# Lezione 7A – modelli generativi

Lezione del 28/03/2023

# Grammatiche regolari: automi a stati finiti

- Resta da mostrare che la classe dei linguaggi regolari coincide con la classe dei linguaggi decisi da automi a stati finiti
- la dimostrazione procederà in 4 passi
  - intanto, poiché sappiamo che le computazioni di un automa a stati finiti terminano sempre, dimostreremo che la classe dei linguaggi regolari coincide con la classe dei linguaggi accettati da automi a stati finiti deterministici, poi
  - PASSO 1) dimostreremo che per ogni automa a stati finiti deterministico A esiste una grammatica $G_A$ tale che $L(A) = L(G_A)$ – precedente lezione
  - PASSO 2) per dimostrare l'inverso del PASSO 1 dovremo definire una classe più ampia di automi a stati finiti, gli *automi a stati finiti non deterministici*
  - PASSO 3) dimostreremo che per ogni grammatica G esiste un automa a stati finiti non deterministico $NA_G$ esiste tale che $L(G) = L(NA_G)$
  - PASSO 4) dimostreremo che per ogni automa a stati finiti non deterministico NA esiste un automa a stati finiti deterministico A tale che L(A) = L(NA)

# Grammatiche regolari e ASFD: PASSO 2

- Un **automa a stati finiti non deterministico (ASFND)** è una quintupla $\langle \Sigma, Q, q_0, Q_F, \delta \rangle$ in cui $\Sigma$, Q, $q_0$ e $Q_F$ hanno lo stesso significato che negli automi deterministici e $\delta: Q \times \Sigma \rightarrow \mathcal{P}(Q)$ è la funzione ***totale*** di transizione che, ad ogni coppia (stato, carattere) associa *un insieme* di nuovi stati possibili
  - analogamente al concetto di multi-quintupla nelle macchine di Turing non deterministiche
- Le definizioni di configurazione, transizione, funzione di transizione estesa e computazione sono analoghe a quelle del caso deterministico
- **Un ASFND A accetta una parola $x=x_1x_2...x_n$ se esistono n stati $q_1, q_2, ..., q_n$ tali che $q_1 \in \delta^*(q_0, x_1)$, $q_2 \in \delta^*(q_1, x_2)$, ..., $q_n \in \delta^*(q_{n-1}, x_n)$ e $q_n \in Q_F$**
  - ossia, esiste una sequenza deterministica di transizioni che termina con uno stato finale – analogamente alla definizione di accettazione per una macchina di Turing non deterministica
- il linguaggio accettato da un ASFND A = $\langle \Sigma, Q, q_0, Q_F, \delta \rangle$ è l'insieme delle parole accettate da A, ossia

# Grammatiche regolari e ASFD: PASSO 2

- Un **automa a stati finiti non deterministico (ASFND)** è una quintupla $\langle \Sigma, Q, q_0, Q_F, \delta \rangle$ in cui $\Sigma$, Q, $q_0$ e F hanno lo stesso significato che nella Macchina di Turing e $\delta: Q \times \Sigma \rightarrow \mathcal{P}(Q)$ è la funzione ***totale*** di transizione che, ad ogni coppia (stato, carattere) associa *un insieme* di nuovi stati possibili

- Oppure, utilizzando la funzione $\delta^*$:

  Un ASFND A accetta una parola x se $\delta^*(q_0, x) \cap Q_F \neq \emptyset$

- Con questa notazione:

  il linguaggio accettato da un ASFND A = $\langle \Sigma, Q, q_0, Q_F, \delta \rangle$ è l'insieme delle parole accettate da A, ossia

$$L(A) = \{ x \in \Sigma^*: \delta^*(q_0, x) \cap Q_F \neq \emptyset \}$$

# Grammatiche regolari e ASFD: PASSO 3

- **TEOREMA G.15**: per ogni grammatica $G = \langle V_T, V_N, P, S \rangle$ esiste un ASFND $A_G = \langle \Sigma, Q, q_0, Q_F, \delta \rangle$ tale che $L(G) = L(A_G)$
- Dimostrazione. Siano le produzioni di G nella forma $B_i \rightarrow a\,B_j \mid a$
  - 1) definiamo l'automa $A_G$:
    - poniamo $\Sigma = V_T$;
    - sia $V_N = \{B_0, \ldots, B_k\}$ con $S = B_0$: poniamo $Q = \{q_0, \ldots, q_k\}$;
    - poniamo $Q_F = \{q_F\}$;
    - per ogni $a \in \Sigma$ e $q_i, q_j \in Q$, se $B_i \rightarrow aB_j$ è in P poniamo $q_j \in \delta(q_i,a)$
    - per ogni $a \in \Sigma$ e $q_i \in Q$, se $B_i \rightarrow a$ è in P poniamo $q_F \in \delta(q_i,a)$
    - se L(G) contiene $\varepsilon$ e dunque $A_0 \rightarrow \varepsilon$ è in P poniamo $q_F \in \delta(q_0, \square)$

# Grammatiche regolari e ASFD: PASSO 3

- **TEOREMA G.15**: per ogni grammatica $G = \langle V_T, V_N, P, S \rangle$ esiste un ASFND $A_G = \langle \Sigma, Q, q_0, Q_F, \delta \rangle$ tale che $L(G) = L(A_G)$
- Dimostrazione. Siano le produzioni di G nella forma $B_i \rightarrow a\ B_j \mid a$
  - 2) Dimostriamo che $x \in L(G)$ se e soltanto se $x \in L(A_G)$:
    - $x = x_1x_2...x_n \in L(G) \Leftrightarrow$

      P contiene le produzioni $B_0 \rightarrow x_1\ B_1$ , $B_1 \rightarrow x_2\ B_2$ , ... , $B_{n-2} \rightarrow x_{n-1}\ B_{n-1}$ , $B_{n-1} \rightarrow x_n\ B \Leftrightarrow$

      Q contiene gli stati $q_0, q_0, ..., q_{n-2}, q_{n-1}$ tali che

      $q_1 \in \delta(q_0,x_1)$, $q_2 \in \delta(q_1,x_2)$, ... , $q_{n-1} \in \delta(q_{n-2},x_{n-1})$, $q_F \in \delta(q_{n-1},x_n) \Leftrightarrow$

      $x = x_1x_2...x_n \in L(A_G)$
- Siano le produzioni di G nella forma $B_i \rightarrow B_j\ a \mid a$
  - simile al caso precedente e lasciato per esercizio

QED

# Grammatiche regolari e ASFD: PASSO 4

- Resta da mostrare che automi deterministici e non deterministici decidono gli stessi linguaggi
- Poiché un ASFD è un particolare ASFND

  e, dunque, se un linguaggio è deciso da una ASFD è deciso anche da un ASFND

  quello che resta da dimostrare è solo il seguente teorema
- **TEOREMA G.16**: per ogni ASFND NA = $\langle \Sigma, Q, q_0, Q_F, \delta \rangle$ esiste un ASFD A = $\langle \Sigma, Q_D, q_{0D}, Q_{FD}, \delta_D \rangle$ tale che L(NA)=L(A)
- Idea della dimostrazione: sia dato NA; mostriamo soltanto la struttura di A

  Osserviamo che A opera sullo stesso alfabeto di NA; poi:

  - poniamo $Q_D = 2^Q$, ossia gli stati di A sono i sottoinsiemi degli stati di NA ; ossia, se indichiamo $Q_D = \{\omega_1, \ldots, \omega_H\}$, ciascun $\omega_i$ è un sottoinsieme di Q
  - poniamo $q_{0D} = \{q_0\}$;
  - poniamo $Q_{FD} = \{\omega_i \subseteq Q : \omega_i \cap Q_F \neq \emptyset\}$, ossia, gli stati finali di A sono i sottoinsiemi di Q che contengono uno stato finale di NA
  - per ogni $a \in \Sigma$ e $\omega \in Q_D$, poniamo $\delta_D(\omega,a) = \omega'$ tale che $\omega' = \bigcup_{q \in Q} \delta(q,a)$

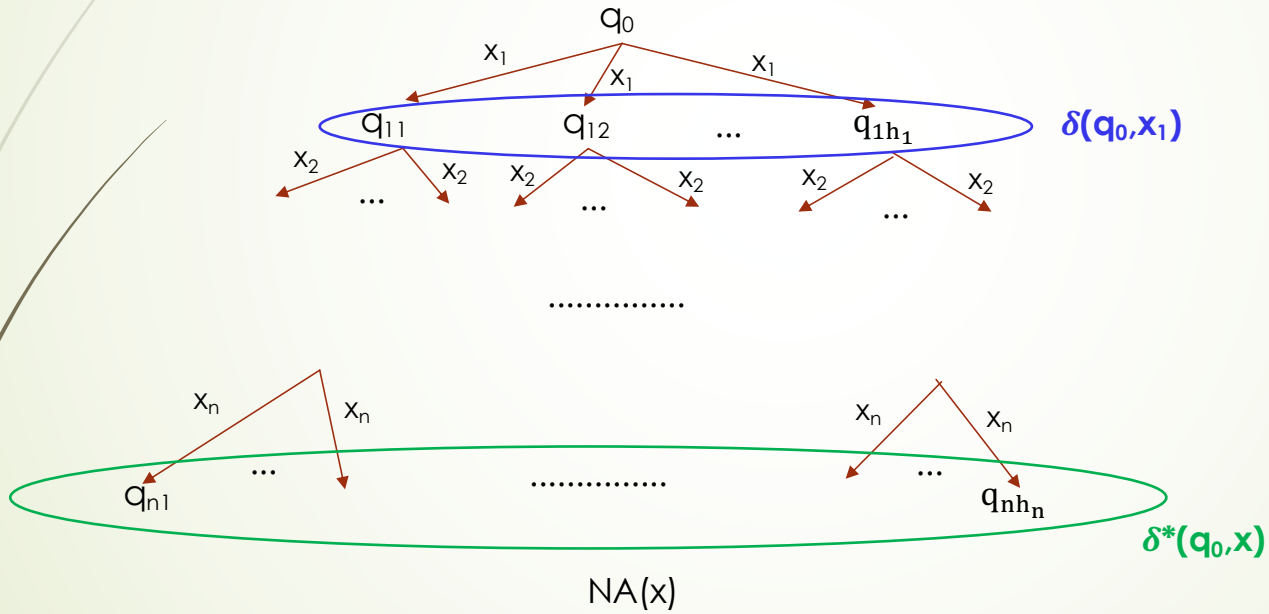
# Grammatiche regolari e ASFD: PASSO 4

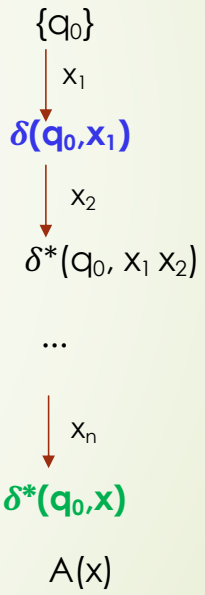
- **TEOREMA G.16**: per ogni ASFND NA = $\langle \Sigma, Q, q_0, Q_F, \delta \rangle$ esiste un ASFD A = $\langle \Sigma, Q_D, q_{0D}, Q_{FD}, \delta_D \rangle$ tale che L(NA)=L(A)
- Idea della dimostrazione. Dopo aver definito A resta da mostrare che

  $x \in L(NA)$ se e soltanto se $x \in L(A)$

  e lo vediamo informalmente, utilizzando la figura alla prossima pagina
  - sia $x = x_1 \ldots x_n$
  - La parte destra della figura illustra l'albero che corrisponde alla computazione non deterministica NA(x): i nodi dell'albero sono etichettati con gli stati interni di NA e gli archi con il carattere letto sul nastro. Le ellissi racchiudono l'insieme degli stati di NA che corrisponde allo stato di A utilizzato a quel punto della computazione
  - La parte destra della figura illustra la sequenza di transizioni che corrisponde alla computazione A(x): si osservi che lo stato in cui si trova A quando legge il carattere $x_1$ è $\delta(q_0,x_1)$, lo stato in cui si trova A quando legge il carattere $x_2$ è $\delta^*(q_0, x_1 x_2)$, ..., e per ogni i= 1, 2, ..., n lo stato in cui si trova A quando legge il carattere $x_i$ è $\delta^*(q_0, x_1 x_2 \ldots x_i)$
  - Dalla figura si evince che una computazione deterministica di NA(x) termina in uno stato finale se e soltanto se A(x) termina in uno stato finale

- A sinistra computazione NA(x) a destra computazione A(x) con $x = x_1 \ldots x_n$ : i nodi sono etichettati con gli stati interni di NA/A e gli archi con il carattere letto sul nastro. Le ellissi rappresentano gli insiemi di stati di NA che costituiscono gli stati di A



# Grammatiche regolari e ASFD: PASSO 4

- In conseguenza dei teoremi G.14-G.16 possiamo infine concludere che

**TEOREMA G.17: La classe dei linguaggi regolari coincide con la classe dei linguaggi decisi da automi a stati finiti deterministici**

# Grammatiche regolari: chiusura

- Ci occupiamo ora di studiare se la proprietà di essere regolari si trasporta all'unione, all'intersezione e al complemento
- **Unione di linguaggi regolari**: se $L_1$ e $L_2$ sono due linguaggi regolari definiti sullo stesso alfabeto $\Sigma$ allora $L = L_1 \cup L_2$ è regolare
  - per dimostrarlo ricorriamo alla decidibilità dei linguaggi regolari mediante ASFD
  - Sia $A_1 = \langle \Sigma, Q_1, q_{01}, Q_{F1}, \delta_1 \rangle$ l'automa che decide $L_1$ e sia $A_2 = \langle \Sigma, Q_2, q_{02}, Q_{F2}, \delta_2 \rangle$ l'automa che decide $L_2$
  - consideriamo l'ASFND $NA = \langle \Sigma, Q, q_0, Q_F, \delta \rangle$ tale che
    - $Q = Q_1 \cup Q_2 \cup \{q_0\}$
    - $Q_F = Q_{F1} \cup Q_{F2}$
    - per ogni $a \in \Sigma$, $\delta(q_0,a) = \{\delta(q_{01},a), \delta(q_{02},a)\}$
    - per ogni $a \in \Sigma$ e per ogni $q \in Q_1$, $\delta(q,a) = \delta_1(q,a)$
    - per ogni $a \in \Sigma$ e per ogni $q \in Q_2$, $\delta(q,a) = \delta_2(q,a)$
  - ESERCIZIO: dimostrare che NA decide L e, quindi, L è regolare

# Grammatiche regolari: chiusura

- Ci occupiamo ora di studiare se la proprietà di essere regolari si trasporta all'unione, all'intersezione e al complemento
- **Complemento di linguaggi regolari**: se L è un linguaggio regolare allora $L^C$ è regolare
  - di nuovo, per dimostrarlo ricorriamo alla decidibilità dei linguaggi regolari mediante ASFD
  - Sia $A = \langle \Sigma, Q, q_0, Q_F, \delta \rangle$ l'automa che decide L
  - consideriamo l'automa $A^c = \langle \Sigma, Q, q_0, Q\text{-}Q_F, \delta \rangle$ che differisce da A per la sola definizione dell'insieme degli stati finali:
    - gli stati finali di $A^c$ sono tutti e soli gli stati interni di A che non sono stati finali per A
  - ESERCIZIO: dimostrare che $A^c$ decide $L^c$ e, quindi, $L^c$ è regolare

# Grammatiche regolari: chiusura

- Ci occupiamo ora di studiare se la proprietà di essere regolare si trasporta all'unione, all'intersezione e al complemento
- **Intersezione di linguaggi regolari**: se $L_1$ e $L_2$ sono due linguaggi regolari allora $L = L_1 \cap L_2$ è regolare
  - infatti:
    - $L_1^C$ e $L_2^C$ sono regolari
    - allora, $L_1^C \cup L_2^C$ è regolare
    - allora, $(L_1^C \cup L_2^C)^C$ è regolare
  - e dunque, poiché $(L_1^C \cup L_2^C)^C = L_1 \cap L_2$, allora $L = L_1 \cap L_2$ è regolare

# Espressioni regolari

- Terminiamo la parte di corso dedicata alle grammatiche descrivendo uno strumento che permette di descrivere una particolare classe di linguaggi
- Dati un insieme di caratteri $\Sigma$, una parola r sull'alfabeto $\Sigma \cup \{+, *, (, ), \cdot, \emptyset\}$ è una ***espressione regolare*** se
  - 1) $r = \emptyset$ , oppure
  - 2) $r \in \Sigma$ , oppure
  - 3) date due espressioni regolari s e t, $r = (s + t)$ oppure $r = (s \cdot t)$ oppure $r = s^*$
- ESEMPIO: dato $\Sigma = \{a,b\}$, sono espressioni regolari le seguenti
  - $((a^* \cdot b) + a)$, infatti:
    - poiché a è una espressione regolare allora $a^*$ è una espressione regolare
    - allora, poiché b è una espressione regolare, $(a^* \cdot b)$ è una espressione regolare
    - allora, poiché a è una espressione regolare, $((a^* \cdot b)+a)$ è una espressione regolare
  - $(a + (b \cdot (a \cdot b)^*))$
    - dimostrare che è una espressione regolare

# Espressioni regolari

- Interpretando opportunamente i simboli +, * , · e ∅, e associando il significato di singleton (insieme costituito da un unico elemento) ai caratteri in Σ, è possibile utilizzare le espressioni regolari per definire linguaggi:
  - 1) l'espressione regolare ∅ definisce il linguaggio L = ∅
  - 2) per ogni a ∈ Σ, l'espressione regolare a definisce il linguaggio L = {a}
  - 3) detti L(s) e L(t) i linguaggi definiti dalle espressioni regolari s e t,
    - l'espressione regolare (s + t) definisce il linguaggio L(s) ∪ L(t)
      - ossia, + viene interpretata come operazione di unione di due parole
    - l'espressione regolare (s · t) definisce il linguaggio L={xy: x ∈ L(s) e y ∈L(t)}
      - ossia, le parole in L sono la concatenazione di una parola in L(s) e una parola in L(t)
    - l'espressione regolare r = s* definisce il linguaggio L contenente parole che sono la concatenazione di (0 o più) parole di L(s)
      - così come, ad esempio, Σ* è l'insieme delle parole che sono concatenazione di (0 o più) caratteri di Σ

# Espressioni regolari

- ESEMPIO: dato $\Sigma = \{a,b\}$, il linguaggio associato all'espressione regolare $((a^* \cdot b) + a)$ è $L = \{a^n b: n \geq 0\} \cup \{a\}$ , infatti
  - $a^*$ definisce la concatenazione di 0 o più a, ossia il linguaggio $\{\varepsilon, a, aa, aaa, ...\}$
  - allora, $(a^* \cdot b)$ definisce la concatenazione degli elementi di $a^*$ con b, ossia il linguaggio {b, ab, aab, aaab, ... }
  - allora, $((a^* \cdot b)+a)$ definisce l'unione fra $(a^* \cdot b)$ e il linguaggio {a} associato all'espressione regolare a, e quindi $((a^* \cdot b)+a)$ definisce il linguaggio {a, b, ab, aab, aaab, ... }
- ESERCIZI:
  - esplicitare il linguaggio definito da $(a + (b \cdot (a \cdot b)^*))$
  - esplicitare il linguaggio definito da $(b + (a \cdot (b \cdot a)^*))$
  - $(a + (b \cdot (a \cdot b)^*))$ e $(b + (a \cdot (b \cdot a)^*))$ definiscono lo stesso linguaggio?

# Espressioni regolari

- Resta da chiarire quale sia il potere espressivo delle espressioni regolari, ossia: possiamo definire un qualunque linguaggio mediante una espressione regolare?

- La risposta alla precedente domanda è no.

- Infatti, si può dimostrare il seguente teorema

**TEOREMA G.18**: un linguaggio è generato da una grammatica di tipo 3 se e soltanto se esso è definito da una espressione regolare

# Espressioni regolari

- Il Teorema G.18 mostra che la classe dei linguaggi generati da grammatiche regolari coincide con la classe dei linguaggi definiti da espressioni regolari

- D'altra parte, il Teorema G.17 mostra che la classe dei linguaggi generati da grammatiche regolari coincide con la classe di linguaggi decisi da automi a stati finiti

- In conclusione possiamo definire indifferentemente i linguaggi regolari come
  - i linguaggi generati da grammatiche di tipo 3
  - i linguaggi decisi da automi a stati finiti
  - i linguaggi definiti da espressioni regolari

# E con questo termina la parte di corso dedicata alla Calcolabilità

all'interno della quale abbiamo caratterizzato la classe dei linguaggi accettabili $\mathscr{A}$, la classe dei linguaggi decidibili $\mathscr{D}$, e, mediante la gerarchia di Chomsky, abbiamo anche individuato una struttura all'interno di $\mathscr{D}$ che possiamo così riassumere:

$$\mathbf{G3} \subset \mathbf{G2} \subset \mathbf{G1} \subseteq \mathscr{D} \subset \mathbf{G0} = \mathscr{A}$$

# Complessità: si parte!

- Abbiamo studiato cosa si intende per problema risolvibile
  - o meglio, per linguaggio decidibile
  - o linguaggio accettabile
  - o funzione calcolabile
- E abbiamo visto che esistono problemi non risolvibili.
- Ma anche (va da sé) problemi risolvibili.

- Uhmmm… Siamo davvero sicuri di poter risolvere i problemi risolvibili?

# La Torre di Hanoi

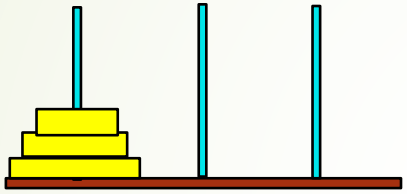
- *Narra la leggenda che, in un tempio nascosto nella foresta vicino ad Hanoi, sia custodito un grande piatto di ottone dal quale partono tre aste verticali di diamante. All'inizio dei tempi, su una delle tre aste vennero impilati da Brahma 64 dischi d'oro, di grandezze diverse gli uni dagli altri, a formare una torre: alla base era posto il disco più grande, sopra di esso quello immediatamente più piccolo, e così via, fino alla sommità della torre costituita dal disco più piccolo di tutti gli altri. Dall'inizio dei tempi, compito dei monaci è, semplicemente, spostare la torre dall'asta sulla quale Brahma l'aveva impilata ad un'altra asta. Due regole i monaci sono tenuti a rispettare: un solo disco alla volta può essere spostato da un'asta all'altra, e mai un disco può essere appoggiato su un disco più piccolo. Secondo la leggenda, quando i monaci termineranno il loro compito, quando, cioè, l'ultimo disco sarà finalmente piazzato a formare di nuovo la torre su un'asta diversa da quella sulla quale Brahma l'ha posta, allora arriverà la fine del mondo e tutto si trasformerà in polvere.*
- *Dobbiamo preoccuparci? Beh, prima di farlo, cerchiamo almeno di capire come funziona lo spostamento di una torre.*
  - Da "Il sentiero dei problemi impossibili", di cui sono autrice, Franco Angeli ed.

# La Torre di Hanoi

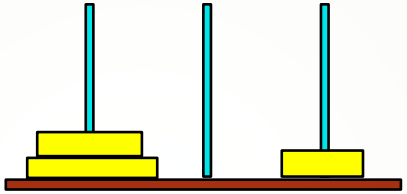
- Consideriamo una torre di 3 soli dischi impilata, diciamo, sull'asta a sinistra: l'obiettivo è spostarla sull'asta a destra
  - l'asta centrale avrà la funzione di "asta d'appoggio".
- Portiamo a termine il compito eseguendo le seguenti mosse (fig. nella prossima pag.):
  - 1) poiché possiamo spostare un solo disco alla volta, spostiamo il disco più piccolo sull'asta a destra;
  - 2) ora possiamo spostare il disco di grandezza intermedia e, poiché non possiamo appoggiarlo sul disco più piccolo, lo impiliamo nell'asta al centro;
  - 3) a questo punto, spostiamo il disco più piccolo sull'asta al centro, appoggiandolo sul disco di grandezza intermedia;
  - 4) spostiamo il disco più grande sull'asta a destra;
  - 5) spostiamo il disco più piccolo dall'asta centrale sull'asta a sinistra ;
  - 6) spostiamo il disco di grandezza intermedia sull'asta a destra, appoggiandolo sul disco più grande
  - 7) Infine, spostiamo il disco più piccolo sull'asta a destra, appoggiandolo sul disco di grandezza intermedia: fatto!

# La Torre di Hanoi (3 dischi)

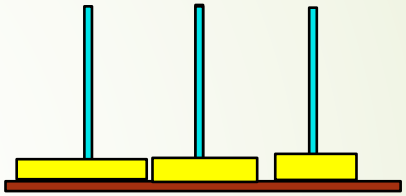
1)

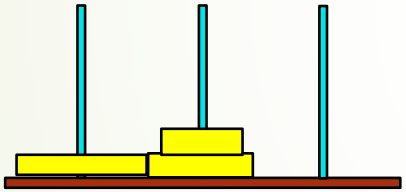


2)

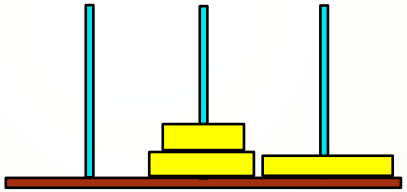


3)

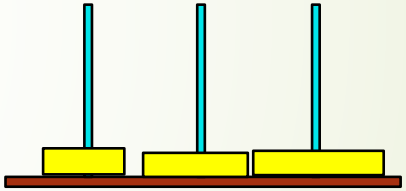




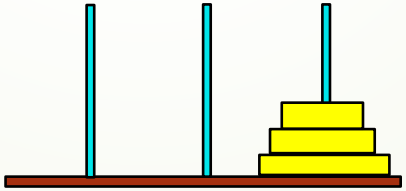
4)



5)



6)



7)

# La Torre di Hanoi

- Dunque, abbiamo spostato una torre di 3 dischi utilizzando 7 spostamenti di dischi singoli
  - e non è possibile realizzare il nostro compito utilizzando un numero inferiore di spostamenti di dischi singoli.
- Per spostare una torre di 4 dischi è necessario:
  - spostare la sotto-torre costituita dai 3 dischi più piccoli dall'asta di sinistra a quella centrale,
  - poi spostare il disco più grande sull'asta di destra ,
  - e, infine, spostare la sotto-torre costituita dai 3 dischi più piccoli dall'asta centrale a quella di destra.
  - E non possiamo far di meglio!

# La Torre di Hanoi

- Questo procedimento è generalizzabile
- Per spostare una torre di n dischi è necessario:
  - spostare la sotto-torre costituita dagli n-1 dischi più piccoli dall'asta di sinistra a quella centrale (configurazione 4) nella figura),
  - poi spostare il disco più grande sull'asta di destra (configurazione 5) nella figura),
  - e, infine, spostare la sotto-torre costituita dagli n-1 dischi più piccoli dall'asta centrale a quella di destra (configurazione 7) nella figura).
  - E non possiamo far di meglio!
- Quindi, se indichiamo con M(n) il numero di spostamenti di dischi singoli *necessario* a spostare una torre di n dischi, vale la seguente relazione di ricorrenza:

$$M(n) = 2\,M(n-1) + 1$$

- Che ha come soluzione $M(n) = 2^n - 1$
- E non possiamo far di meglio!

# La Torre di Hanoi

- Quindi, se indichiamo con M(n) il numero di spostamenti di dischi singoli *necessario* a spostare una torre di n dischi, abbiamo che
  $$M(n) = 2^n-1$$
- E non possiamo far di meglio!
- Ma che significa?
- Che per spostare la Torre di Hanoi occorrono (sono necessari)
  $$2^{64} - 1 = 18.446.744.073.709.551.615$$
  spostamenti di dischi
- e che, anche se i monaci riuscissero a spostare un disco in 1 secondo, occorrerebbero almeno 18.446.744.073.709.551.615 secondi per spostare la torre
- che corrispondono a circa 5.845.580.504 secoli
- un tempo così lungo che quando il sole diverrà una gigante rossa e brucerà la Terra, il gioco non sarà ancora stato completato.
- ...

# E allora?

- Intanto, possiamo stare ragionevolmente tranquilli: se sarà fine del mondo, non sarà per colpa dei monaci di Hanoi…
- Ma, soprattutto: lo sappiamo risolvere o no il problema della torre di Hanoi?
- Certo, che lo sappiamo risolvere!
  - vi ho mostrato il procedimento che sposta una torre di n dischi da un'asta all'altra!
- Tuttavia…
- Tuttavia, anche se *sappiamo come fare* a spostare una torre grande quanto ci pare, se la torre è abbastanza grande l'intera nostra vita non sarà sufficiente a vedere la torre spostata
- Se il tempo necessario a calcolare la soluzione di (un'istanza di) un problema è troppo elevato, saper calcolare quella soluzione è equivalente a non saperla calcolare
- …

# La Teoria della Complessità Computazionale

- Studia la "quantità di risorse" **necessarie** a risolvere un problema
  - meglio: a decidere un linguaggio
- E suddivide i problemi in "trattabili" e "intrattabili"
  - dipendentemente dal fatto che la "quantità di risorse" necessarie cresca come un polinomio o più di un polinomio
- Ma perché la crescita polinomiale è discriminante fra trattabilità e intrattabilità?
  - Beh, lo avete visto quanto è grande $2^{64}$: un numero di 20 (venti!) cifre. Invece, $64^2$ è il minuscolo 4096. Piccolo.
  - Chiara l'idea?
- Una funzione più che polinomiale cresce infinitamente più velocemente di una funzione polinomiale!
  - e, se quella funzione rappresenta la "quantità di risorse" necessaria a risolvere un problema…

# La Teoria della Complessità Computazionale

- Sì, ma qui stiamo parlando di funzioni che rappresentano la "quantità di risorse" necessaria a risolvere un problema
- Ma qual è *l'argomento* di queste funzioni?
  - Cioè: *in funzione di cosa* esprimiamo la complessità di un problema?
- E, poi, quali sono le "risorse" che prendiamo in considerazione?

- Che dire? La risposta nelle prossime puntate…