Agenda e Comunicazioni

- *In http://gordion.casaccia.enea.it/SicurezzaInformatica/Exercises aggiunte cartelle*
- *Vigenere e Scytale*
- *Alcuni algoritmi di cifratura (come Vigenere) sono più "complessi" di altri (tipo Caesar) e non siamo stati in grado di ispezionare esaustivamente lo spazio delle chiavi.*
- *Oggi introdurremo il concetto di Complessità. Calcoleremo la complessità di alcuni algoritmi e risolveremo qualche esercizio.*

Complexity

Definizione del Webster

- *the state or quality of having many interrelated parts or aspects <the complexity* of the company's computer system is such that a full-time repairman is needed>
- Synonyms complexness, complicacy, complicatedness, complication, elaborateness, intricacy, intricateness, involution, knottiness, sophistication
- Related Words diversity, heterogeneity, heterogeneousness, multifariousness; impenetrability, incomprehensibility, inexplicability
- Near Antonyms simplification; homogeneity, uniformity
- Antonyms plainness, simpleness, simplicity

Complessità Treccani

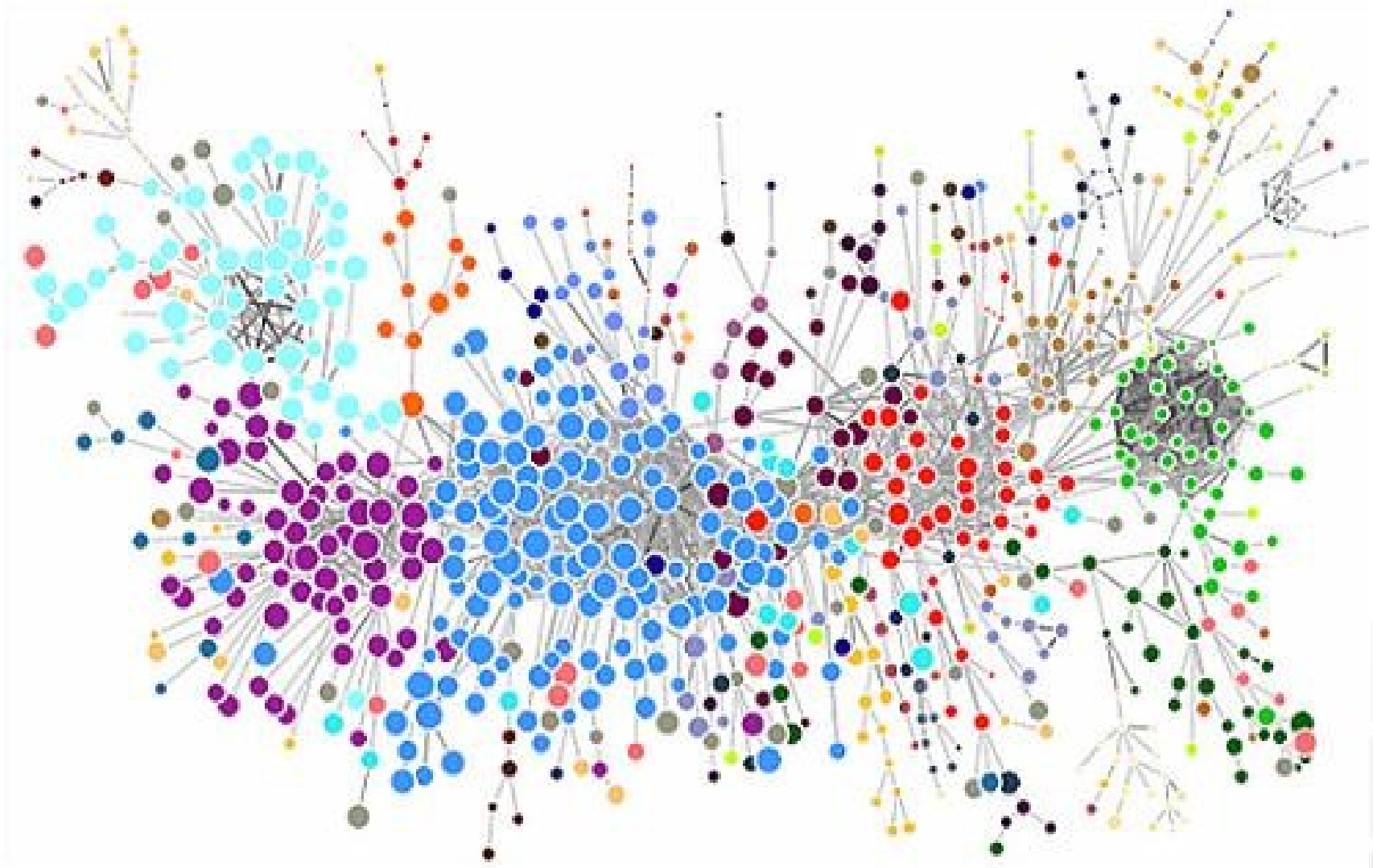
- *1. L'esser complesso (nelle varie accezioni dei sign. 1 e 2 di quest'agg.): c. di una questione, di un ragionamento, di una costruzione teorica; c. di un atto giuridico; esaminare una situazione in tutta la sua complessità; solo il discorso chiaro può essere di una c. inesauribile* (*Giuseppe Pontiggia*).
- *2. Caratteristica qualitativa di un sistema, cioè di un aggregato organico e strutturato di parti tra loro interagenti, che gli fa assumere proprietà che non derivano dalla semplice giustapposizione delle parti. È la proprietà specifica dei sistemi complessi (v. sistema, n. 1 b), rappresentata in varia forma da quell'insieme di teorizzazioni matematiche, informatiche e scientifiche che taluni caratterizzano con la locuz. scienza della c., per indicare una nuova metodica di indagine che si contrappone alla tradizionale tendenza a ridurre il complesso al semplice. In matematica e in informatica teorica, teoria della c., studio degli algoritmi in base ai quali possono essere sviluppati i programmi informatici che permettono di calcolare i valori assunti da determinate funzioni o successioni numeriche, che, nelle applicazioni scientifiche e tecniche, descrivono l'evoluzione di sistemi complessi; in partic., c. algoritmica, con riferimento a una successione di cifre, o a una funzione matematica, indice della difficoltà degli algoritmi (basato principalmente sulla loro lunghezza) elaborati per costruire i programmi di calcolo e, conseguentemente, confrontare e classificare la complessità degli oggetti matematici costruiti dai programmi* stessi.

Complessità

- *Può essere riferito alla varie caratteristiche:*
  - *Rappresentazione di un sistema.*
  - *Organizzazione di un sistema*
  - *Regole evolutive di un sistema*
  - *Problema*
  - *Soluzione di un problema*
  - *Reti complesse*

Complesso è diverso da Complicato

Reti Complesse



Problemi

Un problema in genere si formula in modo astratto

$F(x,d)=0$

Trovare ***x*** *(****incognita****) conoscendo* ***d*** *(****dato****) e sapendo che vale la relazione su scritta. La soluzione è* $x=x(d)$.

*Il problema è "****ben posto****" se* ***esiste una soluzione*** *unica o se è noto a priori che esistono le soluzioni ed è possibile definirne il numero.*

Problemi

Dimensione *di una problema è la dimensione dei dati assegnati (o raramente della soluzione).*

Se i dati sono esprimibili in termini di numeri interi, la dimensione è il numero (minimo) di bit necessario per definire il dato.

Es: la somma di ***n*** *numeri a 64 bit necessita* ***64 n*** *bits. La dimensione è* ***n****.*

Algoritmo

*Un **algoritmo** è una procedura che consente di risolvere un problema. (es. F(x,d)=0)*

*Una procedura è una serie di **azioni** fattibili che creano dati a partire ad un insieme iniziale.*

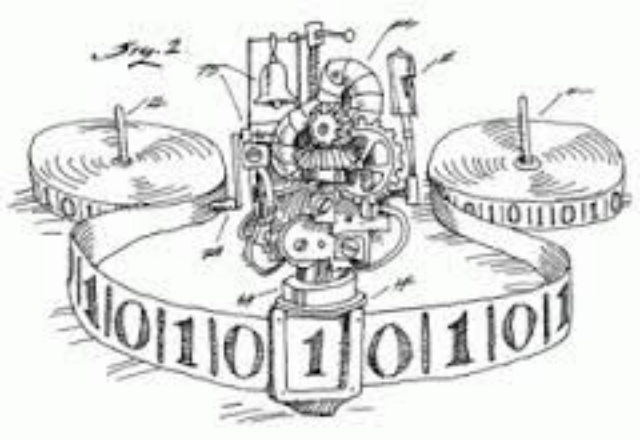
$$X=P(d)=\dots P_n(P_{n-1}(P_{n-2}(P_{n-3}(\dots P_j(\dots P_2(P_1(d)\,)\,)\,)\,)\,)\,)$$

Ogni azione P_i deve essere eseguibile da una ***Macchina di Turing.***

Macchina di Turing

*MdT: (**S**, s_0, S_{fin}, **A**, vuoto, **rule**)*

- ***S*** *Spazio degli stati*
- S_{fin} *stati finali;* s_0 *stato iniziale.*
- ***A*** *Alfabeto;* ***vuoto*** *carattere vuoto*



- ***Rule****: (s,a) -> (t, b,* ***shift****)*
- ***Shift****: -1,0,1 (indietro, ferma, avanti)*

"Partendo dallo stato s, con input a,
si va allo stato t con output b e shift"

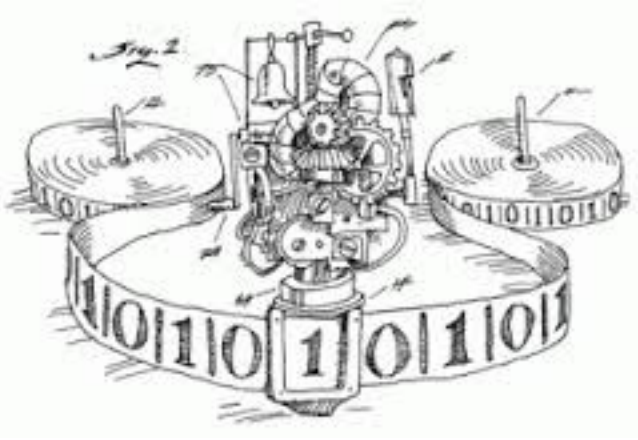
Alan Turing
23 Giugno 1912, 7 Giugno 1954

Macchina di Turing

*MdT: (**S**, s_0, $\mathbf{S_{fin}}$, **A**, vuoto, **rule**)*

La Complessità della MdT dipende da

|S|,|A| e dalla complessità della regola.



Tutte le regole possibili sono date da:

$(3|S||A|)^{|S||A|}$

|S|=cardinalità di S= numero di stati

|A|=cardinalità di A= numero di caratteri

La macchina Enigma

La macchina e segue una cifratura sostituzionale (solo 12 casatteri) seguita da 3 trasposizionali (3 "rotori").

Turing decriptò le chiavi dei tedeschi con metodi di "ingegneria inversa" (Reverse Engineering").

La macchina fu creata per quello scopo, ma il concetto va al di là.

Dicotomia "Tertium non datur"

Spazio di due soli elementi: 0 <-> Falso, 1<-> Vero

Operazioni E (AND), O-Vel (OR), O-Aut (XOR)

A	B	A.AND.B
0	0	0
0	1	0
1	0	0
1	1	1

A	B	A.OR.B
0	0	0
0	1	1
1	0	1
1	1	1

A	B	A.XOR.B
0	0	0
0	1	1
1	0	1
1	1	0

A	B	A.NOR.B
0	0	1
0	1	0
1	0	0
1	1	0

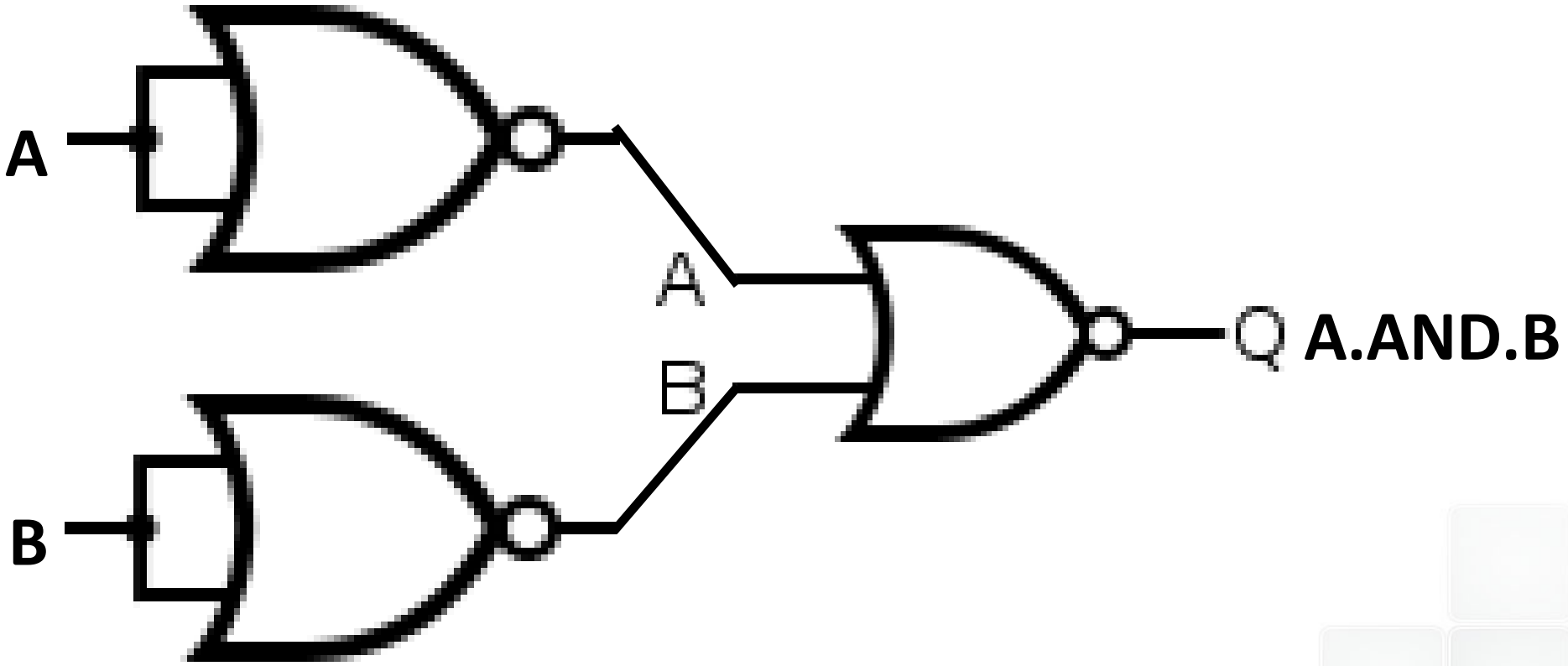
A	\A
0	1
1	0

- *Non *

Connettivi fondamentali

- *Tutti i connettivi si possono ricavare dal NOR o dal NAND.*
- \A=A.NOR.A
- A.OR.B=\(A.NOR.B)
- A.AND.B=\A.NOR.\B=\(\A.OR.\B)
- A->B=\A.OR.(A.AND.B)
- A.XOR.B=(A.AND.\B).OR.(\A.AND.B)

Es: AND con NOR



\A .NOR. \B=\(\A .OR. \B)=A.AND.B

Rappresentazione base binaria

- *Per trovare la cifra a_0:*

Se n è pari si scrive 0 se è dispari si scrive 1

Si sottrae a_0 ad n

n-> n-a_0. Adesso n è sempre pari. Si divide per 2.

Si itera il procedimento per a_1 a_2 etc n-> …$a_2a_1a_0$.

Esempio numero: 13

a_0=1, -> 13-1=12 ->12/2= 6 -> a_1=0 ->6-0=6 ->6/2=3

-> a_3=1 -> 3-1=2 ->2/2=1 -> a_4=1 $(13)_{base2}$=1101

Verifica: 13=8+4+0+1

Macchina di Turing per la somma

- *Somma (16 bit) c=c+1100101 c=101100110 (101+358)*
- *Numeri sul nastro di Turing (macchina a 1 nastro)*
- *0000000001100101000000101100110*
- *"s_0"=Devo leggere q^{16}_0(16Dx) -> q^{16}_0(15Dx)-> …-> dopo 16 passi a dx-> q^{16}_0*
- *"s_1"=leggo q^{16}_{01} (16 Dx) (1 in questo caso) -> vai avanti 16 dx -> q^{16}_{01}*
- *"s_2"=leggo q^{16}_{010} (16 Sx) (0 in questo caso) scrivi vuoto; dopo indietro 16 -> q^{16}_{010}*
- *"s_3"=scrivi xor (1 qui) s=q^{15}_{and} (1 Sx) , va uno a Sx -> q^{15}_{and}*
- *Si Ricomincia fino a q^0 .*

0000000001100101+
0000000101100110=
0000000111001011

Ci sono $17x33x2^3$ stati

Macchina di Turing per la somma

- *Alfabeto =(vuoto, 0,1) |A|=3*

- *S= Stati possibili della macchina:*

Uno stato è definito da

$Q^{bitstep}{}_{riporto\ val1\ val2}$ *(shift-to-do Direction)* **esempio $q^{16}{}_{011}$ (15Dx)**

Bitstep (bit che si sta calcolando) va da zero a 16 (17 valori)

Shift va da 16 Dx a 16 Sx (33 valori 16Dx … 0 … 16Sx =(-16,16))

Riporto è 0 o 1; val1, val2 sono vuoti oppure 0 o 1 (alfabeto +1)

*Quindi ci sono 17 * 33 * 2*3² stati possibili = 3366 "stadi"*

$S_0 = q^{16}$ *stato iniziale* $S_f = q^0$ *due stati finale (f=0 ok f=1 numeri troppo grandi)*

Algoritmo

Un algoritmo è una procedura che consente di risolvere un problema. F(x,d)=0

Una procedura è una serie di azioni fattibili che creano dati a partire ad un insieme iniziale.

$$X=P(d)=\ldots P_n(P_{n-1}(P_{n-2}(P_{n-3}(\ldots P_j(\ldots P_2(P_1(d)\,)\,)\,)\,)\,))$$

Complessità di un Algoritmo

Misura le risorse di tempo e di spazio necessarie per risolvere un problema usando l'algoritmo.

Complessità di allocazione di memoria e di tempo.

*La **complessità intrinseca di un problema** è la complessità dell'algoritmo più semplice che lo risolve.*

L'algoritmo ottimale può dipendere dalle dimensioni del problema e può essere ignoto e non unico.

Allocazione Memoria

Indici di complessità sono:

-La massima memoria allocata

-La memoria minima allocata

-La memoria media allocata

Indice di Riscalamento (scaling): come scala la memoria in funzione della dimensione.

-Esempio: per sommare n numeri:

$M_{max}=n$; $M_{min}=1$; $M_{ave}=n/2$; *Scala linearmente in n*

- Decriptare con decritta.m: $M_{max}=26\ n$ *(lineare)*

Tempo di esecuzione

Indici di complessità sono:

-Tempo Massimo di esecuzione

-Tempo Minimo di esecuzione

-Tempo Medio di esecuzione

Indice di Riscalamento (scaling): come scala il tempo di esecuzione in funzione della dimensione.

-Esempio: per sommare n numeri:

T_{max}=n; T_{min}=n; M_{ave}=n; Scala linearmente in n

- decritta.m: M_{max}=26 LEN (lineare)

Tempo di esecuzione

Si misura in cicli di una macchina.

Il tempo reale (in secondi) si ottiene moltiplicando il numero di cicli per il periodo di un ciclo cioè dividendo per la frequenza ν della macchina (in Hertz).

$$T = ncicli / \nu$$

Le prestazioni di una macchina si misurano in cicli al secondo o multipli.

Spesso serve di fare operazioni tra numeri reali a virgola variabile (floating point number) e quindi si misura in Flops (floating point operations per second) o multipli Mflops, Gflops, Tflops, Peta-flops, Exa-flops.

Tempo di esecuzione

Spesso si distingue tra CPU-time I/O time and enlapsed time.

CPU= Central Processor Unit

I/O input output

Le memorie sono gerarchicizzate L1 L2 etc. I tempi di accesso sono diversi. Quindi anche il tempo di esecuzione cambia.

*La **complessità** in termini di operazioni della macchina per risolvere un problema non dipende dalle prestazioni, ma solo dall'algoritmo.*

Complessità di calcolo

La legge con cui scala il numero di passi da eseguire per completare l'algoritmo definisce il suo tipo.

***P** = Polinomiale (n è dimensione del dato).*

$N_{cicli} = P(n) = a\,n^k + b\,n^{k-1} + c\,n^{k-1} + \ldots$

*Polinomiale di **ordine k**.*

***NP**=Verificabile polinomialmente, ma non necessariamente polinomiale.*

Esempio

*Decrittazione **per ispezione esaustiva** della crittazione trasposizionale a blocchi T=a * Nblocco!*

Complessità degli algoritmi usati

Esercizi: Calcolare la complessità di:

- *Sostituzionale per traslazione (shift)*
- *Sostituzione traslazione + 1 carattere (break)*
- *Sostituzionale per permutazione*
- *Trasposizionale a blocchi:*
  - *Permutazioni cicliche*
  - *Permutazioni senza punti fissi*
  - *Permutazioni qualsiasi*

Esercizi Trasposizione semplice

- *Scytale*

- *Scytale riflesso (cambia poco)*

Trasposizione a Colonne

Blocchi di nxm (es. 5 x 3)

Si permuta l'ordine all'interno del blocco (n! poss.)

1 2 3 4 5 6 -> 3 1 4 6 5 (UNA QUALSIASI PERMUTAZIONE)

A_1	A_2	A_3	A_4	A_5
A_6	A_7	A_8	A_9	A_{10}
A_{11}	A_{12}	A_{13}	A_{14}	A_{15}

A_2	A_5	A_1	A_3	A_4
A_7	A_{10}	A_6	A_8	A_9
A_{12}	A_{15}	A_{11}	A_{13}	A_{14}

Si rilegge per colonne:

$A_1A_6A_{11}A_2A_7A_{12}A_3A_8A_{13}A_4A_9A_{14}A_5A_{10}A_{15}$

Messaggio Lezione

- *I nostri algoritmi di decrittazione scalano linearmente nella lunghezza del messaggio, ma non polinomialmente nella chiave.*
- *Le macchine di Touring possono eseguire le somme*
- *Le operazioni elementari si ottengono tutte tramite sequenze di NOR oppure di NEND*
- *Fate gli esercizi*