1 * Definire Ld e dimostrare che non è RE.

Ld, o linguaggio di diagonalizzazione, è ~~il linguaggio~~ l'insieme delle stringhe w tali che ~~codificano una macchina di Turing che non accetta come input~~ la macchina codificata da w non accetta l'input w.

Intuitivamente, Ld non è RE in quanto non esiste alcuna TM che accetti ... DIM

2 * Classe P, NP, PS, RP, ZPP (definizioni) (+RE RICORSIVI)

3 * Dimostrare che se P_1 è PS completo e $P_1 \in P$ allora $P = PS$.
(+ Dimostrare che se esiste una riduzione polinomiale da P_1 a P_2 e $P_2 \in NP$ e $P_1 \in NP$-completi allora P_2 è NP-completo)

4 * Descrivere una T.M. che avendo sul nastro il numero n (>2) scritto in binario si arresta con il numero n+2 scritto in binario.

2) La classe dei problemi P (polinomiali) è quella classe di problemi per cui esiste un polinomio $T(N)$ tale che $L = L(M)$ per una macchina di Turing M deterministica di complessità in tempo $T(N)$.

Un linguaggio L è nella classe NP se esiste una NTM M e una complessità polinomiale in tempo $T(N)$ tale che $L = L(M)$ e in M non ci sono sequenze di mosse più lunghe di $T(N)$.

ORALE:
Teorema di Rice: enunciato e dimostrazione.
Dimostrazione della Savitch con pseudocodice.

4 *

Da N a N + 2

3 $\overset{2}{11} \xrightarrow{5} \overset{3}{101}$

4 $\overset{3}{100} \xrightarrow{6} \overset{3}{110}$

5 $\overset{3}{101} \xrightarrow{7} \overset{3}{111}$

6 $\overset{3}{110} \xrightarrow{8} \overset{4}{1000}$

7 $\overset{3}{111} \xrightarrow{9} \overset{4}{1001}$

15 $\overset{4}{1111} \xrightarrow{17} \overset{5}{10001}$

27 $\overset{5}{11011} \xrightarrow{29} \overset{5}{11101}$

32 $\overset{6}{100000} \xrightarrow{34} \overset{6}{100010}$

36 $\overset{6}{100100} \xrightarrow{38} \overset{6}{100110}$

	0	1	B	
q_0	$(q_0, 0, R)$	$(q_1, 1, R)$	–	primo bit
q_1	$(q_1, 0, R)$	$(q_1, 1, R)$	(q_2, B, L)	scorrimento della stringa
q_2	$(q_3, 0, L)$	$(q_3, 1, L)$	–	salto dell'ultimo bit
q_3	$(q_2, 1, L)$	$(q_4, 0, L)$	–	incremento del penultimo bit
q_4	$(q_2, 1, L)$	$(q_4, 0, L)$	$(q_2, 1, L)$	Accetta su 0, permuta gli 1

Da N a N - 2

(3) 11 → (1) 101

(4) 100 → (2) 10

(5) 101 → (3) 11

(6) 110 → (4) 100

(7) 111 → (5) 101

(15) 1111 → (13) 1101

(27) 11011 → (25) 11001

(32) 100000 → (30) 11110

(36) 100100 → (34) 100010

(61) 111101 → (59) 111010 ~~0001~~

(40) 101000

(38) 100110

(72) 1001000

(70) 1000110

- Il primo bit lo salti
- Il secondo lo permuti
  - se è un 1 esci
  - se è uno 0 prosegui
    - ogni 0 lo permuti e
    - il primo 1 lo permuti ed esci.