Testi di esame precedenti a.a. e soluzioni

1 Problemi

Problema 6.1: Dimostrare che, per ogni costante intera positiva k, 2^{n^k} è una funzione time-constructible. Suggerimento: usare come subroutine la macchina di Turing che dimostra che n^k è una fuzione time-constructible.

Problema 6.2: Utilizzando i risultati noti sulle funzioni limite dimostrare che, per ogni costante intera positiva k e per ogni $(k+1)$-pla $\langle a_0, a_1, \ldots, a_k \rangle$ di costanti intere positive, $f(n) = \sum_{i=0}^{k} a_i n^i$ è una funzione time-constructible.

Problema 6.3: Sia $f(n)$ una funzione time-constructible. Dimostrare che, allora, anche $2^{f(n)}$ è una funzione time-constructible.

Problema 6.4: Dimostrare se la funzione $f : \mathbb{N} \rightarrow \mathbb{N}$ di seguito descritta è time-constructible.

$$f(n) = \begin{cases} \frac{3n}{2} & \text{se } n \text{ è pari} \\ \\ \frac{3(n-1)}{2} & \text{se } n \text{ è dispari} \end{cases}$$

Problema 6.5: Sia $k \in \mathbb{N}$ un valore costante. Dimostrare che le due funzioni $f_k : \mathbb{N} \rightarrow \mathbb{N}$ e $g_k : \mathbb{N} \rightarrow \mathbb{N}$ di seguito descritte sono time-constructible.

$$\begin{aligned} f_k(n) &= \lceil \frac{n}{k} \rceil; \\ g_k(n) &= \lfloor \frac{n}{k} \rfloor. \end{aligned}$$

Problema 6.6: Progettare una macchina di Turing di tipo trasduttore che legga in input un intero n, rappresentato in unario, e, in tempo in $O(n)$, calcoli la funzione

$$f(n) = \left\lfloor \frac{n}{3} \right\rfloor + \left[n - 3 \left\lfloor \frac{n}{3} \right\rfloor \right]$$

scrivendo il risultato (in unario) sul nastro di output (si osservi che il secondo addendo della funzione è il resto della divisione intera di n per 3).
Si può affermare che $f(n)$ è time-constructible?

Problema 6.7: Si ricordi che, per ogni $k \in \mathbb{N}$ costante, la funzione n^k è time-constructible, e sia, per ogni $k \in \mathbb{N}$ costante, T_k il trasduttore deterministico che certifica la time-constructibility di n^k. Utilizzando le macchine T_3 e T_2, dimostrare che esiste una macchina di Turing T di tipo trasduttore che calcola la funzione

$$f(n) = \left\lfloor \frac{n^3+2}{n^2+1} \right\rfloor$$

(parte intera della divisione fra n^3+2 e n^2+1) e tale che $dtime(T,n) \in \mathbf{O}(n^3)$.

2 Soluzioni

Soluzione del problema 6.1

Mostriamo di seguito lo pseudo-codice corrispondente ad una macchina di Turing che calcola2^{n^k} in cui:

- le variabili n1, n2, n3, n4 corrispondono ad altrettanti nastri semi-infiniti della macchina di Turing le cui celle sono numerate a partire dalla posizione 1;
- la variabile i2 corrisponde alla posizione della testina sul nastro n2;
- l'operatore + applicato a variabili di tipo nastro denota la concatenazione dei rispettivi contenuti;
- l'operatore + applicato a variabili di tipo testina denota lo spostamento a destra della stessa;
- l'istruzione calcolaPotenzaK-esima(n1, n2) denota l'utilizzo di un sottoprogramma che scrive sul nastro n2 il valore corrispondente al contenuto del nastro n1 elevato alla potenza k-esima.

n1 $\leftarrow$ n;
calcolaPotenzaK-esima(n1, n2);
n3 $\leftarrow$ 2;
i2 $\leftarrow$ 2;
while (i2 $\leq$ n2) {
    n4 $\leftarrow$ n3;
    n3 $\leftarrow$ n3 + n4;
    i2 $\leftarrow$ i2 + 1;
}
output: n3.

Per dimostrare che 2^{n^k} è una funzione time-constructible occorre ancora mostrare che la macchina di Turing appena descritta opera in tempo O(2^{n^k}). A questo scopo, osserviamo che l'invocazione della routine calcolaPotenzaK-esima(n1, n2) richiede tempo O(n^k) (in quanto n^k è una funzione limite. Rimane da analizzare il costo del ciclo **while**. All'inizio della i-esima iterazione ($i \geq 2$) il nastro n3 contiene il valore 2^{i-1} che viene immediatamente copiato sul nastro n4 e poi concatenato al contenuto del nastro n3 stesso (calcolando così il valore 2^i). Pertanto, l'iterazione i-esima richiede tempo $2^{i-1}+2^i$. Poiché il valore i è compreso fra 2 e n^k, il costo del ciclo **while** è

$$\sum_{i=2}^{n^k}\left[2^{i-1}+2^i\right] = 2^{n^k}-1-1+2^{n^k+1}-1-1-2 < 2^{n^k}+2^{n^k+1} = 2^{n^k}+2\cdot 2^{n^k} = 3\cdot 2^{n^k}.$$

Soluzione del problema 6.2

Si osservi che, essendo k e $a_0,\ldots,a_k$ valori costanti, non abbiamo bisogno di memorizzarli su nastro in quanto essi possono essere codificati direttamente negli stati interni della macchina di Turing che deve calcolare la funzione. In questo modo, ad esempio, con l'istruzione "n2 $\leftarrow a_0$" ci riferiamo in breve alla seguente sequenza di a_0 istruzioni della macchina di Turing:

1. se nello stato $q_{a_0,0}$ la testina sul nastro n2 legge un blank allora scrive un 1 e si sposta a destra e la macchina entra nello stato $q_{a_0,1}$;
2. se nello stato $q_{a_0,1}$ la testina sul nastro n2 legge un blank allora scrive un 1 e si sposta a destra e la macchina entra nello stato $q_{a_0,2}$;
3. ...

4. se nello stato q_{a_0,a_0-1} la testina sul nastro n2 legge un blank allora scrive un 1 e si sposta a destra e la macchina entra nello stato q_{a_0,a_0}.

Con questa convenzione, definiamo una macchina di Turing in cui:

- le variabili n1, n2, n3 corrispondono ad altrettanti nastri semi-infiniti della macchina di Turing le cui celle sono numerate a partire dalla posizione 1;
- la variabile i corrisponde all'insieme di stati necessari a calcolare il monomio $a_i n^i$; l'istruzione "i $\leftarrow$ i +1" indica il passaggio al calcolo del monomio $a_{i+1}n^{i+1}$ (o, a basso livello, il passaggio dallo stato q_{a_i,a_i} allo stato $q_{a_{i+1},0}$);
- la variabile j numera gli stati $q_{a_i,0},\ldots q_{a_i,a_i}$; l'istruzione "j $\leftarrow$ j +1" indica il passaggio della macchina dallo stato $q_{a_i,j}$ allo stato $q_{a_i,j+1}$;
- l'operatore + applicato a variabili di tipo nastro denota la concatenazione dei rispettivi contenuti;
- l'istruzione calcolaPotenza(n1, n3, i) denota l'utilizzo di un sottoprogramma che scrive sul nastro n3 il valore corrispondente al contenuto del nastro n1 elevato alla potenza i-esima.

Possiamo ora mostrare lo pseudo-codice corrispondente ad una macchina di Turing che calcola $f(n) = \sum_{i=0}^{k} a_i n^i$

```
n1 ← n;
n2 ← a0;
i ← 1;
while (i ≤ k) {
    calcolaPotenza(n1, n3, i);
    j ← 1 ;
    while (j ≤ ai) {
        n2 ← n2 + n3;
        j ← j +1;
    }
    i ← i +1;
}
output: n2.
```

Per dimostrare che $f(n) = \sum_{i=0}^{k} a_i n^i$ è una funzione time-constructible, osserviamo che ciascuna invocazione "calcolaPotenza(n1, n3, i)" richiede tempo $p_i n^i$, per una opportunacostante p_i, e che il ciclo **while** interno (la cui unica operazione che richiede tempo di calcolo è la concatenazione del contenuto del nastro n3 al contenuto del nastro n2) richiede tempo $a_i n^i$. Pertanto, il costo del ciclo **while** esterno richiede tempo

$$\sum_{i=1}^{k}(p_i + a_i)n^i \leq n^k \sum_{i=1}^{k}(p_i + a_i) = cn^k$$

dove il valore c è costante. Poiché le operazioni di inizializzazione che precedono il ciclo **while** esterno richiedono tempo costante o lineare in n, questo dimostra che $f(n)$ è una funzione time-constructible.

Soluzione del problema 6.3

Consideriamo una macchina di Turing T a 4 nastri: il nastro N_1 è utilizzato per registrare l'input n, il nastro N_2 per registrare il valore $f(n)$, il nastro N_3 è un nastro di lavoro ed il nastro N_4 è utilizzato per registrare l'output.

Ricordando quanto fatto nella dispensa 5, per il calcolo della funzione $2^{f(n)}$ descriviamo un algoritmo codificato in un qualche linguaggio ad alto livello: indichiamo con una variabile i la posizione della testina sul nastro N_2, con una

Precondizioni:	sul nastro N_1 è scritto il valore n codificato in unario
1	$n_2 \leftarrow f(n_1)$;
2	$n_4 \leftarrow 1$;
3	$i \leftarrow 1$;
4	**while** $(i \leq n_2)$ **do begin**
5	$\quad n_3 \leftarrow n_4$;
6	$\quad n_4 \leftarrow n_4 \oplus n_3$;
7	$\quad i \leftarrow i+1$;
8	**end**

Tabella 6.1: Algoritmo corrispondente alla macchina di Turing T che calcola $2^{f(n)}$.

variabile n_j il contenuto del nastro N_j $(j = 1,2,3,4)$, con l'istruzione "$n_u \leftarrow n_v$;" la copia del contenuto del nastro N_v sul nastro N_u , e con "$\oplus$" l'operatore di concatenazione di due stringhe; l'algoritmo è descritto in Tabella 6.1.

Resta da calcolare il numero di passi eseguiti da T. L'istruzione 1 richiede $O(f(n))$ passi poiché, per definizione, $f(n)$ è una funzione limite. L'istruzione 2 scrive il valore 1 sul nastro N_4, in un numero costante di passi, e l'istruzione 3 posiziona la testina del nastro N_2 sul primo carattere che esso contiene. Quest'ultima operazione richiede $O(f(n))$ passi. Il ciclo **while** viene ripetuto $f(n)$ volte; in ciascuna iterazione, il contenuto del nastro N_4 viene copiato sul nastro N_3 (in n_4 passi) e poi il contenuto del nastro N_3 viene concatenato al contenuto del nastro N_4 (in n_3 passi), riposizionano, infine, la testina del nastro N_4 sul suo primo carattere (in $2n_3$ passi) e la testina del nastro N_2 avanza di una posizione. In definitiva, osservando che nel corso della j-esima iterazione del ciclo **while** viene calcolato il valore 2^j (o, equivalentemente, che all'ingresso del ciclo $n_4 = 2^{j-1}$), l'algoritmo proposto richiede un numero di passi pari a

$$
\begin{aligned}
O(f(n)) + \sum_{1 \leq i \leq f(n)} \left[4 \cdot 2^{i-1} + 1\right] &= O(f(n)) + \sum_{1 \leq i \leq f(n)} \left[2 \cdot 2^i + 1\right] \\
&= O(f(n)) + f(n) + \sum_{1 \leq i \leq f(n)} 2 \cdot 2^i \;=\; O(f(n)) + 2 \sum_{1 \leq i \leq f(n)} 2^i \\
&= O(f(n)) + 2\left[\frac{2^{f(n)+1}-1}{2-1} - 1\right] \;=\; O(f(n)) + 2\left[2^{f(n)+1} - 2\right] \\
&= O(f(n)) + 4 \cdot 2^{f(n)} - 4 \;=\; O(2^{f(n)})
\end{aligned}
$$

e questo dimostra che $2^{f(n)}$ è una funzione limite.

Soluzione del problema 6.4

Definiamo una macchina di Turing T a 3 nastri (a testine indipendenti) che opera in quattro fasi distinte come di seguito descritto:

- il nastro N_1 contiene l'input, ivi memorizzato in unario all'inizio della computazione, preceduto e seguito da $\square$;
- il nastro N_2 è il nastro di lavoro, inizialmente vuoto, sul quale, al termine della prima fase, si troverà il valore $n/2$, se n è pari, o $(n-1)/2$ se n è dispari;
- il nastro N_3 è il nastro di output, inizialmente vuoto, sul quale, al termine della fase k si trova scritto il valore $(k-1)n$, per $k = 2,3,4$.

Durante la prima fase, in cui viene scritto sul nastro N_2 il valore $n/2$ oppure il valore $(n-1)/2$, sono utilizzati gli stati q_0 (stato iniziale) e q_1^1 e q_0^1: il primo (oltre ad essere stato iniziale) indica che sul nastro N_1 è stato letto un numero pari di 1, il secondo che è stato letto un numero dispari di 1. Il valore $n/2$ o $(n-1)/2$ viene calcolato contestualmente al controllo di parità di n: ogni volta che viene letta una coppia di 1 sul nastro N_1 viene scritto un (singolo) 1 sul nastro

N_2; così, al termine della scansione dell'input, se sono stati letti un numero dispari di 1 e viene letto un $\square$ allora nulla viene scritto sul nastro N_2. Formalmente, le quintuple utilizzate nella prima fase sono:

$$\langle q_0, (1, \square, \square), (1, \square, \square), q_1^1, (d, f, f) \rangle \qquad \langle q_0, (\square, \square, \square), (\square, \square, \square), q^2, (f, f, f) \rangle$$

$$\langle q_1^1, (1, \square, \square), (1, 1, \square), q_0, (d, d, f) \rangle \qquad \langle q_1^1, (\square, \square, \square), (\square, \square, \square), q^2, (f, s, f) \rangle,$$

dove q^2 è lo stato iniziale della fase 2. Osserviamo che, al termine della prima fase, la testina del nastro N_2 è posizionata sul carattere 1 più a destra ivi contenuto.
La seconda fase scrive il valore $n/2$ oppure $(n-1)/2$ sul nastro N_3: questo corrisponde a copiare il contenuto del nastro N_2 sul nastro N_3 (con la testina del nastro N_2 che si muove da destra a sinistra e la testina del nastro N_3 che si muove da sinistra a destra). Allo scopo, è sufficiente utilizzare lo stato q^2:

$$\langle q^2, (\square, 1, \square), (\square, 1, 1), q^2, (f, s, d) \rangle \qquad \langle q^2, (\square, \square, \square), (\square, \square, \square), q^3, (f, d, f) \rangle,$$

dove q^3 è lo stato iniziale della fase 3. Osserviamo che, al termine della seconda fase, la testina del nastro N_2 è posizionata sul carattere 1 più a sinistra ivi contenuto.
La terza fase somma il valore $n/2$ oppure $(n-1)/2$ al contenuto del nastro N_3: questo corrisponde ad una concatenazione dei contenuti dei due nastri, ossia, a copiare il contenuto del nastro N_2 sul nastro N_3 a partire dal carattere 1 più a sinistra su quest'ultimo contenuto. In questa fase, le testine del nastro N_2 e del nastro N_3 si muovono entrambe da sinistra a destra. Allo scopo, è sufficiente utilizzare lo stato q^3:

$$\langle q^3, (\square, 1, \square), (\square, 1, 1), q^3, (f, d, d) \rangle \qquad \langle q^3, (\square, \square, \square), (\square, \square, \square), q^4, (f, s, f) \rangle,$$

dove q^4 è lo stato iniziale della fase 4. Osserviamo che, al termine della seconda fase, la testina del nastro N_2 è posizionata sul carattere 1 più a destra ivi contenuto.
La quarta fase concatena il contenuto del nastro N_2 al contenuto del nastro N_3 ed è del tutto simile alla fase 2:

$$\langle q^4, (\square, 1, \square), (\square, 1, 1), q^4, (f, s, d) \rangle \qquad \langle q^4, (\square, \square, \square), (\square, \square, \square), q_F, (f, f, f) \rangle,$$

dove q_F è lo stato finale.

Per dimostrare che $f(n)$ è una funzione time-constructible occorre ancora mostrare che la macchina di Turing appena descritta opera in tempo $O(f(n))$. A questo scopo, osserviamo che la prima fase richiede un numero di passi pari alla lunghezza del contenuto del nastro N_1 più 1, mentre ciascuna delle altre fasi richiede un numero di passi pari alla lunghezza del contenuto del nastro N_2 più 1. Quindi, il numero di passi totale è

$$t(n) = \begin{cases} n + 1 + 3\left(\frac{n}{2} + 1\right) & \text{se } n \text{ è pari} \\ n + 1 + 3\left(\frac{n-1}{2} + 1\right) & \text{se } n \text{ è dispari,} \end{cases}$$

ossia, $t(n) \in O(f(n))$.

Soluzione del problema 6.5

Problema 1. Definiamo una macchina di Turing T a 3 nastri (a testine indipendenti) che calcola simultaneamente $\lceil \frac{n}{k} \rceil$ e $\lfloor \frac{n}{k} \rfloor$ come di seguito descritto:

- il nastro N_1 contiene l'input, ivi memorizzato in unario all'inizio della computazione, preceduto e seguito da $\square$;
- il nastro N_2 è il nastro di lavoro e di output per la funzione $\lceil \frac{n}{k} \rceil$, inizialmente vuoto, sul quale, al termine della computazione, si troverà il valore $\lceil \frac{n}{k} \rceil$;
- il nastro N_3 è il nastro di lavoro e di output per la funzione $\lfloor \frac{n}{k} \rfloor$, inizialmente vuoto, sul quale, al termine della computazione, si troverà il valore $\lfloor \frac{n}{k} \rfloor$.

T utilizza gli stati q_0 (stato iniziale), q_1, q_2, … $q_{k-1}k$ (si ricordi che k è una costante) e lo stato finale q_F: q_0 (oltre ad essere stato iniziale) indica che sul nastro N_1 è stato letto un numero di 1 pari ad un multiplo di k, q_1 indica che sul nastro N_1 è stato letto un numero di 1 pari ad un multiplo di k più un ulteriore 1, e, in generale, q_i ($i < k$) indica che sul nastro N_1 è stato letto un numero di 1 pari ad un multiplo di k più ulteriori i 1.
Il valore $\lceil \frac{n}{k} \rceil$ viene calcolato scrivendo un 1 sul nastro N_2 ogni volta che, nello stato q_0, viene letto un 1 sul nastro N_1: questo significa che n è almeno un multiplo di k (perché la macchina è nello stato q_0) più 1 (perché viene letto un 1). Così, se $n = hk + m$, con $m < k$, al termine della scansione dell'input, risultano scritti sul nastro N_2 h 1 più un eventuale ulteriore 1 se $m > 0$.
Analogamente, il valore $\lfloor \frac{n}{k} \rfloor$ viene calcolato scrivendo un 1 sul nastro N_2 ogni volta che, nello stato q_{k-1}, viene letto un 1 sul nastro N_1: questo significa che n è almeno un multiplo di k (perché la macchina è nello stato q_{k-1} e viene letto un 1). Così, se $n = hk + m$, con $m < k$, al termine della scansione dell'input, risultano scritti sul nastro N_3 h 1.

Formalmente, le quintuple utilizzate sono:

$$
\begin{array}{ll}
\langle q_0, (1, \square, \square), (1, 1, \square), q_1, (d, d, f) \rangle & \\
\langle q_i, (1, \square, \square), (1, \square, \square), q_{i+1}i, (d, f, f) \rangle & \forall 1 \leq i \leq k-2 \\
\langle q_{k-1}, (1, \square, \square), (1, \square, 1), q_0, (d, d, f) \rangle & \\
\langle q_i, (\square, \square, \square), (\square, \square, \square), q_F, (f, f, f) \rangle, & \forall 1 \leq i \leq k-1.
\end{array}
$$

Per dimostrare che $f_k(n)$ e $g_k(n)$ sono funzioni time-constructible occorre ancora mostrare che la macchina di Turing appena descritta opera in tempo O(n)=O($f_k(n)$)=O($g_k(n)$) (in quanto k è una costante). A questo scopo, è sufficiente osservare che la computazione descritta richiede un numero di passi pari alla lunghezza del contenuto del nastro N_1. Quindi, il numero di passi totale è $t(n) = n$.

Soluzione del problema 6.6

Definiamo una macchina di Turing T a due nastri: il nastro di lavoro N_1 (sul quale è inizialmente scritto l'input n codificato in unario) e il nastro di output N_2 (inizialmente vuoto).
T utilizza quattro stati interni: lo stato iniziale q_0, gli stati intermedi q_1 e q_2, e lo stato finale q_F.
La macchina opera in due fasi: durante la prima fase calcola $\lfloor \frac{n}{3} \rfloor$, durante la seconda fase calcola il resto della divisione di n per 3.
Durante la prima fase, gli stati q_0, q_1 e q_2 vengono utilizzati per contare quanti '1' sono stati letti successivamente alla scrittura dell'ultimo '1' sul nastro N_2: solo quando la macchina si trova nello stato q_2 e legge un '1', poiché sono stati letti 3 '1' consecutivi, viene scritto un '1' sul nastro N_2. La prima fase termina, e inizia la seconda fase, quando viene letto un $\square$ sul nastro N_1:

- se questo avviene quando la macchina è nello stato q_0, allora non è stato letto alcun '1' successivamente all'ultima scrittura, ossia, n è un multiplo di 3, ossia, il resto della divisione è 0 e la computazione può terminare;
- se questo avviene quando la macchina è nello stato q_1, allora è stato letto un solo '1' successivamente all'ultima scrittura, ossia, il resto della divisione di n per 3 è 1 e la macchina deve stampare un solo '1' e poi terminare la computazione;
- se questo avviene quando la macchina è nello stato q_2, allora sono stati letti due '1' successivamente all'ultima scrittura, ossia, il resto della divisione di n per 3 è 2 e la macchina deve stampare due '1' prima di terminare la computazione.

Più in dettaglio, descriviamo le quintuple che implementano il comportamento sopra illustrato, cominciando dalla fase 1. Inizialmente, nessun '1' è stato letto e la macchina è nello stato q_0; se, nello stato q_0 viene letto un '1' su $N - 1$ allora la macchina entra nello stato q_1; se è stato già letto un solo '1' (e quindi la macchina è nello stato q_1) e viene letto un '1' su $N - 1$ allora la macchina entra nello stato q_2; se sono stati già letti due '1' (e quindi la macchina è nello stato q_2) e viene letto un '1' su $N - 1$ allora la macchina scrive un '1' su N_2 e torna nello stato q_0:

$$\langle q_0, (1, \square), (1, \square), q_1, (\mathit{destra}, \mathit{ferma}) \rangle \quad \langle q_1, (1, \square), (1, \square), q_2, (\mathit{destra}, \mathit{ferma}) \rangle \quad \langle q_2, (1, \square), (1, 1), q_0, (\mathit{destra}, \mathit{destra}) \rangle.$$

Non appena viene letto un $\square$ ha inizio la fase due, che è implementata dalle quintuple seguenti:

$$\langle q_0, (\square, \square), (\square, \square), q_F, (\mathit{ferma}, \mathit{ferma})\rangle \quad \langle q_1, (\square, \square), (\square, 1), q_F, (\mathit{ferma}, \mathit{ferma})\rangle \quad \langle q_2, (\square, \square), (\square, 1), q_1, (\mathit{ferma}, \mathit{destra})\rangle.$$

Poiché durante la prima fase la macchina T muove sempre la testina sul nastro N_1 a destra e la fase 1 termina appena sul nastro N_1 viene letto $\square$, la fase 1 termina in n passi. La fase 2, inoltre, richiede non più di 2 passi. In conclusione, T calcola $f(n)$ in $p \leq n+2 \leq 2n$ passi.

Ricordando che una funzione è time-constructible se esiste una macchina di Turing che la calcola in un numero di passi proporzionale al suo valore, questo ci permette di affermare che f è time-constructible.

Soluzione del problema 6.7

La macchina T che testimonia la time-constructibility della funzione $f(n)$ utilizza 5 nastri:

- sul nastro N_1 è scritto l'input n, in unario;
- sul nastro N_2 verrà scritto il valore n^2 e, immediatamente dopo, il valore n^2+1;
- sul nastro N_3 verrà scritto il valore n^3 e, immediatamente dopo, il valore n^3+2;;
- sul nastro N_4 verrà scritto il valore di output $f(n)$;
- il nastro N_5 è il nastro di lavoro.

La computazione $T(n)$ è suddivisa in 5 fasi, descritte di seguito.

Fase 1) T simula il comportamento della macchina T_2 e, utilizzando il nastro N_5 come nastro di lavoro, scrive il valore n^2 (in unario) sul nastro N_2.

Fase 2) T scrive un carattere '1' a destra dell'ultimo carattere '1' scritto, nella fase precedente, sul nastro N_2: al termine di questa fase, sul nastro N_2 è scritto il valore n^2+1 (in unario).

Fase 3) T simula il comportamento della macchina T_3 e, utilizzando il nastro N_5 come nastro di lavoro, scrive il valore n^3 (in unario) sul nastro N_3.

Fase 4) T scrive due caratteri '1' a destra dell'ultimo carattere '1' scritto, nella fase precedente, sul nastro N_3: al termine di questa fase, sul nastro N_3 è scritto il valore n^3+2 (in unario).

Fase 5) Utilizzando i contenuti dei nastri N_2 e N_3, T calcola il valore $f(n)$ verificando quante volte il valore scritto sul nastro N_2 è contenuto nel valore scritto sul nastro N_3. All'inizio della Fase 5, le testine sui nastri N_2 e N_3 sono posizionate sui caratteri '1' più a destra su ciascun nastro; dunque la computazione prosegue alternando fra le due sottofasi di seguito descritte:

5.1) se le testine sui nastri N_2 e N_3 leggono entrambe '1', esse vengono spostate di una posizione a sinistra; se la testina sul nastro N_2 legge '$\square$' e la testina sul nastro N_3 legge '1', allora la macchina scrive '1' sul nastro N_4, sposta a destra la testina sul nastro N_2 (lasciando ferma la testina sul nastro N_3) e passa ad eseguire la sottofase 5.2); se le testine sui nastri N_2 e N_3 leggono entrambe '$\square$', allora la macchina scrive '1' sul nastro N_4 e termina; se la testina sul nastro N_2 legge '1' e la testina sul nastro N_3 legge '$\square$', allora la macchina termina (senza scrivere '1' sul nastro N_4);

5.1) se le testine sui nastri N_2 e N_3 leggono entrambe '1', la testina sul nastro N_2 viene spostata di una posizione a destra e la testina sul nastro N_3 viene spostata di una posizione a sinistra; se la testina sul nastro N_2 legge '$\square$' e la testina sul nastro N_3 legge '1', allora la macchina scrive '1' sul nastro N_4, sposta a sinistra la testina sul nastro N_2 (lasciando ferma la testina sul nastro N_3) e passa ad eseguire la sottofase 5.1); se le testine sui nastri N_2 e N_3 leggono entrambe '$\square$', allora la macchina scrive '1' sul nastro N_4 e termina; se la testina sul nastro N_2 legge '1' e la testina sul nastro N_3 legge '$\square$', allora la macchina termina (senza scrivere '1' sul nastro N_4).

Valutiamo, ora, $dtime(T,n)$. Dalla definizione delle macchine T_2 e T_3, segue che le fasi 1 e 3 terminano, rispettivamente, in $\mathbf{O}(n^2)$ e $\mathbf{O}(n^3)$ passi; pertanto, poiché le fasi 2 e 4 richiedono tempo costante, la fase 5 ha inizio dopo $\mathbf{O}(n^3)$ passi. Per quanto riguarda la fase 5, osserviamo che essa termina non appena viene letto '$\square$' sul nastro N_3; poiché durante la fase 5 la testina sul nastro N_3 non inverte mai la direzione del proprio movimento (si vuove sempre verso destra), e poiché essa rimane ferma per una istruzione ogni volta che la testina sul nastro N_2 legge '$\square$', ossia, al più $f(n)$ istruzioni, il simbolo '$\square$' sul nastro N_3 viene incontrato dopo aver eseguito al più $\mathbf{O}(n^3) + f(n)$ istruzioni. Quindi, osservando che $f(n) \in \mathbf{O}(n^3)$, possiamo concludere che $dtime(T,n) \in \mathbf{O}(n^3)$.