}$ ■

  Dimostrazione costruttiva: si può creare una corrispondenza biunivoca tra $\mathcal{P}(A)$ e le stringhe binarie con n elementi. Con n cifre binarie possiamo definire 2^n numeri naturali: $0, 1, 2, ..., 2^{n-1}$. Quindi $|\mathcal{P}(A)| = 2^n$ ■

Sottoinsiemi e stringhe binarie

$A = \{ a_1, a_2, a_3, a_4, a_5 \}$

$A' = \{ a_1, a_2, a_3, a_4, a_5 \}$

$S = 1\ 0\ 1\ 1\ 0$

```
n = 4              S = ( 0, 0, 0, 0 )
i = 1              S = ( 1, 0, 0, 0 )
i = 1, 2           S = ( 0, 1, 0, 0 )
i = 1              S = ( 1, 1, 0, 0 )
i = 1, 2, 3        S = ( 0, 0, 1, 0 )
i = 1              S = ( 1, 0, 1, 0 )
i = 1, 2           S = ( 0, 1, 1, 0 )
i = 1              S = ( 1, 1, 1, 0 )
i = 1, 2, 3, 4     S = ( 0, 0, 0, 1 )
i = 1              S = ( 1, 0, 0, 1 )
i = 1, 2           S = ( 0, 1, 0, 1 )
i = 1              S = ( 1, 1, 0, 1 )
i = 1, 2, 3        S = ( 0, 0, 1, 1 )
i = 1              S = ( 1, 0, 1, 1 )
i = 1, 2           S = ( 0, 1, 1, 1 )
i = 1              S = ( 1, 1, 1, 1 )
i = 1, 2, 3, 4     S = ( 0, 0, 0, 0 )
```

Algoritmo 5 STRINGHEBINARIE(n)

Input: Un numero intero $n > 0$

Output: Tutte le stringhe binarie $S := (s_1, s_2, \ldots, s_n)$ con n cifre

```
 1: sia S := (0,0,...,0) (s_i := 0 per ogni i = 1,2,...,n)
 2: i := 1
 3: fintanto che i ≤ n ripeti
 4:     i := 1
 5:     fintanto che i ≤ n e s_i ≠ 0 ripeti
 6:         s_i := 0
 7:         i := i + 1
 8:     fine-ciclo
 9:     s_i := 1
10:     scrivi la stringa S = (s_1, s_2, ..., s_n)
11: fine-ciclo
```

Esercizio: possiamo modificare l'algoritmo per produrre tutti i sottoinsiemi di una lista/array?

Permutazioni

- **Permutazione** di un insieme A: corrispondenza biunivoca di A con se stesso.

  Es.: $A = \{1, 2, 3\}$, $\pi(A) = \{1 \to 2, 2 \to 1, 3 \to 3\}$

- Se $|A| = n$ allora possono essere definite $n!$ permutazioni di A.

  Dimostrazione: abbiamo n scelte per il primo elemento, per ciascuna di queste abbiamo $n-1$ scelte per il secondo elemento, ecc. Complessivamente quindi $n\ (n-1)\ (n-2)\ \ldots\ 1 = n!$ ■

- Algoritmo per la costruzione delle permutazioni di $A = \langle a_1, a_2, \ldots, a_n \rangle$:
  - a partire da una generica permutazione $\pi(A)$ si ricava la permutazione «successiva», $\pi'(A)$, mediante il seguente procedimento:
  - si individua il massimo indice k, $0 \le k < n$, tale che $a_k < a_{k+1}$ e si scambia l'elemento a_k con il più piccolo elemento $a_h > a_k$ con $k < h \le n$
  - infine si ordinano in ordine crescente gli elementi $a_{k+1}, a_{k+2}, \ldots, a_n$

Permutazioni

A partire da una generica permutazione $\pi(A)$ si ricava la permutazione «successiva», $\pi'(A)$, mediante il seguente procedimento:

- si individua il massimo indice k, $0 \leq k < n$, tale che $a_k < a_{k+1}$ e si scambia l'elemento a_k con il più piccolo elemento $a_h > a_k$ con $k < h \leq n$
- infine si ordinano in ordine crescente gli elementi a_{k+1}, a_{k+2}, ..., a_n

```
n = 4                          A = ( 1, 2, 3, 4 )

k = 3                          A = ( 1, 2, 4, 3 )
k = 2                          A = ( 1, 3, 2, 4 )
k = 3                          A = ( 1, 3, 4, 2 )
k = 2                          A = ( 1, 4, 2, 3 )
k = 3                          A = ( 1, 4, 3, 2 )
k = 1                          A = ( 2, 1, 3, 4 )
k = 3                          A = ( 2, 1, 4, 3 )
k = 2                          A = ( 2, 3, 1, 4 )
k = 3                          A = ( 2, 3, 4, 1 )
k = 2                          A = ( 2, 4, 1, 3 )
k = 3                          A = ( 2, 4, 3, 1 )
k = 1                          A = ( 3, 1, 2, 4 )
...                            ...
```

Algoritmo 6 PERMUTAZIONI(A, n)

Input: L'insieme $A = \{1, 2, \ldots, n\}$
Output: Tutte le permutazioni dell'insieme A

1: sia $P := (a_1 = 1, a_2 = 2, \ldots, a_n = n)$ la prima permutazione
2: scrivi la permutazione P
3: **ripeti**
4: $k := n - 1$
5: **fintanto che** $k > 0$ e $a_k > a_{k+1}$ **ripeti**
6: $k := k - 1$
7: **fine-ciclo**
8: **se** $k > 0$ **allora**
9: $h := 0$
10: **per** $i := k+1, k+2, \ldots, n$ **ripeti**
11: **se** $a_i > a_k$ e $(h = 0$ o $a_i < a_h)$ **allora**
12: $h := i$
13: **fine-condizione**
14: **fine-ciclo**
15: scambia a_k e a_h
16: ordina in ordine crescente gli elementi $a_{k+1}, a_{k+2}, \ldots, a_n$
17: scrivi la nuova permutazione $P = (a_1, a_2, \ldots, a_n)$
18: **fine-condizione**
19: **fintanto che** $k > 0$

Disposizioni

- **Disposizione semplice** di k elementi: sequenza *ordinata* di k elementi di A senza ripetizione degli elementi («*ordinata*» significa che conta l'ordine con cui dispongo gli elementi: due disposizioni semplici diverse possono avere gli stessi elementi, ma disposti in ordine diverso)

  Es.: se $A = \{1, 2, 3\}$ le disposizioni su A con $k = 2$ elementi sono $(1, 2)$, $(1, 3)$, $(2, 1)$, $(2, 3)$, $(3,1)$, $(3,2)$

- Se $|A| = n$ allora il numero di disposizioni semplici con k elementi su A è $D_{n,k} = \frac{n!}{(n-k)!}$

  Dimostrazione. Abbiamo n scelte per il primo elemento di ogni disposizione, $n-1$ scelte per il secondo elemento, ... $(n-k+1)$ scelte per il k-esimo elemento. Quindi in generale:

$$D_{n,k} = n(n-1)(n-2)\dots(n-k+1) = \frac{n(n-1)(n-2)\dots(n-k+1)(n-k)(n-k-1)\dots 1)}{(n-k)(n-k-1)\dots 1} = \frac{n!}{(n-k)!}$$

- Le **disposizioni con ripetizioni** di k elementi su un insieme di n elementi sono invece n^k: abbiamo infatti a disposizione n scelte per il primo elemento, altre n scelte per il secondo elemento, ..., ancora n scelte per il k-esimo elemento

Combinazioni

- I sottoinsiemi di A con k elementi sono le **combinazioni semplici** di A con k elementi
- In un insieme l'ordine degli elementi non conta, per cui il numero di combinazioni semplici di k elementi su un insieme di cardinalità n, è dato dal rapporto tra il numero $D_{n,k}$ di disposizioni semplici di k elementi ed il numero $k!$ di permutazioni di un insieme di k elementi:

$$C_{n,k} = \frac{n!}{k!(n-k)!}$$

- Algoritmo ingenuo per la generazione delle combinazioni semplici con k elementi di A:
  - si generano tutte le stringhe binarie con n elementi
  - tra queste si scelgono solo quelle con k elementi uguali a 1
  - a ciascuna di queste stringhe binarie corrisponde una combinazione semplice di k elementi su A
- L'algoritmo ingenuo genera *tutte* le stringhe binarie, ossia 2^n stringhe, quando a noi interessa solo una piccola parte di queste... è eccessivamente oneroso (complesso)!

Combinazioni

- Strategia:
  - generiamo solo combinazioni ordinate, in ordine crescente di k elementi selezionati dall'insieme $\{1, 2, ..., n\}$
  - come elemento i-esimo della sequenza ordinata che rappresenta una determinata combinazione di k elementi su A, scegliamo c_i tale da soddisfare la seguente condizione: $i \leq c_i \leq n - (k - i)$
- Per passare da una combinazione $(c_1, c_2, ..., c_k)$ alla successiva nell'ordine lessicografico, si prendono in esame gli elementi c_i a partire da $i = k$ fino ad $i = 1$ (ciclo alle righe 8–10) fino a quando non si incontra un elemento c_i che non abbia assunto il valore massimo per la posizione i-esima: $c_i < n - (k - i)$
- In tal caso si incrementa di 1 l'elemento c_i e si impostano di conseguenza il valore degli elementi successivi, $c_{i+1}, c_{i+2}, ..., c_k$ con il valore minimo per ciascun elemento

Algoritmo 7 COMBINAZIONI(A, n, k)

Input: L'insieme $A = \{a_1, a_2, \ldots, a_n\}$ e un intero $k \leq n$
Output: Le combinazioni in classi di k degli n elementi di A

```
 1: per i := 1,2,...,k ripeti
 2:     c_i := i
 3: fine-ciclo
 4: scrivi {a_{c_j}}_{j:=1,...,k}
 5: i := k
 6: fintanto che i > 0 ripeti
 7:     i := k
 8:     fintanto che i > 0 e c_i = n − (k − i) ripeti
 9:         i := i − 1
10:     fine-ciclo
11:     se i > 0 allora
12:         c_i := c_i + 1
13:         per j := i + 1, i + 2, ..., k ripeti
14:             c_j := c_{j−1} + 1
15:         fine-ciclo
16:         scrivi {a_{c_j}}_{j:=1,...,k}
17:     fine-condizione
18: fine-ciclo
```

Combinazioni

```
n = 6, k = 4                A = { a, b, c, d, e, f }
                                  1  2  3  4  5  6

C = (1, 2, 3, 4)   →        { a, b, c, d        }
C = (1, 2, 3, 5)   →        { a, b, c,    e     }
C = (1, 2, 3, 6)   →        { a, b, c,       f  }
C = (1, 2, 4, 5)   →        { a, b,    d, e     }
C = (1, 2, 4, 6)   →        { a, b,    d,    f  }
C = (1, 2, 5, 6)   →        { a, b,       e, f  }
C = (1, 3, 4, 5)   →        { a,    c, d, e     }
C = (1, 3, 4, 6)   →        { a,    c, d,    f  }
C = (1, 3, 5, 6)   →        { a,    c,    e, f  }
C = (1, 4, 5, 6)   →        { a,       d, e, f  }
C = (2, 3, 4, 5)   →        {    b, c, d, e     }
C = (2, 3, 4, 6)   →        {    b, c, d,    f  }
C = (2, 3, 5, 6)   →        {    b, c,    e, f  }
C = (2, 4, 5, 6)   →        {    b,    d, e, f  }
C = (3, 4, 5, 6)   →        {       c, d, e, f  }
```

Algoritmo 7 COMBINAZIONI(A, n, k)

Input: L'insieme $A = \{a_1, a_2, \ldots, a_n\}$ e un intero $k \leq n$
Output: Le combinazioni in classi di k degli n elementi di A

1: **per** $i := 1, 2, \ldots, k$ **ripeti**
2: $\quad c_i := i$
3: **fine-ciclo**
4: scrivi $\{a_{c_j}\}_{j:=1,\ldots,k}$
5: $i := k$
6: **fintanto che** $i > 0$ **ripeti**
7: $\quad i := k$
8: $\quad$ **fintanto che** $i > 0$ e $c_i = n - (k - i)$ **ripeti**
9: $\quad\quad i := i - 1$
10: $\quad$ **fine-ciclo**
11: $\quad$ **se** $i > 0$ **allora**
12: $\quad\quad c_i := c_i + 1$
13: $\quad\quad$ **per** $j := i + 1, i + 2, \ldots, k$ **ripeti**
14: $\quad\quad\quad c_j := c_{j-1} + 1$
15: $\quad\quad$ **fine-ciclo**
16: $\quad\quad$ scrivi $\{a_{c_j}\}_{j:=1,\ldots,k}$
17: $\quad$ **fine-condizione**
18: **fine-ciclo**

Coefficiente binomiale

- Il numero di combinazioni di n elementi in classi di k è il **coefficiente binomiale**: $\binom{n}{k} = \frac{n!}{k!(n-k)!}$

- Esprime il coefficiente nell'espressione della potenza di un binomio:

$$(x+y)^n = \sum_{k=0}^{n} \binom{n}{k} x^k y^{n-k}$$

- Dalla definizione di combinazione semplice e di coefficiente binomiale segue che un modo alternativo per contare gli elementi dell'insieme delle parti di A, $\mathcal{P}(A)$, è quello di sommare il numero di combinazioni di n elementi in classi di k, per $k = 0, 1, 2, \ldots, n$; per cui risulta:

$$\sum_{k=0}^{n} \binom{n}{k} = \binom{n}{0} + \binom{n}{1} + \binom{n}{2} + \cdots + \binom{n}{n-1} + \binom{n}{n} = 2^n$$

Coefficiente binomiale

- Osserviamo che per $0 < k < n$ vale la seguente identità: $\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$

  Dimostrazione:

$$\begin{aligned}
\binom{n-1}{k-1} + \binom{n-1}{k} &= \frac{(n-1)!}{(k-1)!\,(n-k)!} + \frac{(n-1)!}{k!\,(n-k-1)!} = \\
&= \frac{(n-1)!}{(k-1)!\,(n-k)\,(n-k-1)!} + \frac{(n-1)!}{k\,(k-1)!\,(n-k-1)!} \\
&= \frac{k(n-1)! + (n-k)(n-1)!}{k\,(n-k)\,(k-1)!\,(n-k-1)!} = \\
&= \frac{(k+n-k)\,(n-1)!}{k\,(k-1)!\,(n-k)\,(n-k-1)!} = \\
&= \frac{n!}{k!\,(n-k)!} = \\
&= \binom{n}{k}
\end{aligned}$$

- Sfruttando questa relazione si può costruire uno «schema tabulare» per semplificare il calcolo del coefficiente binomiale: il **Triangolo di Tartaglia** (o *Triangolo di Pascal*, per i francesi)

Coefficiente binomiale


Niccolò Tartaglia
(1499 – 1557)

- **Triangolo di Tartaglia**: schema tabulare *ricorsivo* per la costruzione dei coefficienti binomiali
- Per $n = 0$ e $k = 0$ risulta $\binom{n}{k} = \binom{0}{0} = 1$
- Per $n = 1, 2, \ldots$ e per $k = 0, 1, \ldots, n$: $$\binom{n}{k} = \begin{cases} \binom{n-1}{0} & \text{se } k = 0 \\ \binom{n-1}{k-1} + \binom{n-1}{k} & \text{se } 0 < k < n \\ \binom{n-1}{0} & \text{se } k = n \end{cases}$$
- In questo modo si può costruire uno schema triangolare con cui diventa banale calcolare il coefficiente binomiale degli elementi di una riga, utilizzando i termini calcolati nella riga precedente

$$\begin{array}{ccccccccccc} & & & & \binom{0}{0} & & & & & & \\ & & & \binom{1}{0} & & \binom{1}{1} & & & & & \\ & & \binom{2}{0} & & \binom{2}{1} & & \binom{2}{2} & & & & \\ & \binom{3}{0} & & \binom{3}{1} & & \binom{3}{2} & & \binom{3}{3} & & & \\ \binom{4}{0} & & \binom{4}{1} & & \binom{4}{2} & & \binom{4}{3} & & \binom{4}{4} & & \end{array} \quad \Rightarrow \quad \begin{array}{ccccccccccc} & & & & 1 & & & & & & \\ & & & 1 & & 1 & & & & & \\ & & 1 & & 2 & & 1 & & & & \\ & 1 & & 3 & & 3 & & 1 & & & \\ 1 & & 4 & & 6 & & 4 & & 1 & & \end{array}$$

Coefficiente binomiale

- Per $n = 0$ e $k = 0$ risulta $\binom{n}{k} = \binom{0}{0} = 1$
- Per $n = 1, 2, \ldots$ e per $k = 0, 1, \ldots, n$:

$$\binom{n}{k} = \begin{cases} \binom{n-1}{0} & \text{se } k = 0 \\ \binom{n-1}{k-1} + \binom{n-1}{k} & \text{se } 0 < k < n \\ \binom{n-1}{0} & \text{se } k = n \end{cases}$$

Algoritmo 8 COEFFICIENTEBINOMIALE(n, k)

Input: La coppia di interi $k \geq 0$ e $n \geq k$
Output: Il coefficiente binomiale $\binom{n}{k}$

```
1: se n = 0 allora
2:     c := 1
3: altrimenti
4:     se k = 0 o k = n allora
5:         c := CoefficienteBinomiale(n − 1, 0)
6:     altrimenti
7:         c := CoefficienteBinomiale(n − 1, k − 1) +
           CoefficienteBinomiale(n − 1, k)
8:     fine-condizione
9: fine-condizione
10: restituisci c
```

Combinatoria su insiemi finiti

Consideriamo un insieme $A = \{a_1, a_2, ..., a_n\}$ con n elementi ($|A| = n$)

Operazione combinatoria	Cardinalità	Descrizione
insieme delle parti di A	2^n	tutti i **sottoinsiemi** di A, di qualsiasi cardinalità
permutazioni di A	$n!$	tutte le ***n*-ple** (sequenze di n elementi) distinte, formate utilizzando tutti gli elementi di A, senza ripeterli
disposizioni semplici di k elementi di A	$\frac{n!}{(n-k)!}$	tutte le ***k*-ple** (sequenze di k elementi) distinte, formate usando solo k elementi di A, senza ripeterli
disposizioni con ripetizione di k elementi di A	n^k	tutte le ***k*-ple** (sequenze di k elementi) distinte, formate usando solo k elementi di A, anche ripetuti più volte
combinazioni semplici con k elementi di A	$\frac{n!}{k!(n-k)!}$	tutti i **sottoinsiemi** di A con soli k elementi

Quadrati Latini

- Un **Quadrato Latino** di ordine n è una matrice quadrata di n righe ed n colonne, i cui elementi appartengono all'insieme $\{1, 2, ..., n\}$ e su cui non si ripetono mai due elementi uguali su nessuna riga e nessuna colonna
  - Es.: Quadrato Latino di ordine 4

1	2	3	4
2	3	4	1
4	1	2	3
3	4	1	2

- Due quadrati latini di ordine n si dicono **ortogonali** se le n^2 coppie formate dagli elementi corrispondenti dei due quadrati sono tutte distinte
- Un quadrato latino **normalizzato** è ottenuto ponendo la prima riga e la prima colonna uguali alla sequenza $(1, 2, ..., n)$

Quadrati Latini

- Calcolare il numero di quadrati latini di ordine n è molto difficile: il risultato è noto solo per quadrati latini di ordine minore di 16
- In generale esiste una relazione che lega il numero di quadrati latini di ordine n, $N(n,n)$, con il numero di quadrati latini normalizzati dello stesso ordine, $L(n,n)$:

$$N(n,n) = n!\ (n-1)!\ L(n,n)$$

- Il problema del calcolo del numero di quadrati latini differenti di ordine n, si riduce al calcolo del numero di quadrati latini normalizzati
- Purtroppo non esiste una formula in grado di calcolare direttamente quel numero: al momento tale risultato, calcolato con algoritmi esaustivi (i cosiddetti metodi «a forza bruta», che si limitano a costruire e contare tutte le possibili configurazioni)
  - Es.: $L(2, 2) = 1$, $L(3, 3) = 1$, $L(4, 4) = 4$, $L(7, 7) = 16.942.080$, $L(9, 9) = 377.597.570.964.258.816$

Il gioco del Sudoku

- Il gioco del Sudoku è un rompicapo giapponese che sta riscuotendo un successo notevole nel cosiddetto «grande pubblico»
- Il problema da risolvere in una partita di Sudoku può essere riassunto nei seguenti termini: riempire una griglia di 9×9 elementi, in modo tale che ogni riga, ogni colonna ed ognuna delle nove sotto-griglie 3×3 contenga le cifre da 1 a 9.
- La matrice è dunque un Quadrato Latino di ordine 9; al problema di determinare un quadrato latino, viene aggiunto il vincolo dell'obbligo di utilizzare tutti gli elementi dell'insieme $\{1, 2, ..., 9\}$ nelle 9 sotto-matrici di quadrate di ordine 3
- Per obbligare il giocatore a produrre ad ogni partita del gioco un quadrato latino differente, il gioco inizia con una matrice di 9×9 elementi in cui alcune delle posizioni sono preimpostate con dei valori

Il gioco del Sudoku

- Esempio:

		2	5		9	1		3
6	3			2			4	5
4								
		5	1		4	8		
2								7
5	4			3			2	1
		8	7		6	5		

8	7	2	5	4	9	1	6	3
9	5	4	3	6	1	2	7	8
6	3	1	8	2	7	9	4	5
4	8	3	2	7	5	6	1	9
7	6	5	1	9	4	8	3	2
2	1	9	6	8	3	4	5	7
5	4	6	9	3	8	7	2	1
1	9	7	4	5	2	3	8	6
3	2	8	7	1	6	5	9	4

Il gioco del Sudoku: il numero di configurazioni

- Quante diverse configurazioni è possibile costruire per il gioco del Sudoku?
- Si deve osservare che due matrici diverse, M e M' possono rappresentare la stessa configurazione nel caso in cui esista un isomorfismo φ: $M \longrightarrow M'$ tale che $\varphi(m_{i,j}) = \varphi(m_{h,k})$ se e solo se $m_{i,j} = m_{h,k}$
- Es.: $\varphi(1) = 8, \varphi(2) = 7, \varphi(3) = 2, \varphi(4) = 5, \varphi(5) = 4, \varphi(6) = 9, \varphi(7) = 1, \varphi(8) = 6, \varphi(9) = 3$

M

1	2	3	4	5	6	7	8	9
6	4	5	9	8	7	3	2	1
8	9	7	1	3	2	6	5	4
5	1	9	3	2	4	8	7	6
2	8	4	7	6	5	1	9	3
3	7	6	8	1	9	5	4	2
4	5	8	6	9	1	2	3	7
7	6	2	5	4	3	9	1	8
9	3	1	2	7	8	4	6	5

$M' = \varphi(M)$

8	7	2	5	4	9	1	6	3
9	5	4	3	6	1	2	7	8
6	3	1	8	2	7	9	4	5
4	8	3	2	7	5	6	1	9
7	6	5	1	9	4	8	3	2
2	1	9	6	8	3	4	5	7
5	4	6	9	3	8	7	2	1
1	9	7	4	5	2	3	8	6
3	2	8	7	1	6	5	9	4

- Altre configurazioni equivalenti possono essere ottenute per rotazione della matrice o per simmetria

Il gioco del Sudoku: il numero di configurazioni

- Teorema (Felgenhuer e Jarvis, 2005). Il numero di configurazioni valide del Sudoku è pari a

$$2^7 \times 27.704.267.971 = 3.546.146.300.288$$

- È sorprendente notare che il fattore $27.704.267.971$ è un numero primo

Il gioco del Sudoku: un algoritmo ricorsivo

- Algoritmo ricorsivo per il calcolo della soluzione del Sudoku, a partire da una configurazione valida (ma incompleta) M
- Indichiamo con $m_{i,j} = 0$ i valori non definiti della configurazione rappresentata dalla matrice M
- La funzione ricorsiva **SudokuSolve(M)** restituisce 1 (*true*) se la configurazione M è compatibile, altrimenti restituisce 0 (*false*)
- Al termine della catena di chiamate ricorsive, quando tutti i valori della matrice sono stati assegnati in modo compatibile con le regole, la funzione stampa la matrice M che rappresenta la configurazione finale che risolve la partita

Algoritmo 9 SUDOKUSOLVE(M)

```
1:  rc := 0
2:  siano i e j gli indici minimi per cui m_{i,j} = 0
3:  se esiste m_{i,j} = 0 allora
4:      per k = 1,2,...,9 ripeti
5:          m_{i,j} := k
6:          se M è una configurazione valida e
            SudokuSolve(M) = 1 allora
7:              rc := 1
8:          fine-condizione
9:      fine-ciclo
10:     se rc = 0 allora
11:         m_{i,j} := 0
12:     fine-condizione
13: altrimenti
14:     Sudoku risolto! Stampa la soluzione M
15:     rc := 1
16: fine-condizione
17: restituisci rc
```

Riferimenti bibliografici

- Cormen, Leiserson, Rivest, Stein, «*Introduzione agli algoritmi e strutture dati*», terza edizione, McGraw-Hill (Appendice C.1)
- Marco Liverani, *Dispense del Corso di Ottimizzazione Combinatoria: Insiemi ed elementi di calcolo combinatorio* (http://www.mat.uniroma3.it/users/liverani/doc/disp_oc_02.pdf)