Exercise 2

1.

pre: $nIn>2 \wedge nIn=nOut \wedge \forall i \, (\, 0 \le i < nIn \rightarrow in[i] > 0 \,)$

post: let $\text{even(i)} \leftrightarrow \exists k(i=2k)$ and $\text{odd(i)} \leftrightarrow \exists k(i=2k+1)$; then the postcondition is

$$\forall i \, (\quad (\, 0 \le i < nIn \wedge even(in[i]) \rightarrow \exists j \, (\, 0 \le j < nOut \wedge odd(\text{j}) \wedge in[i]=out[j] \,) \,) \wedge$$
$$(\, 0 \le i < nIn \wedge odd(in[i]) \rightarrow \exists j \, (\, 0 \le j < nOut \wedge even(j) \wedge in[i]=out[j] \,) \,) \,)$$
$$)$$

2. Given the stated precondition, the postcondition is not satisfiable in case the number of even-valued (or, respectively, odd-valued) elements of array *in* exceeds the number of odd index positions (resp., even index positions) in array *out*; for instance if *in*=[2, 8, 6, 4]
3. If the size of the *out* array parameter is at least twice that of the array *in*, i.e., if $nOut \ge 2 \cdot nIn$, then all elements of the array in can be stored in suitable positions of the array *out* in all cases.

Exercise 3

The satisfiability of an MFO sentence is decidable, because there is an algorithm that, for any given MFO sentence *F*, builds an equivalent FSA, that is, an FSA that accepts exactly the set of strings that satisfy the sentence *F*. Therefore the sentence *F* is satisfiable if and only if the FSA equivalent to it accepts a non-empty language; this latter problem is notoriously decidable.

Exercise 4

1. The TM first reads the first sequence of *a*'s and stores in the tape, in unary using e.g. a '+' symbol, its length, i.e., the number h_1. Then it reads the sequence of *b*'s while it moves leftward on the memory tape; if it reaches the end of the sequence of *b*'s before the first '+' on the memory tape then it changes the current '+' into a '*', otherwise it rejects the string. Then it continues in a similar way reading the sequences of *a*'s and *b*'s, alternating scans of the memory tape from left to right (while reading *a*'s) and from right to left (while reading *b*'s). It accepts the input string if and only if the last sequence of letters is composed of *b* letters.
   Let $n = |x|$ be the input string length; then the time complexity is $\Theta(n)$ and also the space complexity is $\Theta(n)$; in the worst case, i.e., that of a string of type a^n, a string of *n* symbol '+' is written on the memory tape.
   In case the input string is accepted the worst space complexity is $S(n) = n - 1$ (string of type $\boldsymbol{a^{n-1}b}$) and the best case complexity is $S(n)= \sqrt{n}$ (string of type $\boldsymbol{a^k b^{k-1} a^{k-2} b^{k-3} \ldots a^2 b}$).

2. A RAM machine reads all sequences and counts their elements, checking that the length of each sequence is less than that of the previous one. With the uniform cost criterion the space complexity is $\Theta(1)$ (a constant amount of memory is needed), while the time complexity is $\Theta(n)$. With the logarithmic cost criterion the space complexity is $\Theta(\log n)$ (for storing a fixed number of counters) and the time complexity is $\Theta(n \log n)$ (for computing the value of the largest possible counter by a sequence of *n* increment operations).

Theoretical Computer Science Exam, January 19th, 2022

The exam consists of **4 exercises**. Available time: 1 hr 30 min.; you may write your answers in Italian, if you wish.

All exercises must be addressed for the written exam to be considered sufficient.

F.NAME …….. L.NAME …… PERSON CODE ……

Exercise 1 (8 pts)

Consider the following language

$$L = \{ b\, a^i\, b^j\, a^k \mid 1 \leq i \leq k \ \wedge\ j > 0 \}$$

For instance, strings *baabbbaaa*, *baabbaa* $\in L$, while ε, *bbabaa*, *baaaba*, *aabaaa* $\notin L$.

- Design an automaton, of the least powerful type, that accepts language L.
- Write a grammar, of the least general type, that generates language L.

Exercise 2 (8 pts)

Consider a subprogram, called *decrIncr*, having three parameters: an array of integers *in*, its size *nIn*, an array of integers *out*. The input parameters are: *in* and *nIn*, whereas the array *out* is an output parameter. The two arrays can be assumed to have the same size (i.e., number of components), greater than 0; all the elements of array *in* are assumed to be strictly positive; in the input array *in* there is a *minimum* element, that is, an element whose value is strictly less than all the other ones.

After execution of the subprogram the minimum value of array *in* (let us call it *m*) is written, in the same position, into the array *out*; furthermore, all the components of array out in the preceding positions are equal to the elements having the same position in array *in* decremented by the value *m*, and all elements in the following positions are equal to the elements having the same position in array *in* incremented by the value *m*. For instance, the specification is satisfied by the following values: *nIn*=6, *in*=[7, 9, 5, 2, 11, 6], *out*=[5, 7, 3, 2, 13, 8]; the minimum element is 2 at the fourth position, all elements of array *in* in the previous positions (7, 9, 5) are inserted into array *out* decremented by 2 (5, 7, 3), and all elements of array *in* in the following positions (11, 6) are inserted into array *out* incremented by 2 (13, 8).

Please answer the following questions:

1. write pre- and post-conditions specifying the requirements of the above described procedure; assume that indices of the arrays start at 1;
2. for each of the following input/output data say if the procedure *decrIncr* satisfies the specification if it computes the given value of the *out* array when it receives the given input parameter values; please provide suitable justifications for your answers.

   a) *nIn*=6, *in*=[2, 7, 9, 5, 11, 6], *out*=[2, 9, 11, 7, 13, 8]

   b) *nIn*=6, *in*=[7, 9, 2, 5, 11, 6], *out*=[9, 11, 2, 3, 9, 4]

   c) *nIn*=6, *in*=[2, 2, 2, 2, 2, 2], *out*=[4, 9, 2, 5, 11, 6]

Exercise 3 (8 pts)

Assume that a generic regular grammar G_R is given.

Answer the following questions, providing precise, clear justifications.

- Given also a generic Turing machine M, is the problem of determining wether M accepts the language generated by G_R solvable?

- Given also a generic finite state automaton A, is the problem of determining wether A accepts the language generated by G_R solvable?

Exercise 4 (8 pts)

Consider the language composed of strings $x \in \{a, b\}^*$ such that $\#a(x)=2\cdot\#b(x)$, i.e., strings for which the number of a characters is twice the number of b characters. Please answer the following questions.

- Sketch a simple Turing machine that accepts this language. Design this machine so that its time complexity is minimal, and provide an estimation of its time and space complexity as a function of $n=|x|$, the length of the input string; suitably justify the complexity figures.

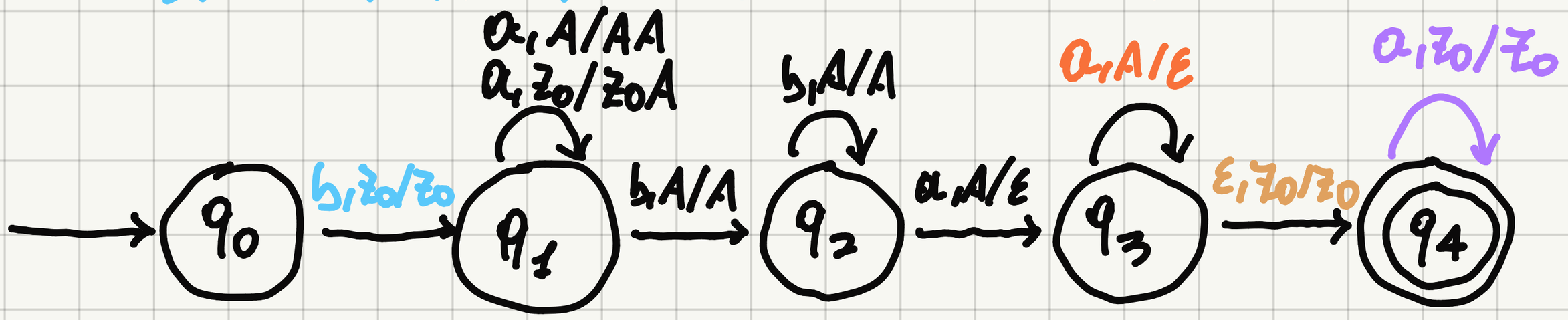
- Sketch another Turing machine that accepts this language with a minimal space complexity, and provide an estimation of its time and space complexity as a function of $n=|x|$, the length of the input string; suitably justify the complexity figures.

1 a

È RICHIESTO UN AUTOMA A PILA PER GESTIRE LA CONDIZIONE $\forall 1 \leq i \leq k$

CONTINUO FINO K = i CASO K > i

DELLA PRIMA b NON CI INTERESSA PER LE CONDIZIONI



MI SPOSTO NELLO STATO FINALE

$$S \xrightarrow{b} b S_1$$

$$S_1 \rightarrow a S_1 a \mid a S_1 aa \mid a S_2 a \mid a S_2 aa$$

$$S_2 \rightarrow b S_2 \mid b$$

2 INPUT:

- in ARRAY DI DIMENSIONE nIn DI ELEMENTI POSITIVI
- nIn > 0

OUTPUT:

- out ARRAY DI DIMENSIONE nIn

PRECONDIZIONI] $nIn > 0 \wedge \forall i (0 \leq i < nIn \rightarrow in[i] > 0) \wedge \exists m (0 \leq m < nIn \wedge \forall i (0 \leq i < nIn \wedge i \neq m \rightarrow in[m] < in[i]) \wedge i \neq m$

POSTCONDIZIONI] $\exists m (0 \leq m < nIn \wedge \forall i (0 \leq i < nIn \rightarrow in[m] < in[i]) \wedge out[m] = in[m] \wedge \forall j (0 \leq j < m \rightarrow out[j] = in[j]) \wedge \forall j (m < j < nIn \rightarrow out[j] = in[j]))$

a SÌ. MINIMO IN int[0]=2

b NO POICHÈ INCREMENTO/DECREMENTO SONO SVOLTI ERRONEAMENTE

c NO POICHÈ in[m]<in[n] NON È RISPETTATA

3

a NO, POICHÈ SI STANNO CONSIDERANDO INFINITI CASI POSSIBILI AFFINNÈ.

QUESTO COMPORTA PROPRIO IL PROBLEMA DELL'IRRISOLVIBILITÀ ALGORITMICA (ULTERIORI INFO A PAG. 327 DEL LIBRO)

b SI, POICHÈ IN GENERALE È SEMPRE POSSIBILE DEFINIRE UN AUTOMA A STATI FINITI CHE RICONOSCA UNA GRAMMATICA REGOLARE. CIÒ NON SIGNIFICA CHE L'AUTOMA RAGGIUNGA UNO STATO FINALE. È SOLO RICHIESTO SE È DEFINIBILE.

4 a

È NECESSARIA UNA MACCHINA DI TURING A 2 NASTRI, UNO PER LE a E UNO PER LE b E.E RIPERCORRENDO TUTTA LA MEMORIA SARÀ POSSIBILE STABILIRE SE #a = 2#b

SIA n LA LUNGHEZZA DELLA STRINGA

COMPLESSITÀ TEMPORALE: $T(n) = \tilde{\Theta}(n+n) = \tilde{\Theta}(n)$

COMPLESSITÀ SPAZIALE: $S(n) = \tilde{\Theta}(n)$

b

UNA MACCHINA DI TURING EQUIVALENTE PUÒ EFFETTUARE IL CONFRONTO TRA LE CARDINALITÀ TRATTANDOLA IN NUMERI BINARI E, QUINDI, CON UNA COMPLESSITÀ CHE DIVENTA LOG-ARITMICA IN QUANTO

$$|n|_{10} = 10^{\log n} = |n|_2$$

COMPLESSITÀ TEMPORALE: $T(n) = \Theta(n \log n)$, PEGGIORE

COMPLESSITÀ SPAZIALE: $S(n) = \Theta(n \log n)$, PEGGIORE

Theoretical Computer Science Exam, February 8th, 2022

The exam consists of **4 exercises**. Available time: 1 hr 30 min.; you may write your answers in Italian, if you wish.

All exercises must be addressed for the written exam to be considered sufficient.

F.NAME …….. L.NAME …… PERSON CODE ……

Exercise 1 (8 pts)

1. Consider the following language

$$L_1 = \{ a^m b^n \mid m > 0 \ \wedge \ n > 0 \ \wedge \ n > m \ \wedge \ n - m \text{ is even} \ \}$$

For instance, strings *abbb*, *aabbbb, abbbbb* $\in L_1$, while ε, *abb*, *aaabbb* $\notin L_1$.

a) Design an automaton, of the least powerful type, that accepts language L_1.

b) Write a computation of the automaton accepting string *aabbbb*

c) Write a grammar, of the least general type, that generates language L_1.

d) For the grammar generating L_1 write a derivation of the string *aabbbb*

2. Next, consider the language

$$L_2 = \{ a^m b^n \mid m > 0 \ \wedge \ n > 0 \ \wedge \ m > n \ \wedge \ m - n \text{ is even} \ \}$$

For instance, strings *aaab*, *aaaabb, aaaaab* $\in L_2$, while ε, *aab*, *aaabbb* $\notin L_2$.

Consider the language $L_3 = L_1 \cap L_2$

a) Design an automaton, of the least powerful type, that accepts language L_3.

b) Write a grammar, of the least general type, that generates language L_3.

Exercise 2 (8 pts)

Consider the following languages L_1 and L_2 over the alphabet $\{a, b, c\}$, defined as follows.

L_1 includes the strings where there is at least one pair of *a*'s that are separated only by letters *b*. For instance, *bcabbbabbcba* $\in L_1$, while *cabcabbcab* $\notin L_1$.

L_2 includes the strings where all pairs of *a*'s are separated by an odd number of letters, and at least one of these is a *c*. For instance, *bacbbacac* $\in L_2$, while *babcbabac* $\notin L_2$.

a) Write a sentence, in a monadic logic over words, first-order or second-order only if necessary, that characterizes the language L_1.

b) Write a sentence, in a monadic logic over words, first-order or second-order only if necessary, that characterizes the language L_2.

Exercise 3 (8 pts)

Consider the following two sets S_1 and S_2.

$$S_1 = \{ k \mid \forall x \ (f_k(x) = \perp \ \vee \ f_k(x) > x) \ \}$$

$$S_2 = \{ k \mid \exists x \, \exists h \ (\ h > 0 \ \wedge \ f_k(x) = x - h) \ \}$$

Answer the following questions, providing precise, clear justifications.

a) Is set S_1 decidable? Is it semidecidable?

b) Is set S_2 decidable? Is it semidecidable?

c) Is set $S_1 \cap S_2$ decidable? Is it semidecidable?

Exercise 4 (8 pts)

Given a string x over a k-element alphabet $\sum = \{a_1, \ldots a_k\}$ sketch (that is, describe informally but precisely) a multitape Turing machine that, taking x as input, determines which is the symbol of the alphabet that occurs most times in x. In case different symbols occur a maximal number of times, then the machine indicates arbitrarily one of them. For instance, in the case when $\sum = \{ a, b, c\}$ and $x = cbcbaabcb$ the machine indicates b as the symbol occurring most times; if x = *cbcbaabc* the machine can indicate either b or c. You can choose the number of tapes of the Turing machine.

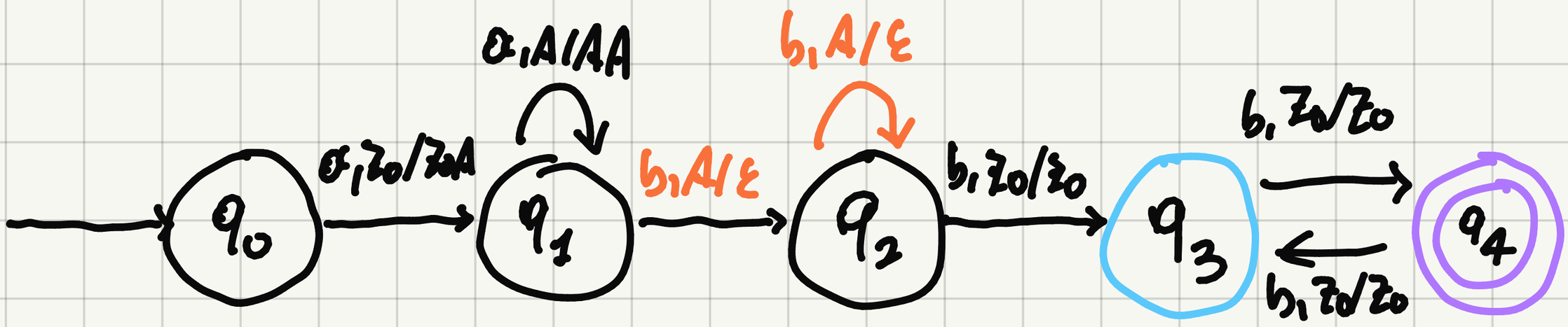
Analyze the time and space complexity of the Turing machine as a function of the length of the input string $n = |x|$.

Then sketch (that is, describe informally but precisely) a RAM machine that performs the same computation and analyze its time and space complexity, still as a function of the length of the input string $n = |x|$, using both the uniform and the logarithmic cost criterion.

Please provide suitable justifications for the results of your analysis.

1 1

a OCCORRE UN AUTOMA A PILA PER GESTIRE LA CONDIZIONE SUGLI ESPONENTI. DA NOTARE CHE $n > m$, PER CUI LA STRATEGIA MIGLIORE È "PAREGGIARE" LE n E POI SALTELLARE TRA LO STATO "n-m DISPARI" E "n-m PARI"



b $\langle q_0, aabbbb, Z_0 \rangle \vdash \langle q_1, abbbb, Z_0 A \rangle \vdash \langle q_1, bbbb, Z_0 AA \rangle$

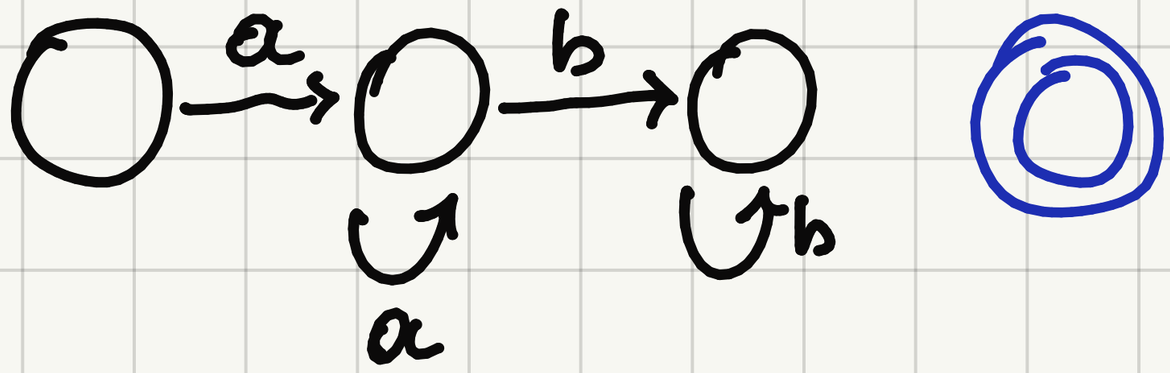
$\vdash \langle q_2, bbb, Z_0 A \rangle \vdash \langle q_2, bb, Z_0 \rangle \vdash \langle q_3, b, Z_0 \rangle \vdash \langle q_4, \varepsilon, Z_0 \rangle$

c $S \to aSb \mid aS_1b \quad S_1 \to S_1bb \mid bb$

d $S \to aSb \to aaS_1bb \to aabbbb$

2

$L_3 = \{ a^m b^n \mid m>0 \wedge n>0 \wedge \emptyset \}$

a 

b $S \to aS \mid Sb$

2

L_1

$\exists x (a(x) \wedge \exists y (a(y) \wedge y > x) \wedge \forall z (x < z < y \rightarrow b(z))$

L_2

INTRODUCO IL PREDICATO $D(x)$: "NUMERO DISPARI DI LETTERE"

$\forall x \forall y (a(x) \wedge a(y) \wedge y > x \rightarrow$

$(\exists z (x < z < y \rightarrow c(z)) \wedge$

$(\forall z (x < z < y \rightarrow$

$(first(z) \rightarrow D(z)) \wedge$

$\neg last(z) \wedge D(z) \rightarrow \neg D(z+1) \wedge$

$\neg last(z) \wedge \neg D(z) \rightarrow D(z+1) \wedge$

$last(z) \leftrightarrow D(z))$

3 a

S_1 NON È DECIDIBILE PER RICE THEOREM, POICHÈ NON È L'INSIEME DI TUTTE LE FUNZIONI COMPUTABILI ($\varphi_k(x) = \perp$). PER SEMPLICITÀ, SI STUDIA LA SEMIDECIDIBILITÀ DEL COMPLEMENTARE $\neg S_1 = \{k \mid \exists x (\varphi_k(x) \neq \perp \wedge \varphi_k(x) \leq x)\}$. ESSENDO IMMAGINE

DI UNA FUNZIONE TOTALE E COMPUTABILE, $\neg S_1$ È DECIDIBILE

→ S_1 È NON SEMIDECIDIBILE

b

S_2 NON È DECIDIBILE PER RICE THEOREM, POICHÈ NON È L'INSIEME DI TUTTE LE FUNZIONI COMPUTABILI. TUTTAVIA È IMMAGINE DI UNA FUNZIONE TOTALE E COMPUTABILE → S_2 È SEMIDECIDIBILE

c

$S_1 \cap S_2 = \emptyset$. PER RICE THEOREM, È DECIDIBILE → SEMIDECIDIBILE

4 a

È NECESSARIA UNA MACCHINA DI TURING A K NASTRI, UNO PER OGNI LETTERA DELL'ALFABETO Σ. DOPO LA LETTURA DELLA STRINGA, ESSA VIENE RISCANSIONATA PER EFFETTUARE L'OPPORTUNO CALCOLO DI BC CARATTERE PIÙ FREQUENTE

COMPLESSITÀ TEMPORALE: $T(n) = \tilde{\Theta}(k \cdot n) = \tilde{\Theta}(n)$

COMPLESSITÀ SPAZIALE: $S(n) = \tilde{\Theta}(n)$

b

UNA MACCHINA RAM RICHIEDE K CELLE DI MEMORIA PER I K

CONTATORI, UNO PER OGNI LETTERA DELL'ALFABETO Σ. PRESA IN INPUT LA STRINGA, VIENE LETTA CARATTERE PER CARATTERE E L'OPPORTUNO CONTATORE.

COSTO UNIFORME

COMPLESSITÀ TEMPORALE: $T(n) = \approx(n)$

COMPLESSITÀ SPAZIALE: $S(n) = \Theta(k) = \simeq(1)$

COSTO LOGARITMICO

COMPLESSITÀ TEMPORALE: $T(n) = \approx(n \cdot \log n)$

COMPLESSITÀ SPAZIALE: $S(n) = \Theta(k) = \simeq(\log n)$

Theoretical Computer Science Exam, July 8, 2022

The exam consists of **4 exercises**. Available time: 1 hr 30 min.; you may write your answers in Italian, if you wish.

All exercises must be addressed for the written exam to be considered sufficient.

F.NAME …….. L.NAME …… PERSON CODE ……

Exercise 1 (8 pts)

Consider the language

$$L = \{\ a^{2n+1}\ b^{3n-1} \mid n > 0\ \ \}$$

For instance, strings *aaabb*, a^5b^5, $a^7b^8 \in L$.

a) Write a grammar, of the least general type, that generates language *L*. For this grammar, write a derivation of the string a^5b^5.
b) Design an automaton, of the least powerful type, that accepts language *L*. Write a computation of the automaton for the string *aaabb* and one for the string a^5b^5.

Exercise 2 (8 pts)

Consider a subprogram, called *ratios*(*in*, *nIn*, *out*), having three parameters: an array of integers *in*, its size *nIn*, an array of integers *out*. The input parameters are: *in* and *nIn*, whereas the array *out* is an output parameter. The two arrays can be assumed to have the same size (i.e., number of components), greater than 1; all the elements of array *in* are assumed to be strictly positive and strictly increasing; every element of the array *in*, except the first one, is a multiple of the preceding one. For instance, an acceptable input array *in* is the following [3, 9, 18, 54, 270].

After execution of the subprogram the first element of array *in* is also stored, in the first position, into the array *out*; furthermore, every element of array *out* in the other positions is equal to the value of the element having the same position in array *in* divided by the previous element in array *in*. For instance, the specification is satisfied by the following values: *nIn*=5, *in*=[3, 9, 18, 54, 270], *out*=[3, 3, 2, 3, 5].

Please answer the following questions:

1. write pre- and post-conditions specifying the requirements of the above described procedure; assume that indices of the arrays start at 0;
2. for each of the following input/output data say if the procedure *ratios* satisfies the specification if it computes the given value of the *out* array when it receives the given input parameter values; please provide suitable justifications for your answers.

   a) *nIn*=5, *in*=[2, 7, 9, 11, 6], *out*=[2, 9, 7, 13, 8]

   b) *nIn*=5, *in*=[2, 6, 12, 48, 96], *out*=[2, 3, 2, 3, 2]

   c) *nIn*=5, *in*=[2, 6, 12, 48, 96], *out*=[2, 3, 2, 4, 2]

Exercise 3 (8 pts)

Let us consider a fixed finite set S of natural numbers.

1. Is the set S', that includes exactly the natural numbers that are multiple of all elements of S, decidable?
2. Is the set S'' of Turing machines that, starting the computation with an empty input tape, *do not* print all the numbers that belong to S decidable?
3. Is the set S'' of Turing machines that, starting the computation with an empty input tape, *do not* print all the numbers of S semidecidable?

Please provide precise, detailed explanations for your answers.

Exercise 4 (8 pts)

The following C-like program reads from the input a positive integer value n and writes the first n powers of 2, that is, 2, 2^2, … 2^n.

```
int n, ev;
read(n);
ev = 1;
for ( i from 1 to n ) {
   ev = 2 * ev;
   write(ev);
}
```

Assuming that the algorithm, after a standard translation, is executed on a RAM machine, evaluate its time and space complexity class using as parameter the ***value*** of input n, adopting both the uniform and the logarithmic cost criteria.

In addition, evaluate the same complexity figures using as parameter the input ***size***, that is, the length x of the string that encodes, in a standard way, the value of n.

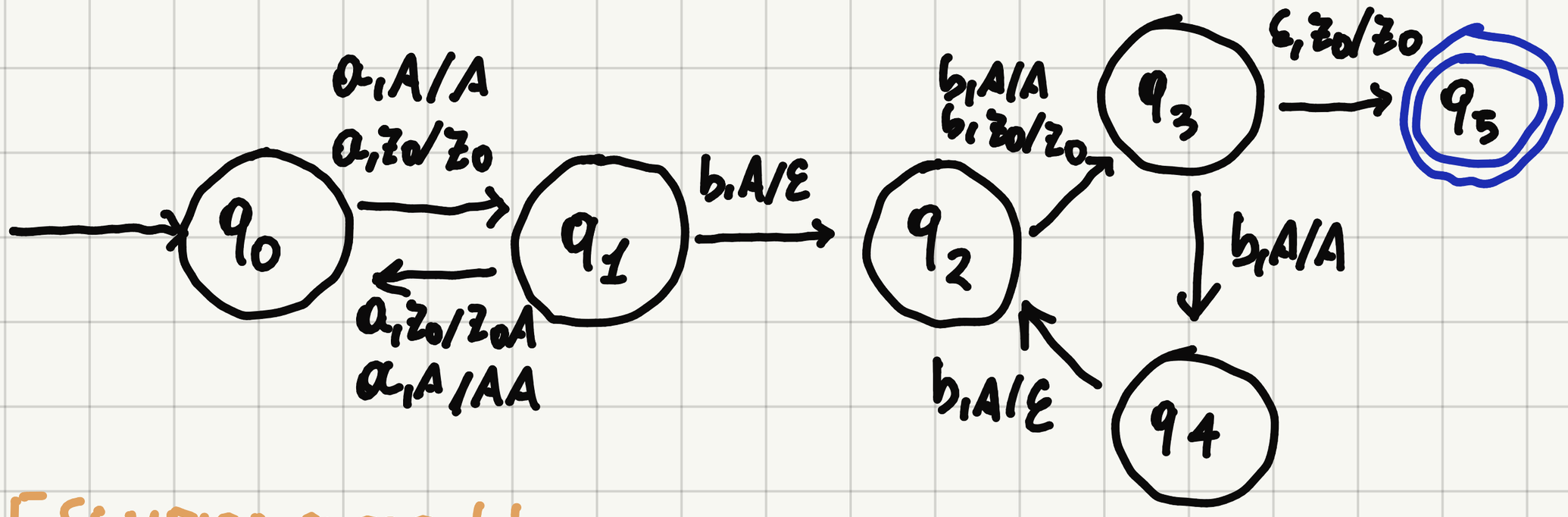
1

a $S \to aaabb \mid aaSbbb$ CASO $n=1$ $n>1$. QUANTE a E b IN PIÙ?

ESEMPIO: $aaaaabbbbb$

$S \to aaSbbb \to aaaaabbbbb$

b PER GESTIRE LA CONDIZIONE SUGLI ESPONENTI, OCCORRE UN AUTOMA A PILA



ESEMPIO: $aaaabb$

$\langle q_0, aaaabb, z_0\rangle \vdash \langle q_1, aaabb, z_0\rangle \vdash \langle q_0, aabb, z_0A\rangle \vdash$

$\vdash \langle q_1, abb, z_0\rangle \vdash \langle q_2, b, z_0\rangle \vdash \langle q_3, \varepsilon, z_0\rangle \vdash \langle q_5, \varepsilon, z_0\rangle$

ESEMPIO: $a^5b^5 = aaaaabbbbb$

$\langle q_0, aaaaabbbbb, z_0\rangle \vdash \langle q_1, aaaabbbbb, z_0\rangle$

$\vdash \langle q_0, aaabbbbb, z_0A\rangle \vdash \langle q_1, aabbbbb, z_0A\rangle$

$\vdash \langle q_0, abbbbb, z_0AA\rangle \vdash \langle q_1, bbbbb, z_0AA\rangle \vdash$

$\langle q_2, bbbb, z_0A\rangle \vdash \langle q_3, bbb, z_0A\rangle \vdash \langle q_4, bb, z_0A\rangle \vdash$

$\langle q_2, b, z_0\rangle \vdash \langle q_2, \varepsilon, z_0\rangle \vdash \langle q_5, \varepsilon, z_0\rangle$

2 INPUT:

- in ARRAY DI DIMENSIONE nIn
- nIn>1

OUTPUT:

- OUT ARRAY DI DIMENSIONE nIn

[PRECONDIZIONI] $nIn>1 \land a[0]>0 \land \forall i\,(0<i\leq nIn \rightarrow a[i]>a[i-1])$

[POST CONDIZIONI] $out[0]=in[0] \land \forall i\,(0<i\leq nIn \rightarrow$
$\rightarrow out[i]=in[i]+in[i-1])$

- PRECONDIZIONI NON RISPETTATE

- POST CONDIZIONI NON RISPETTATE (3≠48:12)

- PRECONDIZIONI E POST CONDIZIONI RISPETTATE

3 a SI, POICHÈ È SEMPRE POSSIBILE DETERMINARE, DATO UN QUALUNQUE NUMERO NATURALE, UNA PROCEDURA CHE TERMINA SE È MULTIPLO DI TUTTI GLI ELEMENTI IN S

b NON ESSENDO L'INSIEME DI TUTTE LE FUNZIONI COMPUTABILI, PER RICE THEOREM È NON DECIDIBILE

c "¬S" È L'INSIEME DELLE MACCHINE DI TURING CHE STAMPA TUTTI GLI ELEMENTI DI S. TALE INSIEME È IMMAGINE DI UNA FUNZIONE TOTALE

È COMPUTABILE ED È QUINDI SEMIDECIDIBILE → S'' È NON SEMIDECIDIBILE

4 SIANO M[1] → n E M[2] → c(n) LE CELLE DI MEMORIA RICHIESTE.

COSTO UNIFORME

T(n) = Θ(n) (CICLO FOR)

S(n) = Θ(1) (NUMERO FISSO DI CELLE DI MEMORIA RICHIESTO)

COSTO LOGARITMICO

$S(n) = \Theta(\log 2^n) = \Theta(n)$

DUE CELLE DI MEMORIA RICHIESTE: "CONTENUTO" DA TRADURRE (IN BINARIO)

LA COMPLESSITÀ TEMPORALE È FORTEMENTE INFLUENZATA DA c(n)·2+c(n)

LOAD 2 $\log 2 + \log 2^{i-1}$

MULT= 2 $\log 2 + \log 2 + \log 2^{i-1}$

STORE 2 $\log 2 + \log 2^{i}$

$T(n) = \sum_{i=1}^{n} (K + 3\log 2^{i}) \simeq Kn + \frac{3}{2} n(n+1) = \Theta(n^2)$

USANDO size PER IL VALORE DI n, $x = \log n \rightarrow n = 2^x$ E

COSTO UNIFORME $T(n) = \Theta(2^x)$ $S(n) = \Theta(1)$

COSTO LOGARITMICO $T(n) = \Theta(2^{2x})$ $S(n) = \Theta(2^x)$

Theoretical Computer Science Exam, January 16, 2023

The exam consists of **4 exercises**. Available time: 1 hr 30 min.; you may write your answers in Italian, if you wish.

All exercises must be addressed for the written exam to be considered sufficient.

F.NAME …….. L.NAME …… PERSON CODE ……

Exercise 1 (8 pts)

Consider the language

$$L = \{\, a^{2n}\, b^{2m}\, c^{k} \mid n \geq 0 \ \wedge\ m \geq 1 \ \wedge\ k \geq 1 \,\}$$

For instance, strings *aaaabbc*, *bbbbcc* $\in L$.

a) For the strings of language L, consider the translation $\tau_1(\, a^{2n}\, b^{2m}\, c^{k}\,) = a^{n}\, b^{3m}\, c$ and design an automaton of the least powerful type that computes such translation.

b) For the strings of language L, consider the translation $\tau_2(\, a^{2n}\, b^{2m}\, c^{k}\,) = b^{3m}\, a^{n}\, c^{\lfloor k\,/\,2 \rfloor}$; for instance, $\tau_2(\, aaaabbccccc\,) = bbbaacc$. Design the automaton of the least powerful type that computes such translation.

Exercise 2 (8 pts)

Consider a subprogram, called *checkOccurr*(*in*, *nIn*, *out*), having three parameters: an array of characters *in*, its size *nIn*, a Boolean *out*. The input parameters are: *in* and *nIn*, whereas *out* is an output parameter.

It is assumed that the number of elements of the array *in* is greater than 4, and that at most two of the elements of array *in* are equal to the character ***a***.

After execution of *checkOccurr* the output parameter *out* must be true if and only if

- the array *in* includes two distinct elements equal to character *a*, and
- the segment of the array *in* included within the two elements equal to the character *a* either
  - includes at least one element equal to character *b*, or
  - is composed entirely of characters ***c*** or is composed entirely of characters ***d***

1. Write pre- and post-conditions specifying the above described requirements for subprogram *checkOccurr*. Please assume that the array indices range from 1 to *nIn*.
2. For each of the following input/output data say if the procedure *checkOccurr* satisfies the specification if it computes the given value of the *out* parameter when it receives the given input parameter values; please provide suitable justifications for your answers.
   a) *nIn* = 6, *in* = [*b, a, c, b, c, a*], *out* = true
   b) *nIn* = 6, *in* = [*b, a, c, c, a, a*], *out* = false
   c) *nIn* = 6, *in* = [*b, a, b, b, a, b*], *out* = true
   d) *nIn* = 6, *in* = [*b, a, c, d, a, b*], *out* = true

Exercise 3 (8 pts)

Consider the set of prime numbers sorted in increasing order, so that one can refer to the first, the second, … the i-th prime number.

Please answer the following questions; provide suitable concise justifications for your answers.

1. Is the set S1 = { $\langle x, y \rangle$ | y is the x-th prime number } decidable? Is it semidecidable?
2. Is the set S2 = { z | $\forall x \, \forall y$ ($f_z(x, y) = 1$ if y is the x-th prime number; $f_z(x, y) = 0$ otherwise) } decidable?
3. Is the set S3 = { z | $\exists x \, \exists y$ ($f_z(x, y) = 1$ if y is the x-th prime number ; $f_z(x, y) = 0$ otherwise) } decidable? Is it semidecidable?
4. Is the set S4 = { $\langle x, y \rangle$ | $\exists z$ ($f_z(x, y) = 1$ if y is the x-th prime number; $f_z(x, y) = 0$ otherwise) } decidable? Is it semidecidable?

Exercise 4 (8 pts)

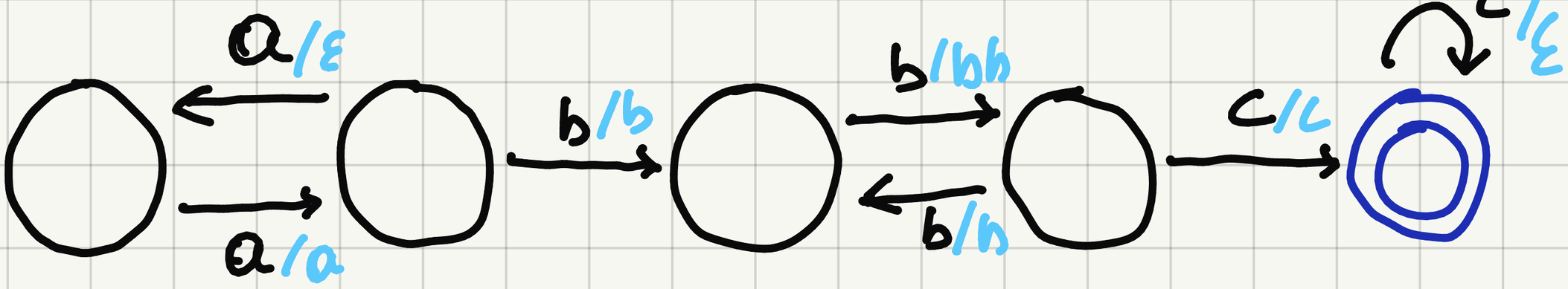
Consider the following C-like program, and assume that it reads from the input a ***positive*** value for variable n.

```
int n, i, s, k;
read(n);
i = 1;  s = 1;
while (i < n) {
       i = i + 1;
       k = i * i;
       s = s + k;
}
```

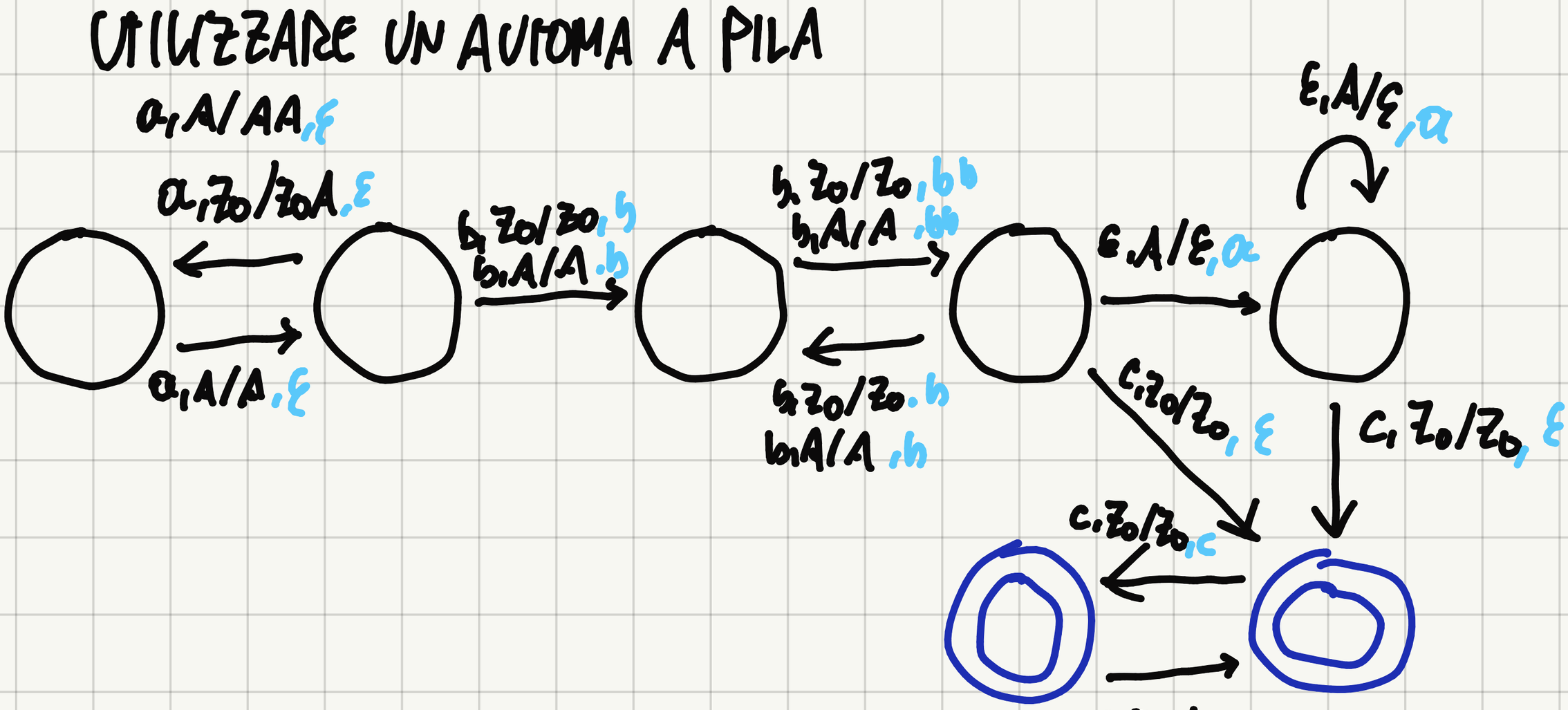
Assuming that the algorithm, after a standard translation, is executed on a RAM machine, evaluate its time and space complexity classes as a function of the input value n, using both the uniform and the logarithmic cost criteria.

1

a È SUFFICIENTE UN AUTOMA A STATI FINITI



b POICHÉ, IN USCITA, SI STAMPANO LE b PRIMA DELLE a, OCCORRE UTILIZZARE UN AUTOMA A PILA



2 INPUT:

- in ARRAY DI DIMENSIONE nIn CON AL PIÙ DUE 0

- nIn > 4

OUTPUT:

- OUT BOOLEANO

PRECONDIZIONI] $nIn > 4 \wedge \exists i,j\ (0 \leq i < nIn \wedge 0 \leq j < nIn \wedge$

$\wedge i \neq j \wedge in[i] = a \wedge in[j] = a \wedge \forall k$

$(0 \leq k < nIn \rightarrow (in[k] = a \leftrightarrow (k = i \vee k = j)))$

POSTCONDIZIONI $(out = T) \leftrightarrow$

$(\exists i, j\ (0 \le i \le n \le n \wedge 0 \le j \le n \le n \wedge$

$\wedge\ i \neq j \wedge in[i] = a \wedge in[j] = a \ \wedge$

$(\exists k\ (i < k < j \rightarrow in[k] = b)) \vee$

$(\forall k\ (i < k < j \rightarrow in[k] = c)) \vee$

$(\forall k\ (i < k < j \rightarrow in[k] = d)))$

a) CORRETTO. ABBIAMO "ALMENO UNA b"

b) CORRETTO. ABBIAMO 3 a, QUINDI LE PRECONDIZIONI NON SONO SODDISFATTE

c) CORRETTO. ABBIAMO "ALMENO UNA b"

d) ERRATO. LA POST-CONDIZIONE NON È SODDISFATTA

3

a L'INSIEME È DECIDIBILE, IN QUANTO INSIEME DI TUTTE LE FUNZIONI COMPUTABILI (RICE THEOREM). E QUINDI SEMIDECIDIBILE. INOLTRE, È SEMPRE DEFINITA UNA FUNZIONE CHE, DATI X E Y, STABILISCE SE Y È L'X-MO NUMERO PRIMO

b L'INSIEME È NON DECIDIBILE, IN QUANTO OCCORREREBBE VALUTARE INFINITI CASI POSSIBILI. PER DEFINIZIONE IRRISOLVIBILE

c L'INSIEME DI FUNZIONI COMPUTABILI CONSIDERATE NON COMPRENDE TUTTE LE FUNZIONI COMPUTABILI → È NON DECIDIBILE.

RISULTA, PERÒ, SEMIDECIDIBILE. APPLICANDO DOVETAILING.

OTTENIAMO SEMPRE UNO STATO ACCETTABILE

(1,1) ✓ → R=1 (2,1) → R=0 (2,2) → R=1...

d NON È ALTRO CHE UNA FORMALIZZAZIONE DEL PUNTO a

4 COSTO UNITARIO:

COMPLESSITÀ TEMPORALE: $T(n) = \tilde{\Theta}(n)$ (DETERMINATA DAL CICLO WHILE)

COMPLESSITÀ SPAZIALE: $S(n) = \tilde{\Theta}(1)$ (NUMERO FISSO DI CELLE DI MEMORIA USATE)

COSTO LOGARITMICO:

COMPLESSITÀ TEMPORALE: $T(n) = \tilde{\Theta}(n \log n)$

COMPLESSITÀ SPAZIALE: $S(n) = \tilde{\Theta}(\log n)$

Theoretical Computer Science Exam, February 7, 2023

The exam consists of **4 exercises**. Available time: 1 hr 30 min.; you may write your answers in Italian, if you wish.

All exercises must be addressed for the written exam to be considered sufficient.

F.NAME …….. L.NAME …… PERSON CODE ……

Exercise 1 (8 pts)

Consider the language

$$L = \{ a^m \; e^j \; b^n \mid n, m > 0 \;\wedge\; m < n \;\wedge\; j \geq 0 \}$$

For instance, strings *aeebbb, aabbb* $\in L$, while *aaeb*, *ebb* $\notin L$.

a) Design an automaton of the least powerful type that accepts language L. Write a computation of the automaton that accepts the string *aeebb* and one that rejects the string *aaebb.*

b) Write a grammar, of the least general type, that generates language L. For this grammar write a derivation of the string *aeebbb*.
c) Design an automaton of the least powerful type that accepts language L^+.
d) Write a grammar, of the least general type, that generates language L^+.

Exercise 2 (8 pts)

Specify, by means of a sentence of Monadic First-Order logic (or Second-Order only if necessary) the language L over the alphabet $\{a, b, c\}$ defined by the following expression

$$L = a\, b^* c$$

Explain why the following strings do not satisfy the specification sentence. Please refer as precisely as possible to the structure of the sentence.

- *abca*
- *bbc*
- *abb*
- *aa*

Consider now the following variant L_1 of language L

$$L_1 = a\, (bb)^* c$$

and say how the specification sentence for language L could be modified to specify L_1.

Exercise 3 (8 pts)

Please answer the following questions providing suitable explanations for your answers.

1) Consider the problem of determining if the language accepted by a (generic) Turing machine is empty.
   a) Is this problem decidable?

   b) Is it semidecidable?
2) Consider next the problem of determining if, given two generic Turing machines M_1 and M_2, there is a string accepted by both of them.
   a) Is this problem decidable?

   b) Is it semidecidable?

Exercise 4 (8 pts)

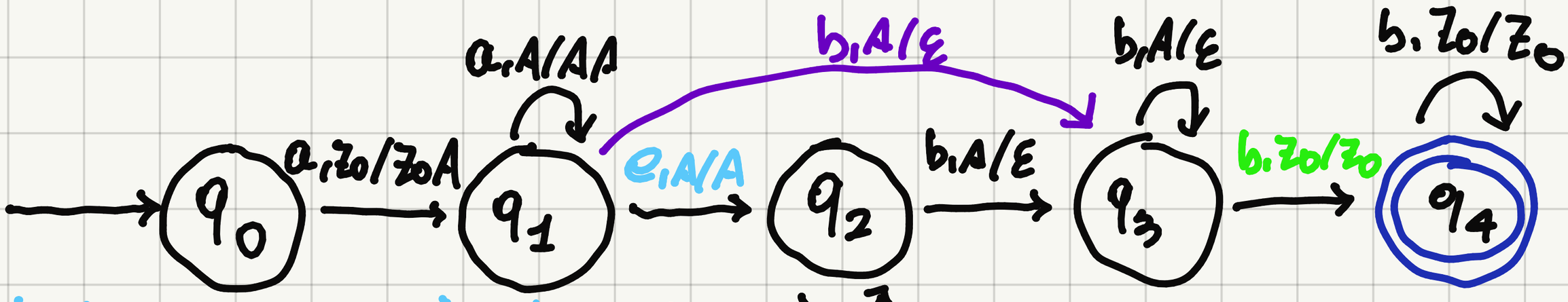
Consider the language $L = \{ w\, w^R \mid w \in \{ a, b \}^+ \}$.

1. Describe a ***deterministic*** k-tape Turing machine M that recognizes language L. You do not need to provide a precise description of M in terms of its transition function, but you should provide an informal description sufficiently clear and detailed so that the time complexity $T_M(n)$ (where $n = |x|$ is the length of the input string) can be estimated. Efficient solutions are preferred and the time complexity should be $\Theta(n)$. Provide an estimation of the time complexity function as precise as possible; notice that you should evaluate the complexity *function*, not only its Θ class.

2. Say if a ***nondeterministic*** k-tape Turing machine N can recognize language L with a time complexity $T_N(n)$ such that $T_N(n) < T_M(n)$ for all n. In the positive case, describe N in a sufficiently clear and detailed way so that the time complexity $T_N(n)$ (where $n = |x|$ is the length of the input string) can be estimated, and estimate $T_N(n)$ as precisely as possible. In the negative case, explain why such a nondeterministic Turing machine does not exist.

1 a

PER IL CONFRONTO m,n OCCORRE UN AUTOMA A PILA



IL NUMERO DI e È IRRILEVANTE CASO $j=0$ $n>m$

$\langle q_0, aeebb, Z_0 \rangle \vdash \langle q_1, eebb, Z_0A \rangle \vdash \langle q_2, ebb, Z_0A \rangle$

$\vdash \langle q_2, bb, Z_0A \rangle \vdash \langle q_3, b, Z_0 \rangle \vdash \langle q_4, \varepsilon, Z_0 \rangle$ ✓

$\langle q_0, aaebb, Z_0 \rangle \vdash \langle q_1, aebb, Z_0A \rangle \vdash \langle q_1, ebb, Z_0AA \rangle$

$\vdash \langle q_2, bb, Z_0AA \rangle \vdash \langle q_3, b, Z_0A \rangle \vdash \langle q_3, \varepsilon, Z_0 \rangle$ CONFUSED UNGRAMMATICAL

b

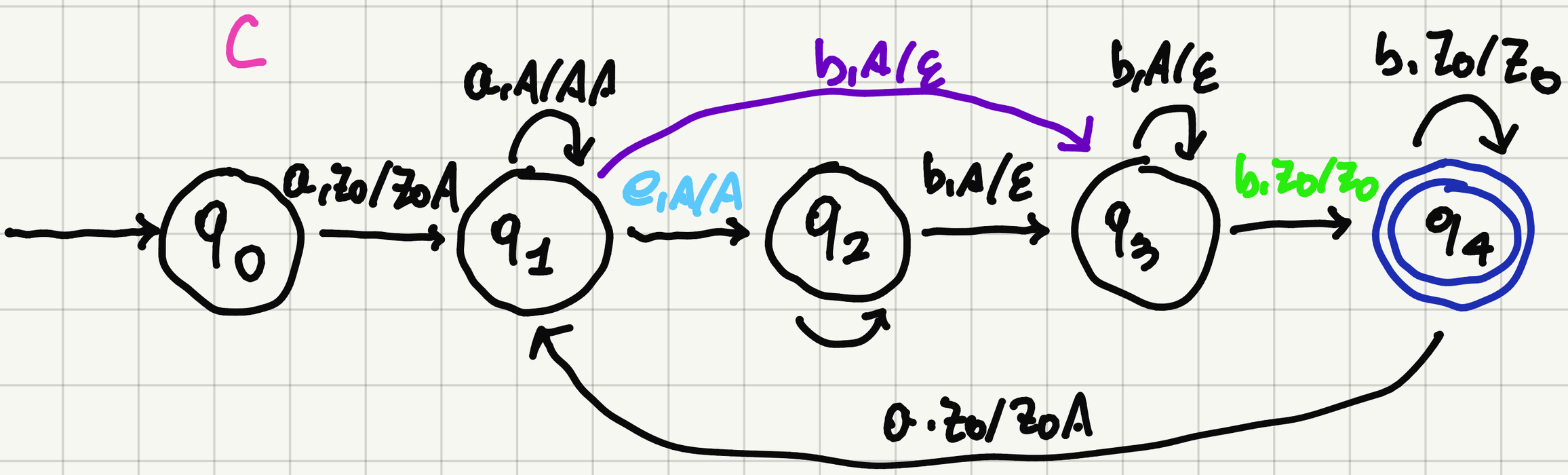
$B \to Bb \mid Sb$

$S \to aSb \mid aEb$

$E \to eE \mid \varepsilon$

$B \to Bb \to Sbb \to aEbbb \to aeEbbb \to aeeEbbbb \to aeebbb$

c



d

$X \to SX \mid S$

$B \to Bb \mid Sb$

$S \to aSb \mid aEb$

$E \to eE \mid \varepsilon$

2 L

$\alpha(0) \wedge ((C(x) \to last(x)) \wedge$

$\exists x (\neg last(x) \to b(x) \wedge$

$b(x) \to b(x+1) \vee C(x+1))$

- abca | NON È RISPETTATA "$C(x) \to last(x)$"
- bbc | NON È RISPETTATA "$\alpha(0)$"
- abb | NON È RISPETTATA "$C(x) \to last(x)$"
- aa | NON È RISPETTATA "$C(x) \to last(x)$"

L_1

INTRODUCO IL PREDICATO P(x) "NUMERO PARI DI b"

$\alpha(0) \wedge P(0) \wedge (C(x) \wedge P(x) \to last(x) \wedge$

$\exists x ((\neg last(x) \wedge P(x) \to b(x+1) \wedge \neg P(x)) \wedge$

$(\neg last(x) \wedge \neg P(x) \to b(x+1) \wedge P(x))$

3 1

a

NO, POICHÈ STIAMO CONSIDERANDO IL SOLO INSIEME DI FUNZIONI COMPUTABILI RELATIVO ALLE M.T. CHE ACCETTANO UN LINGUAGGIO VUOTO. ESSO NON È NE Ø NE L'INSIEME UNIVERSO, PERTANTO (RICE THEOREM) NON È DECIDIBILE

b

IL COMPLEMENTARE "M.T. CHE ACCETTANO ALCUNE STRINGHE" È SEMIDECIDIBILE (DOVETAILING). PERTANTO IL PROBLEMA DATO È NON SEMIDECIDIBILE

2

a

NON È DECIDIBILE, IN QUANTO CONSIDERIAMO LE SOLE MACCHINE DI TURING CHE, DATO UN INPUT x, È ACCETTATO DA ENTRAMBE

b

1) SEMIDECIDIBILE. IN QUANTO È TOTALE E, SOPRATTUTTO, COMPUTABILE. INFATTI, CONSIDERANDO UNA STRINGA, UNA FUNZIONE CHE VERIFICHI SE È RICONOSCIUTA DA M_1 (ANALOGO PER M_2) È SEMPRE REALIZZABILE.

4 a OCCORRE USARE UNA MACCHINA DI TURING A DUE NASTRI.

NEL PRIMO VIENE LETTA L'INTERA STRINGA, NEL SECONDO SOLO LA METÀ. SCORRENDO, POI, ENTRAMBI I NASTRI VERSO SINISTRA, VIENE VERIFICATO SE w^R È EFFETTIVAMENTE IL SIMMETRICO DI w

$$T_M(n) = \underline{n} + c_1 + \underline{\frac{n}{2}} + c_2 + \underline{(\frac{n}{2} + \frac{n}{2})} + c_3 = \frac{5}{2}n + c$$

b UNA MACCHINA DI TURING NON DETERMINISTICA PUÒ LAVORARE CON UN SINGOLO NASTRO. PER OGNI CARATTERE DI w IL NASTRO SI SPOSTA DI UNA POSIZIONE VERSO DESTRA E "INDOVINATO" L'ULTIMO CARATTERE DI w, PER OGNI CARATTERE DI w^R SI SPOSTA DI UNA POSIZIONE A SINISTRA E, PASSO DOPO PASSO, VIENE VERIFICATO SE w^R È EFFETTIVAMENTE IL SIMMETRICO DI w.

È IMMEDIATO NOTARE CHE $T_N(n) = n + c$, E CHE

QUINDI $T_N(n) \in T_M(n)$