



Analisi della complessità di calcolo

Esercizio 1. Quali delle seguenti affermazioni sono vere?

1. Un algoritmo $\Theta(n)$ è $O(n)$
2. Un algoritmo $\Theta(n)$ è $O(n^2)$
3. Un algoritmo $\Theta(n^2)$ è $O(n^3)$
4. Un algoritmo $\Theta(n)$ è $O(1)$
5. Un algoritmo $O(1)$ è $\Theta(1)$
6. Un algoritmo $O(n)$ è $\Theta(1)$

1) ✓
2) ✓
3) ✓
4) F
5) ✓
6) F

Esercizio 2. Si consideri la seguente definizione ricorsiva per il calcolo del fattoriale di un numero. Analizzare la complessità dell'algoritmo in funzione delle dimensioni dell'input.

```
int find_factorial(int n)
{
    //Factorial of 0 is 1
    if(n==0)
        return(1);

    //Function calling itself: recursion
    return(n*find_factorial(n-1));
}
```

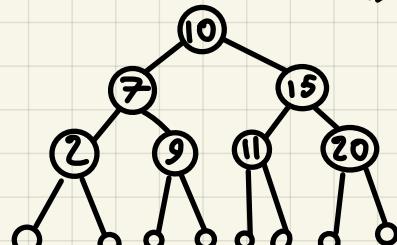
$$O(n) = \underbrace{C + C}_{\text{IF RET}} + O(n-1) = 2 + 2 + O(n-2)$$

L'ISTRUZIONE PIÙ FREQUENTE È IF($n=0$) RETURN (1);
CHE SI RIPETE $n-1$ VOLTE

Esercizio 3. Provare che il problema della ricerca in un insieme ordinato di n elementi ha costo $\Omega(\log n)$ se si utilizzano confronti. (Sugg. utilizzare un ragionamento analogo a quanto fatto per l'ordinamento. Rappresentare il problema come un albero la cui profondità rappresenta il costo nel caso peggiore di un algoritmo basato su confronti. Valutare la profondità dell'albero in funzione di n .

2 7 9 10 11 15 20

LIV 0 → 1 ELEM
LIV 1 → 2 ELEM
LIV 2 → 4 ELEM
LIV 3 → 8 ELEM



IL NUMERO DI ELEM PER RIGA = 2^h , h RIGA
ELEM TOTALI $n = 2^{h_{\max}+1} - 1$
 $\log_2 n+1 = h+1$
 $h = \log_2(n+1) - 1 \approx O(\log n)$

Esercizio 4. Per $n > 1$, sia $T(n)$ il numero esatto di esecuzioni del comando

$a = a + 2$

nel seguente frammento di codice:

```
for i = 1 to n - 1 do
    for j = n - i + 1 to n do
        a = a + 2
```

- i) Esprimere $T(n)$ utilizzando la notazione $O(n)$.
- ii) Calcolare il valore $T(10)$ oppure un valore maggiore di $T(10)$ che sia il più possibile vicino al valore esatto.

Esercizio 6. Per $n > 0$, si consideri una ricorrenza $T(n)$ il cui albero delle chiamate è binario e pieno (cioè; ogni nodo interno ha due figli) con $2n$ foglie, ciascuna foglia associata a costo di calcolo 1 e con ogni nodo interno che contribuisce 2 al costo.

a) Si determini il valore $T(10)$.

b) Nel caso che ciascuna foglia associata abbia costo di calcolo 1 e che ogni nodo interno che contribuisce 3 al costo quanto vale il Valore $T(10)$?

$$T(n) = 2(\text{COSTO NODI INTERNI}) + \text{COSTO FOGLIA} = 2(2n-1) + 2n = 6n-2$$

SE ALBERO BINARIO COMPLETO, # NODI = # FOGLIE = $2n-1$

a) $T(10) = 60 - 2 = 58$

b) $T(n) = 3(2n-1) + 2n = 8n-3$

$$T(10) = 80 - 3 = 77$$

INTRO COMPLESSITÀ

Nell'analisi di complessità e, in particolare nella notazione $O()$, si ignorano le costanti moltiplicative e additive. Discutere questa scelta indicandone i vantaggi e gli svantaggi.

PER CALCOLARE L'EFFICIENZA DI UN PROGRAMMA UBASIAMO SULLA NOTAZIO. O GRANDE.

POICHE LA VELOCITÀ DI UN PROG. DI PENE DA FATTORI COME LA CPU, IL DISCO, I DATI ECC.. ELIMINIAMO QUESTI LIMITI CERCANDO UNA STIMA ASINTOTICA DEL COSTO (INDIPENDENTE DAL LING. DI PROG.) AL PIÙ LE DIFF TRA DUE PROGRAMMI SCRITTI DA DUE LINGUAGGI DIVERSI SONO DI COSTANTI MOLTIPLICATIVE.

LO SVANTAGGIO È CHE SE NO 2 PROGRAMMI $O(10n)$ E $O(10000n)$ ENTRAMBI SARANNO $O(n)$, PERDENDO QUINDI LA PRECISIONE

Cosa si intende quando diciamo che un algoritmo A è asintoticamente più efficiente dell'algoritmo B?

1. A è sempre meglio di B per tutti gli input
- ✓ 2. A è sempre meglio di B per input di grandi dimensioni
3. A è sempre meglio di B per input di piccole dimensioni
4. B è sempre meglio di A per input di piccole dimensioni

SE NO $100 \times n$ E $10 \times n^2$ NOTIAMO CHE PER INPUT PICCOLI DI n NON NOTIAMO TOLTE DIFFERENZE ($n=1$).

PER VALORI GRANDI I RISULTATI DIFFERENZIANO DI TOLTO ($n=100$)

Verificare e motivare la correttezza o meno delle seguenti affermazioni.

- a) $(n!) \in \Omega(n^2)$ (Corretto)
- (b) $n^{3/2} \in O(n \log n)$ (Errato)
- (c) $2^{(\log_2 n)} \in \Theta(n)$ (Corretto)

a) $(n!) \in \Omega(n^2)$ $n! > n^2$

$\Omega(n^2)$ È IL LIMITE INFERIORE DI $(n!)$, CUORE IL COSTO MINORE DEL CASO PEGGIOR

b) $n^{3/2} \in O(n \log n)$ $n^{3/2}$ E $n \log n$ HANNO COMPORTAMENTI ASINTOTICI DIVERSI $\rightarrow n\sqrt{n} \in n(\log n)$, $\sqrt{n} \in \log n$ CRESCONO IN MODO DIVERSO

c) $2^{\log_2 n} \in \Theta(n)$

$$2^{\log_2 n} = n \in \Theta(n)$$

Spiega la validità della seguente espressione e illustrare la differenza fra costo di un programma e andamento asintotico rappresentato con la notazione O():
 Se $f(n) \leq g(n)$ allora vale la seguente uguaglianza. $O(f(n) + g(n)) = O(g(n))$

**IL COSTO È IL NUMERO DI ISTRUZIONI ESEGUITE DI UN PROG
ASSEGNAVANDO UN VALORE AD OGNI ISTRUZIONE TIPO**

**IL COSTO ASINTOTICO È MENO PRECISO POICHÉ NON DIPENDE DA
FATTORI INTERNI DEL CALCOLATORE**

$O(f(n) + g(n)) = O(g(n))$ SIGNIFICA CHE $f+g$ "CRESCÈ AL PIÙ
COME g " POICHÉ f È MINORE DI g ASINTOTICAMENTE

Per $n > 1$, sia $T(n)$ il numero esatto di esecuzioni del comando $a = a + 2$ nel seguente frammento codice:
 for $i = 1$ to $n - 1$ do

for $j = n - i + 1$ to n do
 $a = a + 2$

Utilizzando la notazione O() si esprima il costo di $T(n)$ in funzione di n ; si calcoli il valore esatto di $T(n)$ per $n=10$.

$$T(n) = \sum_{i=1}^{n-1} \sum_{j=n-i+1}^n 1 = \sum_{i=1}^{n-1} i - 1 = \sum_{i=1}^{n-1} i - \sum_{i=1}^{n-1} 1 = \frac{(n-1)n}{2} - (n-1) = \frac{n^2-n}{2} - (n-1)$$

\downarrow

$$\sum_{k=1}^n k = \frac{n \cdot (n+1)}{2}$$

$i(i-1)$ PER FAR ARRIVARE j A n
 SOMMO $A j (i-1)$

RIS = 0
 FOR ($i=1$; $i \leq n-1$; $i++$)
 RIS += 1

$$T(10) = \frac{100 \cdot 10}{2} - 9 = 45 - 9 = 36$$

Ripetere l'esercizio per il programma seguente

```
int i, j, k = 0;
for (i = n / 2; i <= n; i++) {
    for (j = 2; j <= n; j = j * 2) {
        k = k + n / 2;
    }
}
```

$$T(n) = \sum_{i=n/2}^n \sum_{j=1}^{\log_2 n} 1 = \frac{n}{2} (\log_2 n) = O(n \log n)$$

$$T(10) = 5 \cdot \lfloor \log_2 10 \rfloor$$

Esprimere il costo asintotico dell'algoritmo usuale per la moltiplicazione di due numeri di n bit. Assumi che ciascuna operazione fra due bit abbia costo unitario. Riportiamo un esempio

11.....101 x

n bit moltiplicando

1.....11

n bit moltiplicatore

1101 +

n operazioni

..... 1010 +

n+1 operazioni

.....00 +

}

...

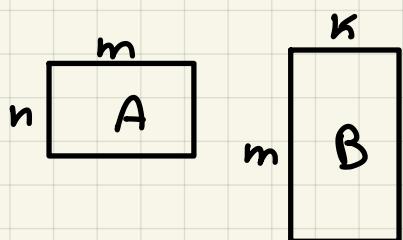
.....00.....0

2n bit nel caso peggiore

Somma di n
valori parziali

$$n \text{ RIGHE} \times n \text{ MOLTIPLICAZIONI} = n^2$$

Quante operazioni sono necessarie con l'usuale algoritmo per eseguire la moltiplicazione di una matrice (n x m) per una matrice (m x k)?



$$C_{ij} = \sum_{h=1}^m A_{ih} \cdot B_{hj}$$

sono NECESSARIE m OPERAZIONI PER C_{ij}

Analizza il numero di istruzioni eseguito dal seguente frammento programma per il calcolo del numero di Fibonacci ed esprimere il costo come funzione di n. Se teniamo conto delle dimensioni dell'input il programma è polinomiale?

```
int fibonacci(int n) {
    if(n == 0){
        return 0;
    } else if(n == 1) {
        return 1;
    } else {
        return (fibonacci(n-1) + fibonacci(n-2));
    }
}
```

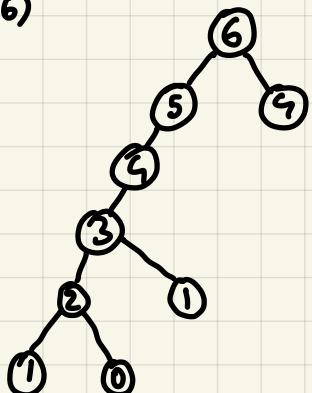
DEF FIB(n):

IF n>1:

RETURN FIB(n-1)+FIB(n-2)

RETURN n

FIB(6)



Sia dato un array ordinato A che memorizza solo 0 e 1, tale che ogni zero appare in una componente prima di componenti che memorizzano 1; in altre parole l'array è del tipo {0, 0, 0, ..., 0, 0, 1, 1, ..., 1, 1, 1}. Progetta un algoritmo che trova la componente più piccola tale che $A[i] = 1$ e abbia costo $O(\log n)$ nel caso peggiore (n rappresenta il numero degli elementi dell'array).

RICERCA BINARIA DI 0 E 1 CHE SI FERMA QUANDO TROVA UN 1 ANTICIPATO DA UNO 0

COMPUTABILITÀ

Enunciare e discutere la tesi di Church-Turing, chiarendo perché viene formulata come tesi e non come teorema.

"È CALCOLABILE TUTTO, UÒ CHE PUÒ ESSERE CALCOLATO DA UNA MACCHINA DI T"

È UNA TESI POICHÉ NON POSSIAMO ESCLUDERE CHE NEL FUTURO ESISTERANNO MACCHINE PIÙ POTENTI CHE RISOLVANO PROBLEMI INDEGIBILI

IL MODELLO DI CHURCH AFFERMA CHE OGNI FUNZIONE CHE POSSA ESSERE CALCOLATA DA UN ALG PUÒ ESSERE DEFINITA USANDO IL λ -CALCOLO, CON L'OBBIETTIVO DI TROVARE UNA MACCHINA PER L'ALG. UNIVERSALE. POICHÉ LA SUA E QUELLA DI TURING SONO EQUIVALENTI

Definire il concetto di numerabilità (o equivalentemente contabilità) e mostrare utilizzando la tecnica della diagonalizzazione che l'insieme dei linguaggi sull'alfabeto {0, 1} non è numerabile.

UN INSIEME È NUMERABILE QUANDO È UNA CORRISPONDENZA BIUNIVOCÀ TRA \mathbb{N} E L'INSIEME, QUANDO È FINITO O INFINTO NUMERABILE

	L ₁	L ₂	L ₃	L ₄	L ₅	⋮	⋮	L _n
	0 0 1 0 1	0 1 0 1 0	1 0 1 1 0	1 1 0 0 1	0 0 1 0 0			
	0	1	0	0	1			
	0	0	1	1	0			
	1	0	0	1	0			
	0	1	1	0	1			
	1	0	1	0	0			
	0	0	0	1	0			
	1	0	0	1	0	⋮	⋮	

NON È NUMERABILE POICHÉ SE IO TROVO UN NUMERO CHE DIFFERISCE PER OGNI CIFRA DAGLI ELEMENTI DELLA DIAGONALE SIGNIFICA CHE IL NUMERO CHE FA PARTE DEL LINGUAGGIO NON È PRESENTE NELLA TABELLA

Assumi che i linguaggi B e C siano decidibili (cioè dato x esistono due algoritmi A1 e A2 che decidono se x appartiene a B e se x appartiene a C rispettivamente).

Mostrare che anche

$B \cup C$ (linguaggio unione), $B \cap C$ (linguaggio intersezione),

$B \setminus C$ (linguaggio differenza formato dalle stringhe che appartengono a B ma non a C)

B^* sono linguaggi decidibili.

Suggerimento: Ricorda che dato A1 segue che $A1(x) = \text{vero}$ se x appartiene a B falso altrimenti

Analogamente per A2 e C. Combinando opportunamente si mostrano le affermazioni

$$B - C = \{x \in B \mid x \notin C\}$$

$$B^* = \{x \notin B\}$$

$$B \cup C = \{x \in B \vee x \in C\}$$

DECIDIBILE POICHÉ SAPPIAMO

$$B \cap C = \{x \in B \wedge x \in C\}$$

SE APPARTIENE O NO

Mostrare che l'insieme delle Macchine di Turing è numerabile.

POICHÉ LE COMPONENTI DI UNA MT SONO FINITI:

- ALFABETO DI CARATTERI FINITO SUL NASTRO Σ
- INSIEME FINITO S DI STATI
- UNO STATO INIZIALE E 2 SPECIALI: ACCETTA, RIFIUTA IN S

Quante sono le possibili macchine di Turing con alfabeto binario e con 4 stati?

$$\Sigma : 0, 1$$

$$S : S_0, S_1, S_2, S_3$$

$$\text{STATO} \times \Sigma \times (\text{DX} \circ \text{SX})$$

4	2	2

$$16 \cdot 8 \text{ bit}$$

$$128 \cdot 4 \text{ (STATI INIZIALI)} \cdot 16 \text{ (POSSIBILI STATI FINALI)}$$

N-P COMPLETEZZA

Definire le classi: P, NP e indicare le relazioni di contenimento fra le classi

LA CLASSE P È L'INSIEME DEI LING. L PER I QUALI È UNA MT CON UN SOLO NASTRO CHE DECIDE L, PER CUI IL # DI PASSI DI CALCOLO SU INPUT DI DIM. n È $O(n^k)$, $k \geq 0$

LA CLASSE NP È L'INSIEME DEI LING L PER CUI È UNA MT N.D. CHE IN TEMPO POLINOMIALE DECIDE SE $x \in L$ O NO

$P \subseteq NP$ POICHÉ LA MTD È UN CASO PARTICOLARE DELLA MTD

Definire la classe dei problemi NP-completi e indicare le relazioni di contenimento rispetto alle classi P e NP

L È NP-COMPLETO SE L ∈ NP E SE OGNI ALTRO LNU. IN NP È POLINOMIALMENTE RIDUCIBILE A L

$$P \cap NP-C = \emptyset \quad P \subseteq NP \wedge NP-C \subseteq NP$$

- a) Indicare un possibile risultato che permetterebbe di stabilire che $P \neq NP$
DOVREI TROVARE UN LOWER-BOUND PER UN PROB. NP PIÙ
UN POLINOMIALE
- b) Indicare un possibile risultato che permetterebbe di stabilire che $P = NP$
DATO UN NP-COMPLETO DOVREI TROVARE UN UPPER-BOUND
POLINOMIALE IN MODO DA DIR CHE QUESTO CONTIENE IN P
- c) Per dimostrare che un problema A è NP-completo cosa è necessario dimostrare?
- d)

Esercizi

Esercizio 2.1. Descrivere una grammatica regolare che generi il linguaggio costituito da tutte e sole le stringhe binarie contenenti un numero pari di simboli 1.

$$A \rightarrow 0A \mid 1B \mid 0 \quad B \rightarrow 0B \mid 1A \mid 1$$

Esercizio 2.2. Descrivere una grammatica regolare che generi il linguaggio costituito da tutte e sole le stringhe binarie contenenti almeno due simboli 0 e almeno un simbolo 1 è regolare.

Esercizio 2.3. Dimostrare che il linguaggio costituito da tutte e sole le stringhe binarie contenenti un numero pari di simboli 0 oppure esattamente due simboli 1 è regolare.

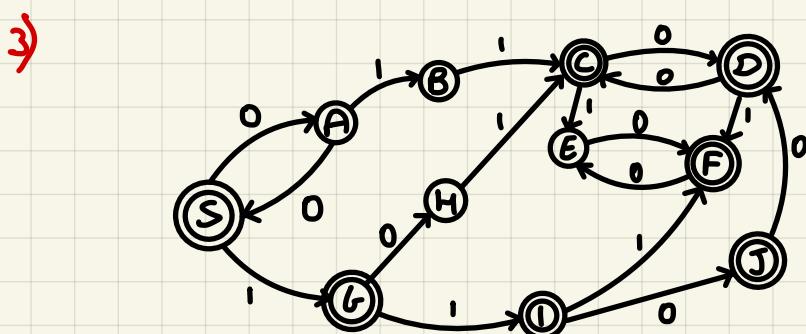
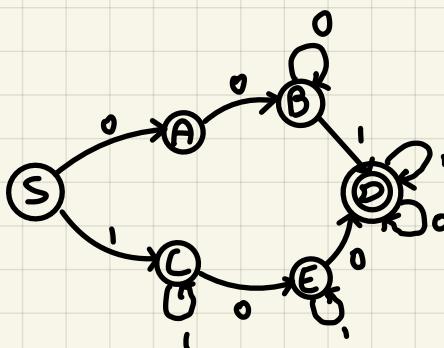
Esercizio 2.4. Descrivere una grammatica regolare che generi il linguaggio costituito da tutte e sole le stringhe binarie contenenti almeno tre simboli 1.

Esercizio 2.5. Descrivere una grammatica libera da contesto che generi il linguaggio costituito da tutte e sole le stringhe binarie del tipo 0^n12^n (n 0 seguiti da un 1 seguito da n 2).

Esercizio 2.6. Descrivere una grammatica libera da contesto che generi il linguaggio costituito da tutte e sole le stringhe binarie contenenti lo stesso numero di 1 e di 0.

Esercizio 2.7. Completare la dimostrazione del Teorema ??.

2) $S \rightarrow 0A \mid 1C \quad D \rightarrow 1D \mid 0D \mid \epsilon$
 $A \rightarrow 0B \mid 1E \quad C \rightarrow 0E \mid 1C$
 $B \rightarrow 0B \mid 1D \quad E \rightarrow 0D \mid 1E$



4)

$S \rightarrow 0S \mid 1A$
 $A \rightarrow 0A \mid 1B$
 $B \rightarrow 0B \mid 1C$
 $C \rightarrow 0C \mid 1C \mid 1 \mid 0 \mid \epsilon$

5) $s \rightarrow 0S2 \mid 1$

6) $s \rightarrow 01S \mid S01 \mid 0S1 \mid 1S0 \mid 10S \mid S10 \mid \epsilon$

Descrivere le fasi di analisi lessicale e di analisi sintattica di un compilatore; per ciascuna delle due fasi descrivi l'input e l'output e come viene elaborato il programma.

LE FASI DI ANALISI LESSICALE E SINTATTICA SI PENDONO SOLTANTO DALLA SINTASSI DEL LINGUAGGIO SORGENTE

- LA FASE LESSICALE PRENDE IN INPUT UNA STRINGA, CONTROLLO SE CI SONO ERRORE ORTOGRAFICI, E DA IN OUTPUT LA STRINGA SOTTO FORMA DI SEQUENZA DI TOKEN
- L'ANALISI SINTATTICA PRENDE IN INPUT LA SEQUENZA DI TOKEN, VERIFICA SE APPARTIENE AL LINGUAGGIO GENERATO DALLA GRAMMATICA G, CONTROLLO SE LA STRINGA È GIUSTA SEGUENDO DELLE REGOLE SINTATTICHE (SOGL + VERBO + COMPL) E RESTITUISCE L'ALBERO SINTATTICO PER LA SEQUENZA DI INPUT

Illustrare la gerarchia di Chomsky delle grammatiche; indicare per ciascuna categoria la proprietà/caratteristica a vostro giudizio maggiormente rilevante.

ABBIANO 4 TIPOLOGIE DI GRAMMATICHE GENERATIVE:

- TIPO 3 O REGOLARE: OGNI PRODUZIONE È LINEARE A DESTRA
 $A \rightarrow \alpha B, A \rightarrow \alpha$ (AUTOMI A STATI FINITI)
- TIPO 2 O LIBERE DA CONTESTO: OGNI PRODUZIONE DEVE ESSERE DEL TIPO $A \rightarrow \alpha$
- TIPO 1 O CONTESTUALI: OGNI PRODUZIONE DEVE ESSERE DEL TIPO $\alpha \rightarrow \beta$ confliziali
- TIPO 0 O NON LIMITATE: NON ABBIANO NESSUN VINCITO SULLE PRODUZIONI

Descrivi l'algoritmo per eliminare le ricorsioni sinistre di una grammatica di tipo 2.

A NON TERMINALE A NELLA GRAMMATICA CHE PRESENTA ALMENO UNA PROD RICORS. A SX IMMEDIATA, ESEGUIAMO:

- 1) SEPARIAMO LE RICORSIONI A SX IMMEDIATE DALLE ALTRE
- 2) INTRODUCCIAMO UN NUOVO TERMINALE A'
- 3) SOSTITUIAMO OGNI PROD NON RICORSIVA $A \rightarrow \delta_i$ CON $A \rightarrow \delta_i A'$
- 4) SOSTITUIAMO OGNI PROD RICORSIVA $A \rightarrow A \alpha_i$ CON $A' \rightarrow \alpha_i A'$
- 5) AGGIUNGIAMO $A' \rightarrow \lambda$

Descrivi il linguaggio generato dalla seguente grammatica (S assioma, S e B simboli non terminali)

$S \rightarrow abc$

$S \rightarrow aSBc$

$cB \rightarrow Bc$

$bB \rightarrow bb$

$S \rightarrow aSBc \rightarrow a a SBc Bc \rightarrow a a a SBc Bc Bc \rightarrow a a a a b c B c B c B c$
 $\rightarrow a a a a b B c B c B c \rightarrow a a a a b b c c B c B c \rightarrow a a a a b b c B c c B c$
 $\rightarrow a a a a b b b b c c c \rightarrow a^n b^n c^n$

Nella gerarchia di Chomsky di che tipo è la grammatica precedente? Prova a trovare una grammatica di tipo 3 che genera lo stesso linguaggio.

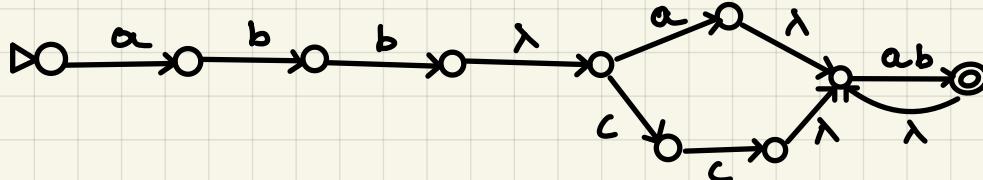
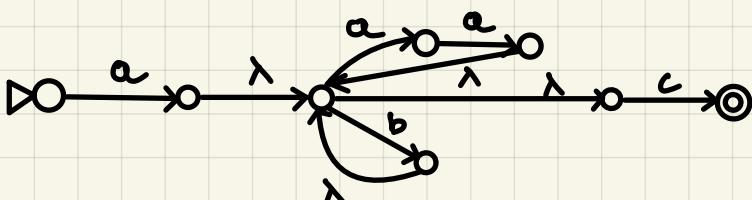
NON SI PUÒ FARE, SI DEVE CONTARE (TIPO 1)

Si considerino le seguenti espressioni regolari sull'alfabeto {a,b,c}:

(1) $R1 = a (b \mid aa)^* c$

(2) $R2 = abb (a \mid cc) (ab)^*$

Per ciascuna costruire un automa a stati finiti deterministico o nondeterministico che riconosce il linguaggio definito dall'espressione regolare.



Data una Grammatica regolare G, è possibile stabilire se G genera il linguaggio vuoto? Come?

Data una Grammatica regolare G, è possibile stabilire se G genera un linguaggio finito o infinito? Come?

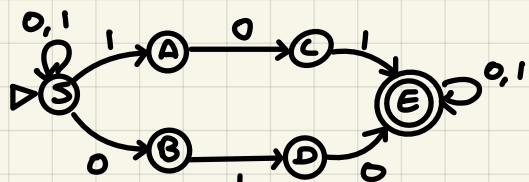
(Sugg. Per i due esercizi precedenti utilizzare il fatto che per ogni grammatica regolare G esiste un automa a stati finiti che riconosce tutte e sole le stringhe che appartengono a G)

a) **✓ INPUT NON ARRIVA UNO STATO ACCETTANTE**

b)

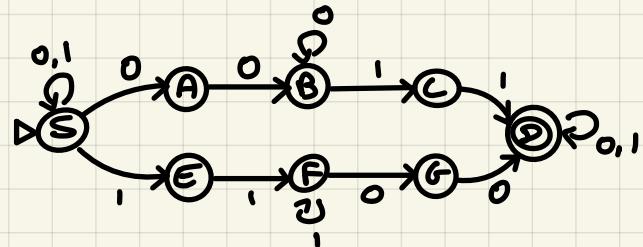
- Per i seguenti linguaggi definisci un'espressione regolare o una grammatica regolare che li genera:
- (1) l'insieme delle stringhe binarie che contengono le stringhe 101 o 010 (o ambedue) come sottostringhe.
 - (2) l'insieme delle stringhe binarie che contengono le stringhe 00 e 11 come sottostringhe.
 - (3) l'insieme delle stringhe binarie che contengono la stringa 00 ma non la stringa 11 come sottostringhe.
 - (4) l'insieme delle stringhe binarie che contengono iniziano con 00 e finiscono con 11.
 - (5) l'insieme delle stringhe sull'alfabeto {x,y,z} in cui ogni x è immediatamente seguita da una y.
 - (6) l'insieme delle stringhe sull'alfabeto {x,y,z} che contengono un numero dispari di y.

1)



$$\begin{aligned}
 S &\rightarrow OS \mid IS \mid IA \mid OB \\
 A &\rightarrow OL \\
 B &\rightarrow ID \\
 C &\rightarrow IE \\
 D &\rightarrow OE \\
 E &\rightarrow OE \mid IE \mid \varepsilon
 \end{aligned}$$

2)



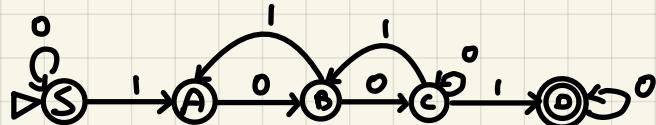
$$\begin{aligned}
 S &\rightarrow OS \mid IS \mid OA \mid IE & D &\rightarrow OD \mid ID \mid \varepsilon \\
 A &\rightarrow OB & E &\rightarrow IF \\
 B &\rightarrow OB \mid IC & F &\rightarrow IF \mid OG \\
 C &\rightarrow ID & G &\rightarrow OD
 \end{aligned}$$

ES:

0101010001000111

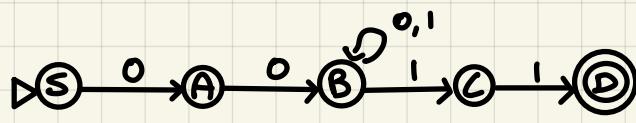
$$\begin{aligned}
 OS \rightarrow 0IS \rightarrow 01OS \rightarrow 010IS \rightarrow 0101IS \rightarrow \\
 01010IS \rightarrow 010101OA \rightarrow 01010100B \rightarrow \\
 \rightarrow 010101000B \rightarrow 0101010001C \rightarrow \\
 0101010001D \rightarrow 01010100010D \rightarrow \\
 010101000100D \dots
 \end{aligned}$$

3)



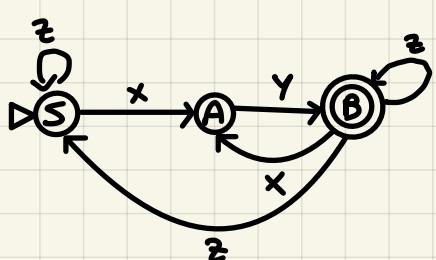
$$\begin{aligned}
 S &\rightarrow OS \mid IA & C &\rightarrow IB \mid OC \mid ID \\
 A &\rightarrow OB & D &\rightarrow OD \mid \varepsilon \\
 B &\rightarrow IA \mid OC
 \end{aligned}$$

4)



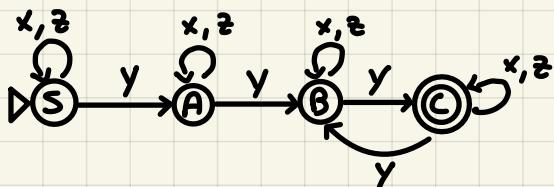
$$\begin{aligned}
 S &\rightarrow OA & C &\rightarrow ID \\
 A &\rightarrow OB & D &\rightarrow \varepsilon \\
 B &\rightarrow OB \mid IB \mid IC
 \end{aligned}$$

5)



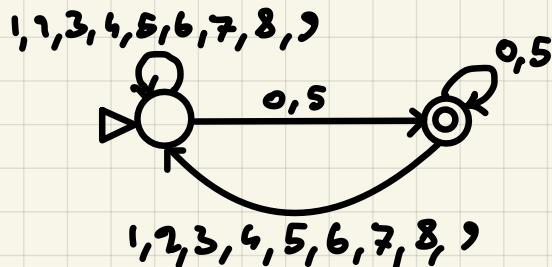
$$\begin{aligned}
 S &\rightarrow 2S \mid XA \\
 A &\rightarrow XA \mid YB \\
 B &\rightarrow XA \mid ZS \mid ZB \mid \varepsilon
 \end{aligned}$$

6)



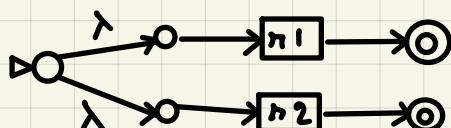
$$\begin{aligned}
 S &\rightarrow XS \mid ZS \mid YA \\
 A &\rightarrow XA \mid ZA \mid YB \\
 B &\rightarrow XB \mid ZB \mid YC \\
 C &\rightarrow XC \mid ZC \mid YB \mid \varepsilon
 \end{aligned}$$

Definire un ASF che, avendo in input stringhe costruite sull'alfabeto $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$, riconosce il linguaggio delle stringhe che descrivono un numero multiplo di 5 in base 10. (È ammesso che un numero inizi per 0, come ad esempio 0051 o 012345)



Sono dati M_1 e M_2 , due automi a stati finiti che riconoscono i linguaggi L_1 e L_2 rispettivamente;

(1) definire un automa M che riconosce il linguaggio $L_1 \cup L_2$ (formato dalle stringhe che appartengono a L_1 o a L_2 o a entrambi). Analogamente per intersezione

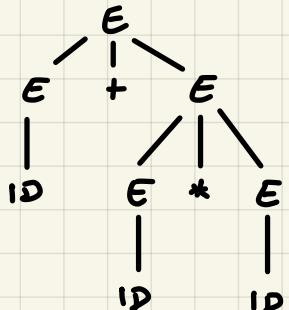


ES DISCUTEREMO IL PROBLEMA DELLA SEGUENTE GRAMMATICA:

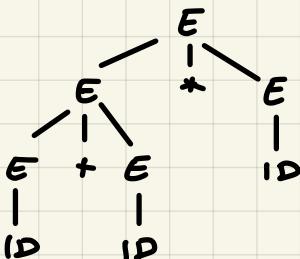
$$E \rightarrow \text{id} \mid E \text{ op } E \mid (E) \quad \text{op} \rightarrow + \mid - \mid * \mid /$$

SE MO COME INPUT: $\text{id} + \text{id} * \text{id}$ HO DUE CASI:

1)



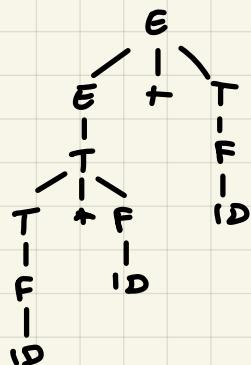
2)



NO UNA GRAMMATICA AMBIGUA \rightarrow 2 INTERPRETAZIONI DIVERSE

VERSIONE NON AMBIGUA \rightarrow $E \rightarrow E + T \mid E - T \mid T$
 $T \rightarrow T * F \mid T / F \mid F$
 $F \rightarrow \text{id}$

INPUT: $\text{id} + \text{id} + \text{id}$



NON PUÒ ESSERE DI TIPO 3 PERCHÉ
ASF NON CONTA
SOLO DA TIPO 2 IN GIÙ

Esercizio 3.5. Derivare un'espressione regolare che generi l'insieme di tutte le sequenze di 0 ed 1 che contengono un numero di 0 divisibile per 3. $1^*(1^*01^*01^*)^*$

Grammatiche libere dal contesto

Considera la seguente grammatica G (assioma S), con simboli terminali $\{a, b, c\}$:

$$S \rightarrow aAb \mid AE$$

$$A \rightarrow aAbE \mid ab$$

$$B \rightarrow AB \mid c$$

$$D \rightarrow AAcE$$

$$E \rightarrow BA$$

$$F \rightarrow FA \mid a$$

(1) Identifica i simboli non terminali che non sono raggiungibili a partire dall'assioma. **D, F**

(2) Trasforma la grammatica eliminando le produzioni inutili. ~~ELIMINA D, F~~

(3)

(Risposta (1): D,F - non esiste la possibilità di generare una stringa che contiene D o F a partire da S e ottenendo una sequenza di terminali;

(2) bisogna eliminare tutte le produzioni che coinvolgono D e F)

Considera la seguente grammatica G (assioma S), con simboli terminali $\{a, b, c\}$:

$$S \rightarrow aAbB \mid AC$$

$$A \rightarrow aAb \mid ab$$

$$B \rightarrow abB$$

$$C \rightarrow ABC \mid c$$

B NON FINISCE MAI, LA TORNARO

$$\begin{array}{l} \hookrightarrow S \rightarrow AC \\ \quad A \rightarrow aAb \mid ab \\ \quad C \rightarrow c \end{array}$$

(1) Identifica i simboli non terminali che non sono utilizzati per definire stringhe del linguaggio.
(2) Trasforma la grammatica eliminando le produzioni inutili

Trova una grammatica libera dal contesto per i seguenti linguaggi

$$L_1 = \{ 0^m 1^n \text{ con } n \neq m, n, m > 0 \}$$

$$L_2 = \{ a^m b^n c^p d^q \text{ con } m+n=p+q>0 \}$$

Sugg. Parti da grammatica per $\{ 0^n 1^n \text{ con } n > 0 \}$

L_1

$$\begin{array}{l} S \rightarrow 0A \mid 1B \\ A \rightarrow 0A \mid 10A \mid 1A \mid 1 \\ B \rightarrow 0B \mid 1B \mid 0B \mid \epsilon \end{array}$$

L_2

$$\begin{array}{l} S \rightarrow \alpha Sd \mid A \mid C \mid ad \\ A \rightarrow aAc \mid B \mid ac \\ B \rightarrow bBc \mid bc \\ C \rightarrow bCd \mid B \mid bd \end{array}$$

Ricorsione IMMEDIATA

$$A \rightarrow A b | \textcircled{C}$$



A'

$$A \rightarrow c A'$$

$$A' \rightarrow b A' | \epsilon$$

Ricorsione non immediata

$$A \rightarrow B s | a b | c d \quad B \rightarrow A c$$

$$B \rightarrow a b c | c d c$$

$$A \rightarrow c A'$$

$$A \rightarrow A b | c B$$

$$A' \rightarrow b A' | \epsilon$$

$$A \rightarrow c B A'$$

~ ~ ~ ~

First()

$$A \rightarrow \boxed{B | C | D}$$

$$\begin{aligned} \text{First}(A) : & \{ a, z, h, c, f, e \} \\ K & \end{aligned}$$

$B \rightarrow \boxed{a B} | \textcircled{c}$

$D \rightarrow c D | F | \lambda$

$C \rightarrow z C | h | \lambda$

Follow() :

Consideriamo la grammatica delle espressioni che qui riformuliamo dopo avere eliminato le ricorsioni sinistre.

$$\begin{array}{lllll} E \rightarrow TQ & Q \rightarrow +TQ & Q \rightarrow -TQ & Q \rightarrow \lambda & T \rightarrow FR \\ R \rightarrow *FR & R \rightarrow /FR & R \rightarrow \lambda & F \rightarrow (E) & F \rightarrow \text{id} \end{array}$$

In questo caso, è facile verificare che gli insiemi FIRST e FOLLOW sono i seguenti:

$$\text{FIRST}(E) = \text{FIRST}(F) = \text{FIRST}(T) = \text{FIRST}(TQ) = \text{FIRST}(FR) = \{ ., \text{id} \}$$

$$\text{FIRST}(Q) = \{ +, -, \epsilon \} \quad \text{FIRST}(R) = \{ *, /, \epsilon \} \quad \text{FIRST}(+TQ) = \{ + \}$$

$$\text{FIRST}(-TQ) = \{ - \} \quad \text{FIRST}(*RF) = \{ * \} \quad \text{FIRST}(/RF) = \{ / \}$$

$$\text{FOLLOW}(E) = \text{FOLLOW}(Q) = \{ ., \} \quad \text{FOLLOW}(F) = \{ +, -, *, /, ., \}$$

$$\text{FOLLOW}(T) = \text{FOLLOW}(R) = \{ ., -, ., \}$$

Esercizio 4.9. Progettare una tabella di parsing LL(1) per la grammatica

$$E \rightarrow -E \quad E \rightarrow (E) \quad E \rightarrow VT \quad T \rightarrow -E \quad T \rightarrow \lambda \quad V \rightarrow iU \quad U \rightarrow (E) \quad U \rightarrow \lambda$$

Tracciare quindi la sequenza di parsing con input $i-i((i))$ e con input $i-(())i$.

$$\text{FIRST}(E) = \{-, (, i\} \quad \text{FIRST}(T) = \{-, \epsilon\} \quad \text{FIRST}(V) = \{i\} \quad \text{FIRST}(U) = \{(, \epsilon\}$$

$$\text{FOLLOW}(E) = \{\$,)\} \quad \text{FOLLOW}(T) = \{\$,)\} \quad \text{FOLLOW}(V) = \{-, \$,)\} \quad \text{FOLLOW}(U) = \{-, \$,)\}$$

	i	$-$	$($	$)$	$\$$
E	VT	$-E$	(E)		
T		$-E$		E	E
V	iU				
U	E	(E)	E	E	

PILA	INPUT	PRODUZIONE	DERIVAZIONE
$\$E$	$i-i((i))$	$E \rightarrow VT$	$E \rightarrow VT$
$\$TV$	$i-i((i))$	$V \rightarrow iU$	$\rightarrow iU$
$\$TU_i$	$i-i((i))$	$U \rightarrow \lambda$	$\rightarrow T$
$\$TU$	$-i((i))$	$T \rightarrow -E$	$\rightarrow -E$
$\$E-$	$-i((i))$	$\rightarrow E$	$\rightarrow E$
$\$E$	$i((i))$	$E \rightarrow VT$	$\rightarrow VT$
$\$TV$	$i((i))$	$V \rightarrow iU$	$\rightarrow iU$
$\$TU_i$	$i((i))$	$U \rightarrow \lambda$	$\rightarrow U$
$\$TU$	$((i))$	$U \rightarrow (E)$	$\rightarrow (E)T$
$\$T)E($	$((i))$		$\rightarrow E)T$
$\$T)E$	$(i))$		$\rightarrow (E))T$
$\$T))E($	$(i))$	$E \rightarrow (E)$	$\rightarrow E))T$
$\$T))E$	$i))$		$\rightarrow VT))T$
$\$T))TV$	$i))$	$E \rightarrow VT$	$\rightarrow iU T))T$
$\$T))TU_i$	$i))$	$V \rightarrow iU$	$\rightarrow VT))T$
$\$T))TU$	$i))$	$U \rightarrow \lambda$	$\rightarrow T))T$
$\$T))T$	$i))$	$T \rightarrow -E$	$\rightarrow))T$
$\$T))$	$i))$		$\rightarrow T$
$\$T)$	$)$		$\rightarrow \$$
$\$T$	$\$$	$T \rightarrow \$$	
$\$$	$\$$		

	i	$-$	$($	$)$	$\$$
E	VT	$-E$	(E)		
T		$-E$		E	E
V	iU				
U	E	(E)	E	E	

PILA	INPUT	PRODUZIONE	DERIVAZIONE
$\$E$	$i-((i))$	$E \rightarrow VT$	$\rightarrow TV$
$\$TV$	$i-((i))$	$V \rightarrow iU$	$\rightarrow TU$
$\$TU_i$	$i-((i))$	$U \rightarrow \lambda$	$\rightarrow T$
$\$TU$	$-((i))$	$T \rightarrow -E$	$\rightarrow E -$
$\$E -$	$-((i))$		
$\$E$	$((i))$	$E \rightarrow (E)$	$\rightarrow)E($
$\$)E($	$((i))$		
$\$)E$	$(i))$	$E \rightarrow (E)$	$\rightarrow))E($
$\$))E($	$(i))$		
$\$))E$	$i))$		

NON SI PUÒ

C a a b b c

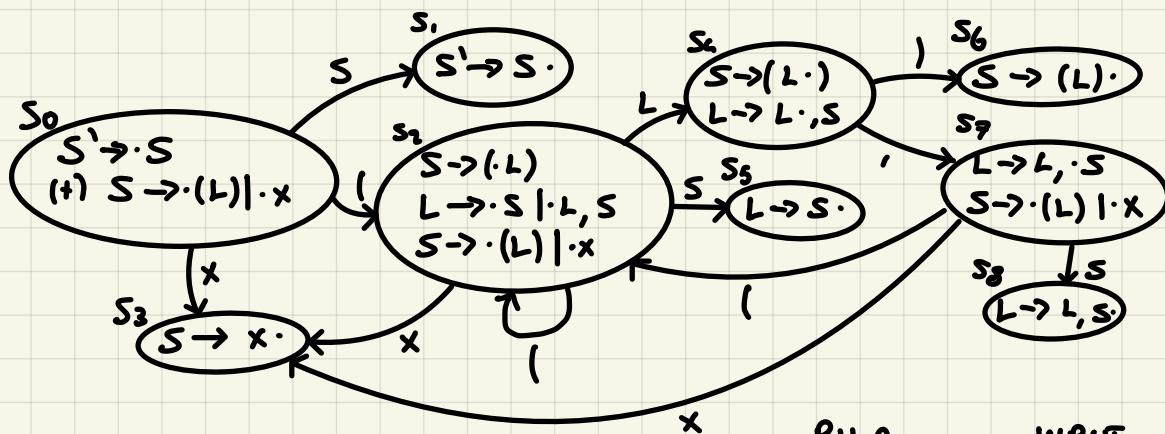
C a c b c a c

$S \rightarrow S_a S \rightarrow C_a S \rightarrow C_a S_a S S \rightarrow C_a S_a S_a S S$
C a

$S \rightarrow S_a S \rightarrow C_a S \rightarrow C_a S_b S \rightarrow C_a C_b S_a S$
L $\rightarrow C_a C_b C_a C$

$S \rightarrow S_b S \rightarrow S_a S_b S \rightarrow C_a C_b S_a S \rightarrow C_a C_b C_a C$

0: $S' \rightarrow S\$$	1: $S \rightarrow (L)$
2: $S \rightarrow x$	3: $L \rightarrow S$
4: $L \rightarrow L, S$	



	()	x	,	\$	S	L
0	S2		S3		Goto 1		
1			ACCETTA				
2	S2		S3		Goto 5	Goto 6	
3			REDUCE $S \rightarrow x$				
4			S6	S7	1		
5			REDUCE $L \rightarrow S$				
6			REDUCE $S \rightarrow (L)$				
7	S2		S3		Goto 8		
8			REDUCE $L \rightarrow L, S$				

TABELLA ACTION / GOTO

PILA	INPUT	ACTION
S0	(x, (x))\$	
S0 S2	x, (x))\$	
S0 S2 S3	, (x))\$	REDUCE $S \rightarrow x$
S0 S2 S5	, (x))\$	REDUCE $L \rightarrow S$
S0 S2 S4	(x))\$	S2
S0 S2 S4 S7	(x))\$	S2
S0 S2 S4 S7 S2	x))\$	S3
S0 S2 S4 S7 S2 S3))\$	REDUCE $S \rightarrow x$
S0 S2 S4 S7 S2 S5))\$	REDUCE $L \rightarrow S$
S0 S2 S4 S7 S2 S4))\$	S6
S0 S2 S4 S7 S2 S4 S6))\$	REDUCE $S \rightarrow (L)$
S0 S2 S4 S7 S2 S4 S8))\$	REDUCE $L \rightarrow L, S$
S0 S2 S4 S6	\$	REDUCE $S \rightarrow (L)$
S0 S1	\$	ACCETTA

ES GRAMMATICA

DATA $S \rightarrow () | SS | (S)$ VEDERE LE PRODUZIONI PER OTTENERE $()((())())$

$S \rightarrow SS \rightarrow S(S) \rightarrow S((S)) \rightarrow S(((S))) \rightarrow S(((()))) \rightarrow ()((())())$

ES GRAMMATICA

DATA: $E \rightarrow E + T | E - T | T$ QUINDI CON UNA UNICA VARIABILE DELL'ESPRESSONE
 $T \rightarrow T * F | T / F | F$
 $F \rightarrow a | (E)$

SCRIVERE UNA GRAMMATICA CON IN PIÙ: RADICE, POTENZA, CAMBIO SEGNO

$E \rightarrow E + T | E - T | T$
 $T \rightarrow T * A | T / A | A$
 $F \rightarrow a | (E)$

$A \rightarrow -B | B$
 $B \rightarrow \sqrt{F} | F^a | F$

IN QUESTO MODO LE PRECEDENZE VENGONO RISPETTATE

ES GRAMMATICA

SCRIVERE LA GRAMMATICA DI $L = \{a^n b^n \mid n \geq 1\}$ $\rightarrow S \rightarrow aSb \mid ab$

ES GRAMMATICA

SCRIVERE LA GRAMMATICA DI $L = \{a^n b \mid n \geq 0 \text{ E PARI}\}$ $\rightarrow S \rightarrow b \mid aaS$

ES GRAMMATICA

SCRIVERE LA GRAMMATICA DI $L = \{a^n b^n c^n \mid n \geq 1\}$

$S \rightarrow abc \mid abcABC$	$cA \rightarrow Ac$
$aA \rightarrow aa$	$bA \rightarrow AB$
$bB \rightarrow bb$	$cB \rightarrow Bc$
$cC \rightarrow cc$	$ABC \rightarrow ABCABC \mid \epsilon$

ES GRAMMATICA

SCRIVERE LA GRAMMATICA DI $L = \{a^n b^m c^{n+m} \mid n \geq 1, m \geq 0\}$

$S \rightarrow aBc \mid aSc$
 $B \rightarrow bBc \mid bc \mid \epsilon$

ES GRAMMATICA

SCRIVERE LA GRAMMATICA DI $L = \{a^{2^n} \mid n \geq 0\}$ COPPIA AA RIPETUTA n VOLTE

$S \rightarrow a | aa | NaaB$
 $N \rightarrow NN \mid A$
 $Aaa \rightarrow aaaaA$
 $AaaB \rightarrow aaaaB | aaaa$

- A. Per $n > 1$, sia $T(n)$ il numero esatto di esecuzioni del comando $a \leftarrow a + 2$ nel seguente frammento codice:

```
for i  $\leftarrow 1$  to  $n - 1$  do
    for j  $\leftarrow n - i + 1$  to  $n$  do
        a  $\leftarrow a + 2$ 
```

Esprimere $T(n)$ utilizzando la notazione $O(n)$.

Calcolare il valore $T(10)$ oppure un valore maggiore di $T(10)$ che sia il più possibile vicino al valore esatto.

$$T(n) = \sum_{i=1}^{n-1} \sum_{j=n-i+1}^n 1 = \sum_{i=1}^{n-1} i - 1 = \sum_{i=1}^{n-1} i - \sum_{i=1}^{n-1} 1 = \frac{(n-1) \cdot n}{2} - (n-1) = O(n^2)$$

$$T(10) = \frac{90}{2} - 9 = 36$$

- B. Definire le classi: P, NP e indicare le relazioni di contenimento fra le classi.

P È L'INSIEME DEI LINGUAGGI DECIDIBILI DA UNA MDT DETERMINISTICA
OPPURE È L'INSIEME DEI PROGRAMMI RISOLUBILI DA UNA MDT DETERM.
IN TEMPO POLINOMIALE

NP: IDENTICO A SU MA CON MDT NON DETER.

P ⊂ NP MA NON SAPPIAMO SE P=NP

- C. Dati due problemi P e Q, come posso dimostrare che P è non più difficile di Q?

P NON È PIÙ DIFFICILE DI Q SE ESISTE UNA RIDUZIONE CHE CI PERMETTE DI PASSARE DA P A Q

- D. Con riferimento al problema SAT, in cui, data una formula booleana in CNF, occorre stabilire se la formula è soddisfacibile o meno, si mostri che il problema appartiene alla classe NP.

Come si modifica la dimostrazione nel caso di 4-SAT, in cui ciascuna clausola della formula ha esattamente 4 letterali?

IL PROBLEMA SAT È NP PERCHÉ NON SIANO ANCORA IN GRADO DI TROVARE UN ALGORITMO CHE LO RISOLVI IN TEMPO POLINOMIALE.
PERÒ ESISTONO PROGRAMMI CHE SONO IN GRADO DI VERIFICARE LA SODDISFAZIBILITÀ

- E. Si descriva il funzionamento di una macchina di Turing a due nastri. Sotto quali aspetti tale macchina è "superiore" a quella a nastro singolo?

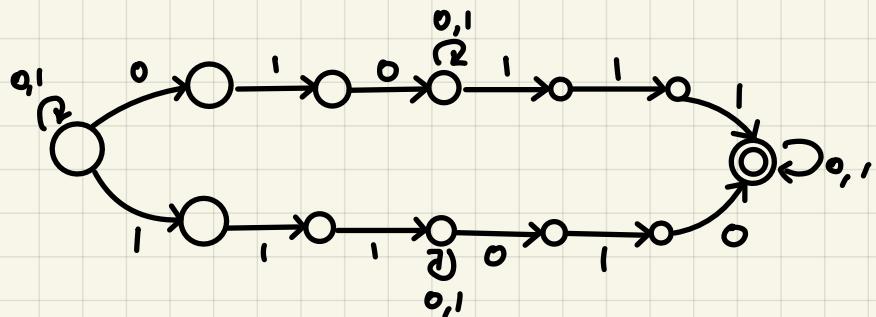
UNA MDT CON UN NASTRO MA LA STESSA COMPUTABILITÀ DI UNA MDT A DUE NASTRI, MA QUELLA A DUE HA 2 TESTINE ED È PIÙ VELOCE

- F. Illustrare la gerarchia di Chomsky delle grammatiche; indicare per ciascuna categoria la proprietà o caratteristica a vostro giudizio maggiormente rilevante.

ABBIANO 4 TIPOLOGIE DI GRAMMATICHE GENERATIVE:

- **TIPO 3** o REGOLARE: OGNI PRODUZIONE È LINEARE A DESTRA
 $A \rightarrow \alpha B, A \rightarrow \alpha$
- **TIPO 2** o LIBERE DA CONTESTO: OGNI PRODUZIONE DEVE ESSERE DEL TIPO $A \rightarrow \alpha$ CON α UNA STRINGA DI TERMINALI E NON
- **TIPO 1** o CONTESTUALI: OGNI PRODUZIONE DEVE ESSERE DEL TIPO $\alpha \rightarrow \beta$ CON $\beta \geq 1$
- **TIPO 0** o NON LIMITATE: NON ABBIANO NESSUN VINCOLO SULLE PRODUZIONI

- G. Descrivere un automa a stati finiti (deterministico o nondeterministico) che riconosce le stringhe appartenenti al linguaggio costituito da tutte e sole le stringhe binarie contenenti la sequenza 010 e la sequenza 111 (in qualunque posizione, cioè possiamo avere prima la stringa 010 e poi la stringa 111, o viceversa).



- H. Descrivere una grammatica libera da contesto che generi il seguente linguaggio

$$L = \{a^n b^m \mid m \geq 0, n \geq 0, m > n\}$$

(stringhe di a, b che iniziano con delle a , seguite da un numero di b strettamente maggiore del numero delle a).

$$\begin{aligned} S &\rightarrow aSb \mid aAb \mid Ab \\ A &\rightarrow Ab \mid b \mid \epsilon \end{aligned}$$

I. Si consideri la seguente grammatica G (con assioma S e simboli terminali {a, b}):

$$S \rightarrow aAb \mid aSb$$

$$A \rightarrow aaAbb \mid ab$$

(1) Fornire la sequenza di derivazioni che a partire dall'assioma deriva la stringa aaaabbbb

(2) Descrivere il linguaggio generato dalla grammatica

(3) Se ci sono produzioni inutili, semplificare la grammatica eliminandole; motivare la risposta.

a) $\text{aaa} \alpha \text{bb} \beta \rightarrow S \rightarrow aSb \rightarrow aaSbb \rightarrow \text{aaa} \alpha A \beta \rightarrow \text{aaa} \alpha ab \beta$

b) $L = \{a^n b^n, n \geq 2\}$

c) $A \rightarrow \alpha A \beta$ è INUTILE per CHÈ SI OTTIENE LA STESSA DERIVANDO UNA VOLTA $S \rightarrow aSb$ E POI $S \rightarrow aAb$

ESAME 13/06/19 B

A. Per $n > 0$, si consideri una ricorrenza $T(n)$ il cui albero delle chiamate è binario e completo (cioè, ogni nodo interno ha due figli e ciascun ramo ha la stessa lunghezza) con n foglie. Il lavoro compiuto in ciascun nodo dell'albero è pari a 1.

Esprimere $T(n)$ utilizzando la notazione $O(n)$. Si determini il valore esatto di $T(8)$ per un albero con 8 foglie.

$$2^{n-1}$$

$$2(\# \text{FORME}) - 1$$

$$T(8) = 15$$

B. Definire la classe dei problemi NP-completi e indicare le relazioni di contenimento rispetto alle classi P e NP

UN LINGUAGGIO L È NP-COMPLETO SE $L \in NP$ E SE OGNI ALTRO LINGUAGGIO È POLINOMIALMENTE RIDUCIBILE A L
UN LINGUAGGIO NP-COMPLETO È TRA I PIÙ DIFFICILI DELLA CLASSE NP, NEL SENSO CHE SE APPARTENESSE ALLA CLASSE P, ALLORA L'INTERA CLASSE NP SAREBBE INCLUSA IN P

C. Dati due problemi P e Q, come posso dimostrare che P è almeno tanto difficile quanto Q?

SE Q È NP-COMPLETO E P È IN NP, ALLORA P POTREBBE ESSERE ALMENO TANTO DIFFICILE QUANTO Q

- D. Dimostrare che il problema Vertex cover che, dato un grafo $G=(V,E)$ e un intero k decide se esiste un insieme di al più k nodi che copre tutti gli archi del grafo, appartiene alla classe NP.
 Come si modifica la dimostrazione nel caso di un grafo $G=(V,E)$ con n nodi e in cui si chiede un copertura di taglia $\lceil n/10 \rceil$?

PER DIMOSTRARE CHE VERTEX COVER È IN NP DOBBIAMO FORNIRE UN CERTIFICATO VERIFICABILE IN TEMPO POLINOMIALE.
UN POSSIBILE CERTIFICATO POTREBBE ESSERE UN SOTTOINSIEME DI VERTICI V'
 1 **VERIFICO CHE V' CONTENGA AL MASSIMO K VERTICI**
 2 **V ARCO (u,v) VERIFICO CHE ALMENO UNO DEI VERTICI u o v SIA PRESENTE IN V'**
 SE TUTTO QÒ È SODDISFATTO ALETTO IL CERTIFICATO COME SOLUZIONE VALIDA
 LA VERIFICA PUÒ ESSERE EFFETTUATA IN TEMPO POLINOMIALE, QUINDI È NP

- E. Si descriva il funzionamento di una macchina di Turing nondeterministica. Sotto quali aspetti tale macchina è "superiore" a quella deterministica?

IL SUO FUNZIONAMENTO È SIMILE A QUELLA DETERMINISTICA MA:
 1 **UNA TRANSIZIONE DI STATO PUÒ ESSERE DEFINITA IN MODO NON DETERMINISTICO, QUINDI PIÙ TRANSIZIONI POSSONO ESSERE ABILITATE CONTEMPORANEAMENTE**
 2 **UNA MDT ND ACCETTA UN INPUT SE ALMENO UN CAMMINO RAGGIUNGE UNO STATO FINALE DI ACCETTAZIONE**

UNA MDT N.D. È SUPERIORE PUÒ EFFETTUARE PIÙ TRANSIZIONI INSIEME MA LA STESSA POTENZA COMPUTAZIONALE DI UNA DETERMINISTICA

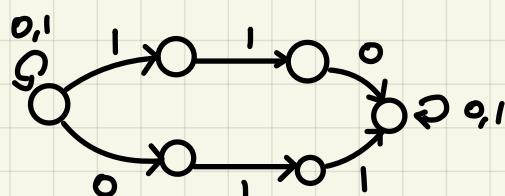
- F. Definire il concetto di grammatica ambigua fornendo e discutendo un esempio. Discutere brevemente la necessità di avere grammatiche non ambigue per definire linguaggi di programmazione.

È UNA GRAMMATICA CHE GENERA ALMENO UNA STRINGA ATTAVERSO PIÙ DERIVAZIONI. QUESTO PUÒ PORTARE AD UN'INTERPRETAZIONE SBAGLIATA. PER ESEMPIO:

$$\begin{array}{ll} E \rightarrow E+E & E \rightarrow E-E \\ E \rightarrow E/E & E \rightarrow (E) \end{array} \quad \begin{array}{ll} E \rightarrow E^*E & E \rightarrow E+E \\ E \rightarrow (E) & E \rightarrow ID \end{array}$$

CON INPUT $ID + ID + ID$ POSSO USARE $E \rightarrow E^*E$, $E \rightarrow E+E$, $E \rightarrow ID$
 OPPURE $E \rightarrow E+E$, $E \rightarrow E^*E$, $E \rightarrow ID$

- G. Descrivere un automa a stati finiti (deterministico o nondeterministico) che riconosce le stringhe appartenenti al linguaggio costituito da tutte e sole le stringhe binarie contenenti un almeno due simboli 1 e un simbolo 0 (in qualunque posizione, cioè possiamo avere prima la stringa 11 e poi uno 0 o viceversa).



H. Descrivere una grammatica libera da contesto che generi il seguente linguaggio

$$L = \{0^n 1^m \mid m \geq 0, n \geq 0, m < n\}$$

(stringhe di 0 e 1 che iniziano con 0 seguite da un numero di 1 strettamente minore del numero di 0).

$$\begin{array}{l} S \rightarrow 0S1 \mid 0A1 \mid A0 \\ A \rightarrow A0 \mid 0 \mid \epsilon \end{array}$$

I. Si consideri la seguente grammatica G (con assioma S e simboli terminali {0, 1}):

$$S \rightarrow 11A00 \mid 11S00$$

$$A \rightarrow 11A00 \mid 10$$

(1) Fornire la sequenza di derivazioni che a partire dall'assioma deriva la stringa 1111100000

(2) Descrivere il linguaggio generato dalla grammatica

(3) Se ci sono produzioni inutili semplificare la grammatica eliminandole; motivare la risposta.

a) 11111 00000

$$S \rightarrow 11S00 \rightarrow 1111A00000 \rightarrow 11111 00000$$

b) L $\{1^n 0^n, n \geq 3 \text{ E DISPARI}\}$

c) S $\rightarrow 11S00$ È INUTILE PERCHÉ IDENTICA A $S \rightarrow 11A00$

ES GRAMMATICA

OTTERNERE L'ESPRESSONE REGOLARE DI

$$\begin{array}{l} A \rightarrow bA \mid aB \mid b \\ B \rightarrow aA \mid bB \mid a \end{array}$$

$$\begin{cases} A = bA + aB + b \\ B = aA + bB + a \end{cases}$$

$$\begin{aligned} B &= aA + bB + a = bB + aA + a = b(aA + bB + a) + aA + a = \\ &= b^2a + bbB + ba + aA + a = b^2(aA + a) \end{aligned}$$

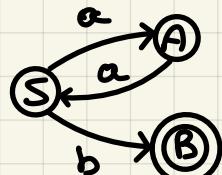
$$\begin{aligned} A &= bA + ab^*(aA + a) + b = bA + ab^*aA + ab^*a + b = (b + ab^*a)A + ab^*a + b \\ &= (b + ab^*a)^+ \end{aligned}$$

ES GRAMMATICA

PASSA DA E.R. AD AUTOMA A GRAMMATICA AD E.R.

1) $L = \{a^n b \mid n \geq 0 \text{ E PARI}\}$

E.R: $(aa)^*b$



AUTOMA.

GRAMMATICA: $\begin{array}{l} S \rightarrow aA \mid b \\ A \rightarrow aS \end{array}$

E.R.

$$\begin{cases} S = aA + b \\ A = aS \end{cases}$$

$$\begin{aligned} S &= aAS + b = aa(aAS + b) + b \\ &= (aa)^+b + b = (aa)^*b \end{aligned}$$

3) $S \rightarrow aS \mid bA \mid \epsilon$ ELIMINO ϵ DA A PER AVERE UNA GRAMMATICA "RILASSATA"

$$A \rightarrow aA \mid bS \mid a$$

$S \rightarrow aS \mid bA \mid \epsilon$

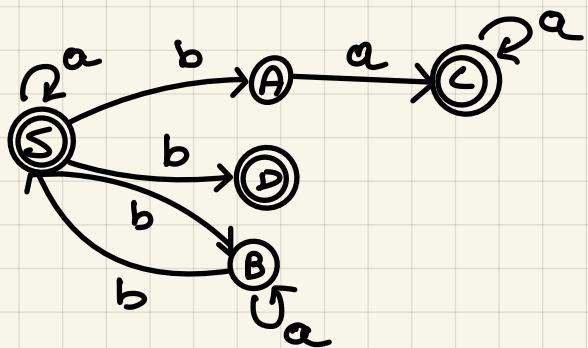
$$A \rightarrow aA \mid bS \mid a$$

ER: $\begin{cases} S = aS + bA + \epsilon \\ A = aA + bS + a \end{cases}$

$$\begin{aligned} A &= aA + bS + a = a(aA + bS + a) + bS + a = a^2A + abS + a^2 + bS + a \\ &= a^2A + (ab + b)S + a^2 + a = a^2 + a^*bS = a^*(bS + a) \end{aligned}$$

$$\begin{aligned} S &= aS + bA + \epsilon = aS + b(a^*(bS + a)) + b + \epsilon = aS + ba^*(bS + a) + b + \epsilon \\ &= aS + ba^*bS + ba^*b + b + \epsilon = (a + ba^*b)S + ba^*b + b + \epsilon \\ &= (a + ba^*b)[(a + ba^*b)S + ba^*b + b + \epsilon] + ba^*b + b + \epsilon \\ &= (a + ba^*b)(a + ba^*b)S + (a + ba^*b)ba^*b + (a + ba^*b)b + (a + ba^*b) + ba^*b + b + \epsilon \\ &= (a + ba^*b)^* (ba^*b + b + \epsilon) = (a + ba^*b)^* (ba^*b) + (a + ba^*b)^* \end{aligned}$$

AUTOMA:

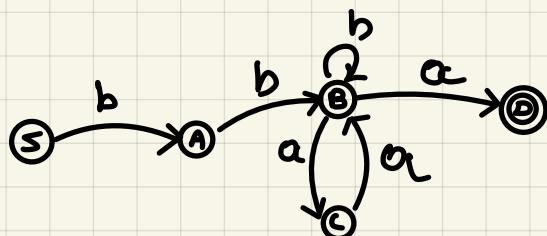


GRAMMATICA:

$$\begin{aligned} S' &\rightarrow S \mid \epsilon \\ S &\rightarrow aS \mid bB \mid bD \mid b \mid a \\ A &\rightarrow aC \mid a \\ B &\rightarrow aB \mid bS \\ C &\rightarrow a \mid aC \end{aligned}$$

ESAME 17/01/2020

- Con riferimento al linguaggio R descritto dall'espressione regolare $bb((aa)^* + b)^*a$
 - definire un ASF (deterministico o non) che riconosce il linguaggio R
 - definire una grammatica regolare che genera R



$$\begin{aligned} S &\rightarrow bA \\ A &\rightarrow bB \\ B &\rightarrow bB \mid ac \mid a \\ C &\rightarrow aB \end{aligned}$$

2. Si consideri la grammatica $G: S \rightarrow T\$, T \rightarrow (T)T \mid \epsilon$, con terminali $\{\$\text{, } (\text{, })\}$, non terminali $\{S, T\}$ e assioma S , che genera il linguaggio L
- 2.1. Stabilire se G è $LL(1)$ oppure no; nel caso non lo sia, modificare G in una grammatica G' che sia $LL(1)$ ed equivalente a G
 - 2.2. Descrivere un algoritmo che effettua il parsing predittivo delle stringhe di L

$$\begin{array}{l} S \rightarrow T\$ \\ T \rightarrow (T)T \mid \epsilon \end{array}$$

$$FIRST(S) = FIRST(T) = \{(, \epsilon\}$$

$$FOLLOW(S) = \{ \$ \} \quad FOLLOW(T) = \{ \$,) \}$$

	()	\$
S	T\$		T\$
T	(T)	ϵ	ϵ

Ogni cella della tabella ha al più una produzione quindi la grammatica è $LL(1)$

3. Descrivere la tecnica di progettazione algoritmica denominata "programmazione dinamica", chiarendo in cosa si distingue dal divide et impera, e illustrare un algoritmo (a piacere) basato sulla programmazione dinamica.

IL DIVIDE ET IMPERA SUDDIVIDE IL PROBLEMA IN TANTI SOTTO PROBLEMI, LI RISOLVE ED UNISCE LE SOLUZIONI.
LA PROGRAMMAZIONE DINAMICA FA LO STESSO VA NON UNISCE LE SOLUZIONI, OGNI SOLUZIONE DI UN SOTTO PROBLEMA AIUTA LA RISOLUZIONE DI UN ALTRO

5. Con riferimento alle classi P ed NP:

- 5.1. Definire la classe P
- 5.2. Definire la classe NP attraverso macchine di Turing non deterministiche
- 5.3. Definire la classe NP senza impiegare il concetto di non determinismo
- 5.4. Confrontare le due definizioni di NP, giustificandone l'equivalenza
- 5.5. Se per un problema della classe NP viene individuato un lower bound polinomiale, cosa possiamo dedurre in merito alla questione P vs NP?
- 5.6. Se per un problema della classe NP viene individuato un lower bound esponenziale, cosa possiamo dedurre in merito alla questione P vs NP?

- 1) LA CLASSE P È L'INSIEME DI TUTTI I LINGUAGGI L PER WI È UNA MDT DETERM. CHE CON INPUT x IN TEMPO POLINOMIALE DEGLI SE $x \in O$ & A L
- 2) LA CLASSE NP È L'INSIEME DI TUTTI I LINGUAGGI L PER WI È UNA MDT NON DETERM. CHE CON INPUT x IN TEMPO POLINOMIALE DEGLI SE $x \in O$ & A L
- 3) LA CLASSE NP È L'INSIEME DI TUTTI I LINGUAGGI L PER WI, SET $x \in L$, ALLORA È UN CERTIFICATO $C(x)$ TALE CHE:
 - $C(x)$ HA LUNGHEZZA POLINOMIALE IN $|x|$
 - È MDT DETERM. CHE CON INPUT x E $C(x)$ VERIFICA CHE $x \in L$ IN TEMPO POLINOMIALE IN $|x|$ E $|C(x)|$
- 5) POTREMO CONCLUDERE CHE $P = NP$ SOLO SE FOSSE NP-COMPLETO, QUINDI NULLA
- 6) POSSIAMO DEDURRE CHE $P \neq NP$

2. Definire le classi P e NP e indicare le relazioni di contenimento fra le classi

LA CLASSE P È L'INSIEME DI TUTTI I LINGUAGGI L PER WI È UNA MDT DETERM.
CHE CON INPUT X IN TEMPO POLINOMIALE DEGDE SE X E O & A L

LA CLASSE NP È L'INSIEME DI TUTTI I LINGUAGGI L PER WI È UNA MDT NON DETERM.
CHE CON INPUT X IN TEMPO POLINOMIALE DEGDE SE X E O & A L

P ⊂ NP MA NON SAPPIANO SE P=NP O P≠NP

3. Dimostrare che se un problema A è riducibile in tempo polinomiale ad un problema B e che B è riducibile in tempo polinomiale a C, allora A è riducibile a C.

VOCABOLARIO DIMOSTRARE CHE SE $A \leq_p B$ E $B \leq_p C$ ALLORA $A \leq_p C$

SE $A \leq_p B$ ALLORA È f_{AB} FUNZIONE DI RIDUZIONE TALE CHE SE A
MA ISTANZA SI(No), f_{AB} NI DA UNA ISTANZA SI(No) DI B

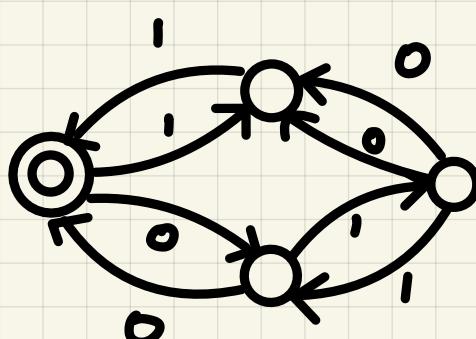
SE $B \leq_p C$ È f_{BC} CON LE STESSE CARATTERISTICHE

$$f_{AC} = f_{AB} \cdot f_{BC}$$

4. Nell'analisi di complessità e, in particolare nella notazione $O()$, si ignorano le costanti moltiplicative e additive. Discutere questa scelta indicandone i vantaggi e gli svantaggi.

SEMPLIFICHIAO L'ANALISI SULLA COMPLESSITÀ ED ELIMINIAMO LIMITI
COME IL LINGUAGGIO DI PROGRAMMAZIONE, LA CPU, I DATI ECC...
LO SVANTAGGIO È CHE CON COSTANTI MOLTO GRANDI, SE ELIMINATE
POTREMMO PERDERE MOLTE INFO

7. Descrivere un automa a stati finiti che riconosce il linguaggio formato dalle stringhe binarie di 0 e 1 costituito da tutte e sole le stringhe binarie contenenti un numero pari di 0 e un numero pari di 1 (in qualunque posizione; quindi l'automa accetta le stringhe 0011, 01010101, 00100100 e rifiuta le stringhe 0111 e 00100).

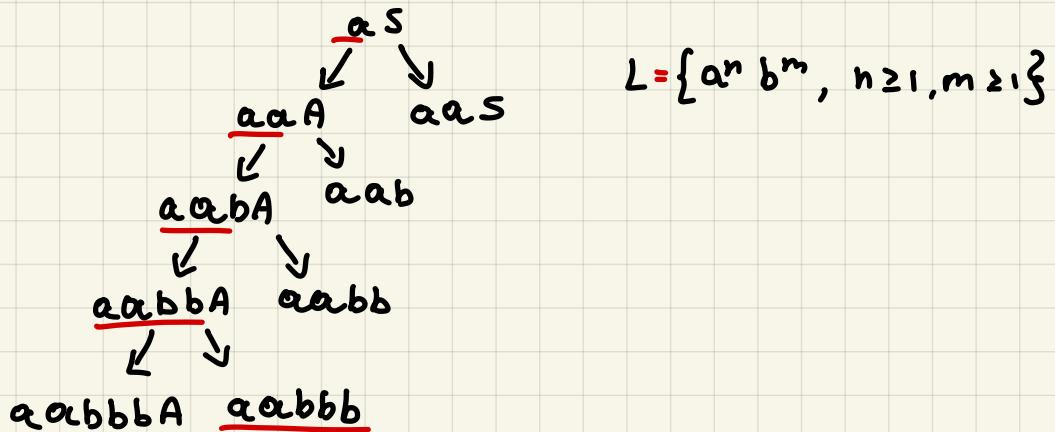


8. Sia data la grammatica $G = (N, T, P, S)$ con insieme dei simboli non terminali $N = \{S, A\}$, insieme simboli terminali $T = \{a, b\}$, assioma S e produzioni specificate nel seguito:

$$\begin{array}{ll} S \rightarrow aS & (1) \\ S \rightarrow aA & (2) \\ A \rightarrow bA & (3) \\ A \rightarrow b & (4) \end{array}$$

Fornire un albero di derivazione della stringa $aabbba$; specificare il linguaggio generato dalla grammatica e discutere se la grammatica è ambigua o no.

$$S \rightarrow aS \rightarrow aaA \rightarrow aabA \rightarrow aabbA \rightarrow aabbba$$



9. Fornire una grammatica non ambigua per generare il linguaggio delle espressioni aritmetiche avente cinque simboli terminali: id (che rappresenta un identificatore), i simboli di operazione + e - e le parentesi tonde (e).

Ovviamente la grammatica deve generare tutte e sole le stringhe che sono corrette (ad esempio non deve generare la stringa $id+((id-id)$ o la stringa $id++id-id$).

Motivare inoltre perché questo linguaggio non può essere generato da una grammatica di tipo 3.

$$\begin{array}{l} S \rightarrow E + E \mid E - E \\ E \rightarrow (S) \mid ID \mid S \end{array}$$

1. Definire una grammatica che genera il linguaggio $L = \{2^n 3^{n+1}, n > 0\}$ e spiegare perché non esiste un ASF che riconosce L .

$S \rightarrow 2S3S \mid 3$

NON È UNA M^TP PERCHÈ NON PUÒ CONTARE (FINO A n),
SEMMO A SARÈBBE UNO STATO A CASO (INFINITO)

2. Definire un ASF che, avendo in input stringhe costruite sull'alfabeto $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$, riconosce il linguaggio delle stringhe che descrivono un numero multiplo di 5 in base 10. (È ammissibile che un numero inizi per 0, come ad esempio 0051 o 012345)

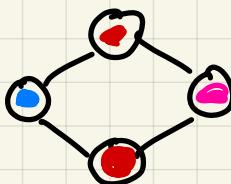
S → cas1cb51c

02 - 2023

L = { $a^{n+1} b^{n-1}$, $n \geq 1$ }

S → aSb|aa

$\varphi_{\text{col}}(G) = \begin{cases} \text{TRUE} & \text{SE IL GRANFO G PUÒ AVERE I NODI COLORATI} \\ & \text{DI 4 COLORI SENZA CHE DUE NODI ADJACENTI} \\ & \text{ABBIANO LO STESSO COLORE} \\ \text{FALSE} & \text{ALTRIMENTI} \end{cases}$



1. Analisi algoritmi

- 1.1. In un albero delle ricorrenze ogni nodo ha 0 o 2 figli. Descrivere le relazioni fra n (numero nodi), f (numero foglie), i (numero nodi interni) e h (altezza¹). [1]

L'ALBERO AVÀÀ AL PIÙ:

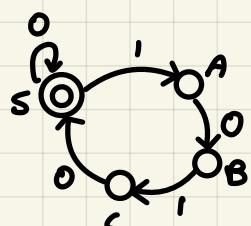
2^{h-1} FOGNE / 2^{h-1} NODI TOTALI / $h = \log_2 n + 1$ / $2^{h-1} - 1$ NODI INTERNI

ESEMPI COMPITI D'ESAME

COMPITO 1

- 1) Con riferimento al linguaggio $L = \{x \in (0+1)^* \mid 1 \text{ è sempre seguito da } 010\}$
- scrivere un opportuno automa che riconosce tutte e sole le stringhe che appartengono al linguaggio
 - scrivere una grammatica del tipo più alto possibile che genera il linguaggio L
 - determinare una espressione regolare che lo rappresenta.

a)



b)

$$\begin{array}{ll} S \rightarrow 0S11A1\epsilon & B \rightarrow 1C \\ A \rightarrow 0B & C \rightarrow 0S \end{array}$$

c) $[0^* (1010)^*]^*$

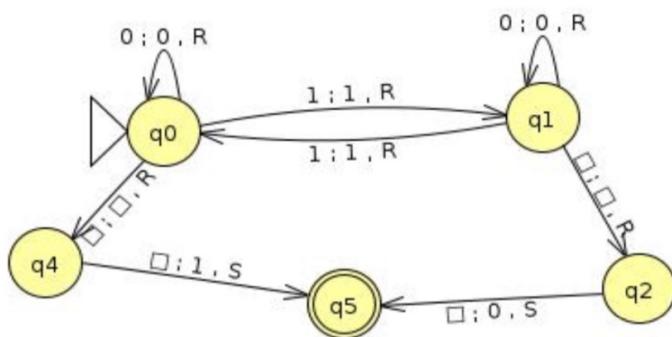
- 2) Mostrare la non chiusura dei linguaggi di tipo 2 rispetto l'intersezione.

- 3) Definire la notazione Ω . Motivare la seguente affermazione:
"Il problema P ha complessità $\Omega(n \log n)$ "

Ω ESPRIME UNA DELIMITAZIONE INFERIORE. È USATA NEL CALcolo DEL COSTO DI UN PROGRAMMA ED INDICA CHE NON ESISTONO ALGORITMI PER UN PROBLEMA CHE LO RISOLVONO IN UN COSTO MINORE DI $\Omega(n)$.

4) Dare la definizione di Macchina di Turing (MdT).

- a) Assumi che una MdT debba calcolare una funzione dei dati presenti all'inizio del nastro. Cosa fa la MdT descritta in figura? C'è una configurazione finale? In caso, qual è? (nella figura il quadratino rappresenta lo spazio - blank, R destra, S stop)



a) È UNA MDT CHE DEFINISCE STRINGHE DI 0 E 1 CON I PARI O DISPARI.
LA CONFIGURAZIONE FINALE

5) Si consideri la seguente grammatica con un solo simbolo non terminale (S)

$$S \rightarrow SaS \mid SbS \mid c$$

- a. Fornire la sequenza di derivazioni della stringa cacbc
- b. definire il linguaggio generato dalla grammatica
- c. Stabilire se la grammatica è ambigua. Se sì, quali interventi si possono fare per eliminare questo problema?

a) $S \rightarrow SaS \rightarrow SaSbS \rightarrow cacbc$

b) IL LINGUAGGIO GENERATO SONO TUTTE LE STRINGHE DEL TIPO $cacbcacacbcab$, SEQUENZE DI a E b ALTERNATE DA DELLE c

$$L = \{c(ac + bc)^*\}$$

c) LA GRAMMATICA È AMBIGUA PERCHÉ POSSO OTTENERE LA STESSA STRINGA IN DUE MODI DIVERSI
LA RISCRIVIAMO COSÌ

$$S \rightarrow caS \mid cbS \mid c$$

6) Determinare l'insieme dei FIRST e dei FOLLOWER della seguente grammatica:

$$S \rightarrow aSTU \mid ST \mid \epsilon$$

$$T \rightarrow baT \mid abU \mid \epsilon$$

$$U \rightarrow babS \mid c$$

La grammatica risulta LL(1)?

$$\text{FIRST}(S) = \{a, \epsilon\} \quad \text{FIRST}(T) = \{b, a, \epsilon\} \quad \text{FIRST}(U) = \{b, \epsilon\}$$

$$\text{FOLLOW}(S) = \text{FOLLOW}(U) = \{b, a, c, \$\} \quad \text{FOLLOW}(T) = \{b, c, a, \$\}$$

	a	b	c	\$
S	$S \Rightarrow aSTU$ $S \Rightarrow ST$			
T				
U				

NON È LL(1) PERCHÉ ABBIAMO PIÙ DI UNA PRODUZIONE PER AVERE a DA S

7) a) Se qualcuno riuscisse a provare che $\text{NP} \supseteq \text{P}$ (strettamente), cosa possiamo dire sui problemi nella classe $\text{NP} \setminus \text{P}$?

b) Sapendo che 3CNF è NP-completo come posso utilizzare questo fatto per dimostrare che un altro problema è NP-completo?

c) Definire il problema PLI di programmazione lineare a numeri interi e illustrare una riduzione da 3CNF a PLI.

b) $A \rightarrow B$ A NON È PIÙ DIFFICILE DI B

c)

8) a) Illustrare la tesi di Church-Turing.

b) Spiegare se l'equivalenza stabilita nella tesi si applica anche agli automi a stati finiti.

b) NON POSSONO COMPUTARE, POSSONO SOLO LEGGERE (VERIFICARE)
SENZA RISOLVERE IL PROBLEMA

COMPITO 2

1. 1.a Illustrare il concetto di istruzione dominante motivando i vantaggi e gli svantaggi del suo utilizzo; fornire un esempio.

1.b Si considerino le seguenti affermazioni; per ognuna dire se è vera o falsa e se è falsa scrivere la versione corretta; $f(n)$ e $g(n)$ sono funzioni di n ($*$ denota la moltiplicazione).

$$O(f(n) * g(n)) = O(f(n)) * O(g(n)) \quad \checkmark$$

$$5 + 100 n + 3 n^2 = O(n^4) \quad \checkmark$$

$$5 + 100 n + 3 n^2 = O(n) \quad F \quad O(n^2)$$

$$n \log n + n + 3 n^2 = \Omega(n^3) \quad F$$

$$5 + 100 n + 3 n^2 = \Omega(n) \quad \checkmark$$

1.c Si considerino due software A e B per la gestione di un archivio: A richiede $(0,001 n)$ millesekondi in media, mentre B richiede $500 (n)^{1/2}$ millesekondi in media ($(n)^{1/2}$ = radice quadrata di n , n numero dei dati in archivio).

Quale dei due software è migliore secondo la notazione $O()$? **B**

Quale dei due software è migliore per archivi di 10^8 dati? **B**

2 Fornire la definizione di Macchina di Turing deterministica e giustificare brevemente l'importanza delle Macchine di Turing.

Se limitiamo il nastro di una macchina di Turing ad essere "semi-infinito", cioè illimitato in una direzione e limitato nell'altra, la macchina ha ancora pieno potere computazionale (cioè è equivalente ad una Macchina di Turing generale). Riesci a giustificarlo?

POSso SCALARE SEMPRE A DESTRA SUL NASTRO

3 Fornire la definizione di linguaggio decidibile, linguaggio indecidibile e linguaggio semi-decidibile.

DECIDIBILE

Il linguaggio delle stringhe che definiscono programmi python sintatticamente corretti e il problema della fermata a quale categoria appartengono? **INDECIDIBILE**

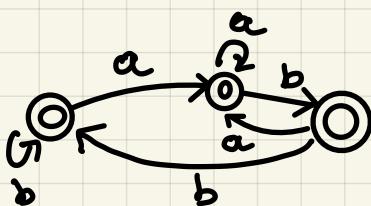
**DECIDIBILE SE È UNA MdT CHE DICE SE E O È L'UNA STRINGA
SEMIDECIDIBILE SE È UNA MdT CHE DICE SE È L'UNA STRINGA**

INDECIDIBILE SE NON È DECIDIBILE O SEMI

5. Costruire l'Automa a stati finiti che riconosce il seguente linguaggio

$$L_1 = \{w \in \{a,b\}^*: w \text{ non contiene mai tre la sequenza 'aba'}\}$$

Esistono linguaggi che sono riconoscibili da un automa a stati finiti non deterministico ma non da uno a stati finiti deterministico? Motivare brevemente la risposta.



6. Fornire l'espressione regolare che definisce il linguaggio costituito da tutte e sole le stringhe di caratteri a,b,c contenenti la sequenza abc e la sequenza ba (in qualunque posizione, cioè possiamo avere prima la stringa abc e poi la stringa ba o viceversa).

$$\left[(a+b+c)^* (abc)^+ (a+b+c)^* (ba)^+ (a+b+c)^* \right]^+ + \\ + \left[(a+b+c)^* (ba)^+ (a+b+c)^* (abc)^+ (a+b+c)^* \right]^+$$

7. Si consideri la seguente grammatica G (con assioma S e simboli terminali {a, b}):

$$S \rightarrow aAb \mid aSb$$

$$A \rightarrow aaAbb \mid ab$$

(1) Descrivere il linguaggio generato dalla grammatica

(2) Discutere se la grammatica è ambigua o no.

(3) Se ci sono produzioni inutili semplificare la grammatica eliminando le produzioni inutili; motivare la risposta.

a) $L = \{a^n b^n, n \geq 2\}$

b) S1

c) $S \rightarrow aAa \mid aSb$
 $A \rightarrow ab$

8. Sia data la seguente grammatica (S è assioma, S e A sono i simboli nonterminali, x e y sono i simboli terminali) con produzioni specificate nel seguito:

$$S \rightarrow A \mid xAy$$

$$A \rightarrow y \mid yx \mid A$$

- a) La grammatica non è LL(1) : motivare perché
- b) Riscrivere la grammatica per renderla LL(1)

$$\text{FIRST}(S) = \{x, y\} \quad \text{FIRST}(A) = \{y\} \\ \text{FOLLOW}(S) = \{\$\} \quad \text{FOLLOW}(A) = \{\$, y\}$$

COMPITO 3

3. Dimostrare che il linguaggio $\{0^h 1^k \mid h > k > 0\}$ è libero dal contesto e non è regolare.

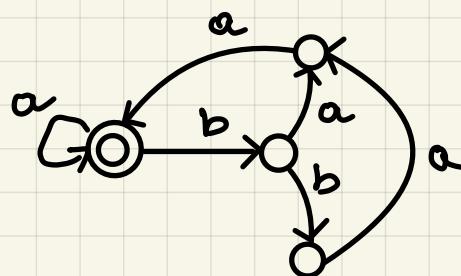
$\tilde{z}=6$

00011
v v w

U Vⁱ W $\in L$ $\forall i?$

000ⁱ1ⁱ1

6. Determinare un ASF per il linguaggio di alfabeto $\{a, b\}$ le cui parole sono caratterizzate dal fatto che dopo ogni b c'è a o b , sempre seguita da a . Esempio: $aabaabbaaa$ è nel linguaggio, mentre $aababa$ non lo è.



7. Si consideri la seguente grammatica G (con assioma S , simboli terminali $\{x, y\}$ e nonterminali S, A e B):

$$S \rightarrow AxyxA \rightarrow yB \quad B \rightarrow \epsilon \mid xyB \quad (\epsilon \text{ rappresenta la stringa vuota})$$

- a) Scrivere il linguaggio generato dalla grammatica
- b) Fornire l'albero di derivazione di una stringa del linguaggio con almeno 5 simboli terminali
- c) Calcolare gli insiemi First e Follow e rispondere se la grammatica è di tipo LL(1) o no

a) $L = \{y^n x^n, n \geq 3\}$

b)

$$\begin{array}{c} A \times yxA \\ \downarrow \\ yB \times yx yB \\ \swarrow \quad \downarrow \end{array}$$

c) $\text{FIRST}(S) = \{y\} \quad \text{FIRST}(A) = \{y\} \quad \text{FIRST}(B) = \{x, \epsilon\}$

$$\text{FOLLOW}(S) = \{\$\} \quad \text{FOLLOW}(A) = \{x, \$\} \quad \text{FOLLOW}(B) = \{x, \$\}$$

	x	y	\$
S	AxyxA		
A		yB	
B			

PUMPING LEMMA

$L = \{a^n b^n, n \geq 0\}$ NON REGOLARE

- 1) SI SCEGLIE UN $K \in \mathbb{N}$
- 2) SCEGLIO UNA STRINGA $x \in L$ CON $|x| \geq K$
- 3) SI DIVIDE x IN 3 SOTOSTRINGHE u, v, w
DOVE $|uv| \leq K$ E $|v| > 0$
- 4) SCRIVIARO $x = uv^i w$ $i \geq 0$
SE $\forall i, x \in L$ L È REGOLARE

ES

aaa bbb $n=3$

$u = \alpha\alpha$ $x = \alpha\alpha a^i bbb$
 $v = \alpha$
 $w = bbb$ MA SE $i=0 \rightarrow x = \alpha\alpha bbb \notin L$

IL LINGUAGGIO NON È REGOLARE

$L = \{a^n b^n c^n, n \geq 0\}$ TIPO 2

- 1) $K > 0$
- 2) STRINGA $x \in L$ CON $|x| \geq K$
- 3) $x = abcde$
 $|bcd| \leq K$ $|b| + |d| > 0$
- 4) $x = ab^{i_1} c d^{i_2} e$
 $\forall i_i \geq 0 \quad x \in L$ L DI TIPO 2 ALTRIMENTI NO

ES

$x = \alpha\alpha bbcc$ $n=2$

$K=4$ $bcd = abb$

$\alpha\alpha^i b b^{i_2} c \rightarrow \alpha bcc \notin L$

ES TOP DOWN

Esercizio 4.9. Progettare una tabella di parsing LL(1) per la grammatica

$$E \rightarrow -E \quad E \rightarrow (E) \quad E \rightarrow VT \quad T \rightarrow -E \quad T \rightarrow \lambda \quad V \rightarrow iU \quad U \rightarrow (E) \quad U \rightarrow \lambda$$

Tracciare quindi la sequenza di parsing con input **i-i((i))** e con input **i-(())i**.

$$\text{FIRST}(E) = \{-, (, i\} \quad \text{FIRST}(T) = \{-, \epsilon\} \quad \text{FIRST}(V) = \{i\} \quad \text{FIRST}(U) = \{(\, , \epsilon\}$$

$$\text{Follow}(E) = \text{Follow}(T) = \{\$\,)\} \quad \text{Follow}(V) = \{-, \$,)\} \quad \text{Follow}(U) = \{-, \$,)\}$$

	-	()	i	\$
E	-E	(E)		VT	
T	-E	E		E	
V				iU	
U	E	(E)	E	E	

PIRA	INPUT
\$E	i . i ((i))
\$TV	i - i ((i))
\$TU <i>i</i>	i . i ((i))
\$TU	- i ((i))
\$T	- i ((i))
\$E-	- i ((i))
\$E	i ((i))
\$TV	i ((i))
\$TUi	i ((i))
\$TU	((i))
\$T)E(((i))
\$T)E	(i))
\$T))E((i))
\$T))E	i))
\$T))TV	i))
\$T))TUi	i))
\$T))TU))
\$T))T))
\$T))))
\$T	\$
\$	ACCETTA

PRODUZIONE	DERIVAZIONE
$E \rightarrow VT$	TV
$V \rightarrow iU$	Ui
$U \rightarrow \lambda$	
$T \rightarrow -E$	$E -$
$E \rightarrow VT$	TV
$V \rightarrow iU$	Ui
$U \rightarrow (E)$	$)E($
$E \rightarrow (E)$	$)E($
$E \rightarrow VT$	TV
$V \rightarrow iU$	Ui
$U \rightarrow \lambda$	
$T \rightarrow \lambda$	
$T \rightarrow \lambda$	

ES BOTTOM-UP

$$\begin{array}{ll}
 S' \rightarrow S & E \rightarrow T+E \\
 S \rightarrow E & T \rightarrow \text{id} \\
 E \rightarrow T; & T \rightarrow (E)
 \end{array}$$

