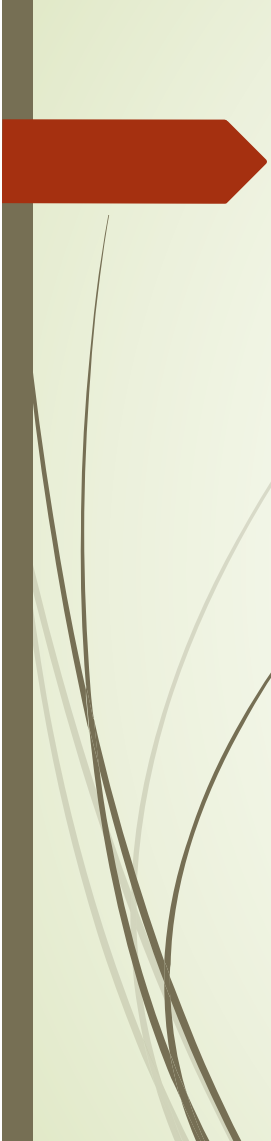




Lezione 23 – funzioni time- e space-constructible e specifiche classi di complessità

Lezione del 30/04/2025



Un paio di questioncine aperte...

- C'erano un paio di cose che erano rimaste lì, un po' in sospeso...
- Diciamo, non del tutto chiuse
- Innanzi tutto, c'era la questione della definizione delle classi di complessità non deterministiche – dove viene richiesta la **accettabilità** di un linguaggio
 - pur sapendo che, ogni volta che fissiamo la quantità massima di risorse (spazio o tempo) utilizzabile, un linguaggio accettabile è anche decidibile
 - non conosciamo la quantità di risorse che occorrono per rigettare le parole che non vi appartengono
- Poi, sappiamo che tutto ciò che è deciso da una macchina non deterministica può essere deciso anche da una macchina deterministica
- Tuttavia, un linguaggio che sappiamo appartenere a $\text{NTIME}[f(n)]$ non sappiamo ancora in quale classe di complessità temporale deterministica collocarlo
 - né sappiamo se il fatto di sapere che appartiene a $\text{NTIME}[f(n)]$ ci fornisca strumenti in grado di affermare “ok, allora sta pure in $\text{DTIME}[\text{qualche altra funzione}]$ ”

La prima questioncina aperta

- Innanzi tutto, non è proprio piacevole dover ammettere che se un certo linguaggio L è in $\text{NTIME}[f(n)]$
 - ossia, sappiamo che esiste una macchina NT che accetta le sue parole x (ossia, le parole $x \in L$) eseguendo $O(f(|x|))$ istruzioni
- non sappiamo quanto tempo occorre per capire che una parola non appartiene a quel linguaggio
 - ossia, quando $x \notin L$ non sappiamo quante istruzioni sono eseguite da ciascuna computazione deterministica di $\text{NT}(x)$ – che, sappiamo, rigetta
- Ebbene, il prossimo teorema afferma che:
 - se f è time-constructible e L è in $\text{NTIME}[f(n)]$, allora una modifica della macchina NT che accetta le parole x di L eseguendo $O(f(|x|))$ istruzioni **è anche capace di rigettare le parole non in L eseguendo $O(f(|x|))$ istruzioni**;
 - se f è space-constructible e L è in $\text{NSPACE}[f(n)]$, allora una modifica della macchina NT che accetta le parole di L utilizzando $O(f(|x|))$ celle del nastro è anche capace di rigettare le parole non in L utilizzando $O(f(|x|))$ celle del nastro;

La prima questioncina aperta

- **Teorema 6.16:** Sia $f : \mathbb{N} \rightarrow \mathbb{N}$ una funzione time-constructible. Se $L \in \text{NTIME}[f(n)]$, allora L è decidibile in tempo non deterministico in $O(f(n))$.

Sia $f : \mathbb{N} \rightarrow \mathbb{N}$ una funzione space-constructible. Se $L \in \text{NSPACE}[f(n)]$, allora L è decidibile in spazio non deterministico in $O(f(n))$

- Dimostriamo soltanto il caso in cui f è time-constructible
- La dimostrazione del caso in cui f è space-constructible è analoga
- Riutilizziamo, aggiustandola opportunamente, la dimostrazione del Teorema 6.2
- **Teorema 6.2** (tempo): Sia $f : \mathbb{N} \rightarrow \mathbb{N}$ una funzione totale calcolabile. Se $L \subseteq \Sigma^*$ è accettato da una macchina di Turing non deterministica NT tale che, per ogni $x \in L$, $\text{ntime}(NT, x) \leq f(|x|)$ allora L è decidibile.
- Lo vedete quanto si assomigliano i due teoremi?

La prima questioncina aperta

- Sia $f : \mathbb{N} \rightarrow \mathbb{N}$ una funzione time-constructible. Se $L \in \text{NTIME}[f(n)]$, allora L è decidibile in tempo non deterministico in $O(f(n))$.
- $L \in \text{NTIME}[f(n)]$: sia NT la macchina che accetta L , e assumiamo che, per $x \in L$, $\text{ntime}(NT, x) \leq c f(|x|)$, per qualche costante $c > 0$
- Poiché f è time-constructible, anche cf è time-constructible: allora, esiste una macchina T_f di tipo trasduttore tale che, per ogni $n \in \mathbb{N}$, $T_f(1^n)$ termina
 - con il valore $c f(n)$ scritto sul nastro di output in unario
 - dopo aver eseguito $O(cf(n))$ istruzioni
- Costruiamo una nuova macchina non deterministica NT' , a tre nastri, che decide L : per ogni $x \in \Sigma^*$
 - $NT'(x)$ scrive $|x|$ in unario sul secondo nastro e invoca $T_f(|x|)$: al termine della computazione sul terzo nastro si troverà scritto $c f(|x|)$ in unario
 - $NT'(x)$ invoca $NT(x)$ e, per ogni quintupla eseguita non deterministicamente da $NT(x)$:
 - se il terzo nastro contiene un '1' allora NT' lo "cancella" e, inoltre,
 - se $NT(x)$ accetta allora anche $NT'(x)$ accetta, se $NT(x)$ rigetta allora anche $NT'(x)$ rigetta;
 - se il terzo nastro di NT' è vuoto (e $NT(x)$ non ha ancora terminato), allora $NT'(x)$ rigetta

La prima questioncina aperta

- Sia $f : \mathbb{N} \rightarrow \mathbb{N}$ una funzione time-constructible. Allora, per ogni $L \in \text{NTIME}[f(n)]$, si ha che L è decidibile in tempo non deterministico in $O(f(n))$.
- Osserviamo, intanto, che le computazioni di NT' terminano sempre
 - se la simulazione di una computazione di $\text{NT}(x)$ dura più di $c f(|x|)$ passi, la interrompiamo!
- Poi, NT' decide L , infatti:
 - se $x \in L$, allora $\text{NT}(x)$ accetta in al più $c f(|x|)$ passi: e, quindi, $\text{NT}'(x)$ accetta
 - se $x \notin L$, allora o $\text{NT}(x)$ rigetta in al più $c f(|x|)$ passi e, quindi, $\text{NT}'(x)$ rigetta, oppure $\text{NT}(x)$ non termina entro $c f(|x|)$ passi e, quindi, $\text{NT}'(x)$, ugualmente, rigetta
- Ma quanto impiega NT' a decidere se $x \in L$ oppure no?
 - $O(c f(|x|))$ per calcolare $c f(|x|)$ – perché $c f$ è time-constructible!
 - e altri $c f(|x|)$ passi per simulare $c f(|x|)$ passi di $\text{NT}(x)$
 - ossia, $O(f(|x|))$ passi
- Per questo possiamo concludere che L è decidibile, in tempo non deterministico $O(f(n))$

La seconda questioncina aperta

- Le uniche relazioni che conosciamo (fino ad ora) fra classi deterministiche e classi non deterministiche sono quelle banali:
 $DTIME[f(n)] \subseteq NTIME[f(n)]$ e $DSPACE[f(n)] \subseteq NSPACE[f(n)]$.
 - basate sull'osservazione che una macchina deterministica è una particolare macchina non deterministica
- A parte ciò, sappiamo che tutto ciò che è deciso da una macchina non deterministica può essere deciso anche da una macchina deterministica
- Tuttavia, un linguaggio che sappiamo appartenere a $NTIME[f(n)]$ non sappiamo in quale classe di complessità temporale deterministica collocarlo
 - non sappiamo se esiste un funzione $g(n)$
 - che magari cresce molto più velocemente di $f(n)$
 - tale che possiamo affermare "se L appartiene a $NTIME[f(n)]$ allora L appartiene a $DTIME[g(n)]$ "
- a meno che la funzione limite f della classe non sia una funzione time-constructible...

La seconda questioncina aperta

- **Teorema 6.17:** Per ogni funzione *time-constructible* $f : \mathbb{N} \rightarrow \mathbb{N}$,
 $\text{NTIME}[f(n)] \subseteq \text{DTIME}[2^{O(f(n))}]$.
- Sia $L \subseteq \{0,1\}^*$ tale che $L \in \text{NTIME}[f(n)]$; allora esistono
 - una macchina di Turing non deterministica NT che accetta L
 - una costante h
- tali che, per ogni $x \in L$, $\text{ntime}(\text{NT}, x) \leq hf(|x|)$.
- Poiché hf è *time-constructible*, esiste T_f che, con input 1^n , calcola $1^{hf(n)}$ in tempo $O(f(n))$.
- Indichiamo con k il grado di non determinismo di NT
 - e ricordiamo che k è una costante, indipendente dall'input
- e utilizziamo di nuovo la tecnica della simulazione per definire una macchina di Turing deterministica T , dotata di 3 nastri, che simuli il comportamento di NT:
 - su input x , T simula in successione, una dopo l'altra, tutte le computazioni deterministiche di NT (x) di lunghezza $hf(|x|)$.

La seconda questioncina aperta

- **Teorema 6.17:** Per ogni funzione *time-constructible* $f : \mathbb{N} \rightarrow \mathbb{N}$,
 $\text{NTIME}[f(n)] \subseteq \text{DTIME}[2^{O(f(n))}]$.
- La macchina T con input x opera in due fasi, come di seguito descritto:
- FASE 1) Simula la computazione $T_f(|x|)$:
 - per ogni carattere di x , scrive sul secondo nastro un carattere '1' - ossia, scrive $1^{|x|}$ sul secondo nastro
 - in seguito, calcola $1^{f(|x|)}$ scrivendolo sul terzo nastro
 - infine, concatena h volte il contenuto del terzo nastro ottenendo il valore $1^{hf(|x|)}$
 - (stiamo dimostrando che: se f è *time-constructible* allora anche hf è *time constructible*
 - cosa che nel teorema precedente avevamo solo enunciato).
- Fase 2) Simula, una alla volta, tutte le computazioni deterministiche di $\text{NT}(x)$ di lunghezza $hf(|x|)$ utilizzando, per ciascuna computazione, la posizione della testina sul terzo nastro come contatore:
 - *to be continued ...*

La seconda questioncina aperta

- **Teorema 6.17:** Per ogni funzione *time-constructible* $f : \mathbb{N} \rightarrow \mathbb{N}$,
 $\text{NTIME}[f(n)] \subseteq \text{DTIME}[2^{O(f(n))}]$.
- La macchina T con input x opera in due fasi, come di seguito descritto:
- Fase 2) Simula, una alla volta, tutte le computazioni deterministiche di $\text{NT}(x)$ di lunghezza $h f(|x|)$ utilizzando, per ciascuna computazione, la posizione della testina sul terzo nastro come contatore:
 - simula al più $h f(|x|)$ passi della computazione più a sinistra di tutte nell'albero $\text{NT}(x)$: se tale computazione accetta entro $h f(|x|)$ passi allora T termina in q_A , altrimenti
 - simula al più $h f(|x|)$ passi della computazione immediatamente più a destra di quella appena simulata: se tale computazione accetta entro $h f(|x|)$ passi allora T termina in q_A , altrimenti
 - ...
 - simula al più $h f(|x|)$ passi della computazione più a destra di tutte nell'albero $\text{NT}(x)$: se tale computazione accetta entro $h f(|x|)$ passi allora T termina in q_A , altrimenti T termina in q_R
- T decide L : infatti... *to be continued* ...

La seconda questioncina aperta

- **Teorema 6.17:** Per ogni funzione *time-constructible* $f : \mathbb{N} \rightarrow \mathbb{N}$,
 $\text{NTIME}[f(n)] \subseteq \text{DTIME}[2^{O(f(n))}]$.
- T decide L: infatti, poiché in al più $h f(|x|)$ passi NT accetta le parole $x \in L$, allora
- se $x \in L$, allora in $h f(|x|)$ passi una delle computazioni deterministiche di NT(x) termina nello stato di accettazione
 - allora, durante la FASE 2), poiché T simula i primi $h f(|x|)$ passi di tutte le computazioni deterministiche di NT(x) fino a quando una di esse accetta oppure non le ha esaminate tutte, prima o poi T simulerà anche quella accettante: e questo porterà T nello stato q_A
- se $x \notin L$, allora in $h f(|x|)$ passi nessuna delle computazioni deterministiche di NT(x) termina nello stato di accettazione
 - allora, durante la FASE 2), T dovrà simulare i primi $h f(|x|)$ passi di **tutte** le computazioni deterministiche di NT(x) (da quella più a sinistra nell'albero a quella più a destra, nessuna esclusa), perché nessuna di esse accetta: e questo porterà T nello stato q_R
- Questo prova che T decide L.
- Ma quanto tempo impiega? ... *to be continued* ...

La seconda questioncina aperta

- **Teorema 6.17:** Per ogni funzione *time-constructible* $f : \mathbb{N} \rightarrow \mathbb{N}$,
 $\text{NTIME}[f(n)] \subseteq \text{DTIME}[2^{O(f(n))}]$.
- T decide L.
- Ma quanto tempo impiega T a decidere L?
- Intanto, la FASE 1) richiede $O(h f(|x|))$ passi – *perché f è time-constructible*
- Poi, riguardo la FASE 2):
 - sia k il grado di non determinismo di NT – *k è costante!*
 - allora, il numero di computazioni deterministiche di NT(x) di lunghezza $h f(|x|)$ è $k^{h f(|x|)}$
 - ciascuna di queste computazioni viene simulata da T in $O(h f(|x|))$ passi.
- Allora, **$\text{dtime}(T, x) \in O(h f(|x|) + h f(|x|) k^{h f(|x|)}) = O(h f(|x|) k^{h f(|x|)}) \subseteq O(2^{O(f(|x|))})$** .
- Infine, in virtù del Teorema 6.3, esiste una macchina T_1 ad un nastro tale che
 - per ogni input x, l'esito della computazione $T_1(x)$ coincide con l'esito della computazione T(x)
 - ed esiste una costante c tale che $\text{dtime}(T_1, x) \leq \text{dtime}(T, x)^c \in O(2^{O(f(|x|))})$.
- Questo conclude la dimostrazione che $L \in \text{DTIME}[2^{O(f(|x|))}]$.

Specifiche classi di complessità

- Siamo al paragrafo 6.6, pronti a introdurre alcune fra le più rilevanti classi di complessità, definite sulla base di funzioni time- e space-constructible:
- $P = \bigcup_{k \in \mathbb{N}} \text{DTIME}[n^k]$
 - la classe dei linguaggi *decidibili in tempo deterministico polinomiale*;
- $NP = \bigcup_{k \in \mathbb{N}} \text{NTIME}[n^k]$:
 - la classe dei linguaggi *accettabili in tempo non deterministico polinomiale*;
 - ma anche *decidibili in tempo non deterministico polinomiale*!
- $PSPACE = \bigcup_{k \in \mathbb{N}} \text{DSPACE}[n^k]$
 - la classe dei linguaggi *decidibili in spazio deterministico polinomiale*;
- $NPSpace = \bigcup_{k \in \mathbb{N}} \text{NSPACE}[n^k]$
 - la classe dei linguaggi *accettabili in spazio non deterministico polinomiale*;
 - ma anche *decidibili in spazio non deterministico polinomiale*!

Specifiche classi di complessità

- Siamo al paragrafo 6.6, pronti a definire alcune fra le più rilevanti classi di complessità:
- $\text{EXPTIME} = \bigcup_{k \in \mathbb{N}} \text{DTIME}[2^{p(n,k)}]$
 - la classe dei linguaggi *decidibili in tempo deterministico esponenziale* ove l'esponente che descrive la funzione limite è un polinomio in n di grado k – indicato come $p(n,k)$
- $\text{NEXPTIME} = \bigcup_{k \in \mathbb{N}} \text{NTIME}[2^{p(n,k)}]$
 - la classe dei linguaggi *accettabili in tempo non deterministico esponenziale* (ove l'esponente che descrive la funzione limite è un polinomio in n di grado k);
 - ma anche *decidibili in tempo non deterministico esponenziale*!
- Infine, una classe di complessità di funzioni: la classe delle *funzioni (totali) calcolabili in tempo deterministico polinomiale*:
$$\text{FP} = \bigcup_{k \in \mathbb{N}} \{ f : \Sigma_1^* \rightarrow \Sigma_2^* : \text{esiste una macchina di Turing deterministica } T \text{ (di tipo trasduttore) che calcola } f \text{ e, per ogni } x \in \Sigma_1^*, \text{dtime}(T,x) \in O(|x|^k) \}.$$

Proprietà – Corollario 6.2

➤ $P \subseteq NP$, $PSPACE \subseteq NPSPACE$ e $EXPTIME \subseteq NEXPTIME$

- conseguenza diretta del Teorema 6.8: una macchina deterministica è una macchina non deterministica con grado di non determinismo 1

➤ $P \subseteq PSPACE$ e $NP \subseteq NPSPACE$

- sono conseguenza diretta del Teorema 6.9: per ogni funzione totale e calcolabile f
 $DTIME[f(n)] \subseteq DSPACE[f(n)]$ e $NTIME[f(n)] \subseteq NSPACE[f(n)]$

➤ $PSPACE \subseteq EXPTIME$ e $NPSPACE \subseteq NEXPTIME$

- sono conseguenza diretta del Teorema 6.10: per ogni funzione totale e calcolabile f
 $DSPACE[f(n)] \subseteq DTIME[2^{O(f(n))}]$ e $NSPACE[f(n)] \subseteq NTIME[2^{O(f(n))}]$

➤ $NP \subseteq EXPTIME$

- conseguenza diretta del Teorema 6.17: per ogni funzione time-constructible f
 $NTIME[f(n)] \subseteq DTIME[2^{O(f(n))}]$
- e i polinomi sono funzioni time-constructible

Relazioni interessanti, ma...

- Tutte le relazioni fra classi complessità che abbiamo, fino ad ora, dimostrato sono **inclusioni improprie**
- Ossia, per ciascuna di quelle relazioni non siamo in grado di dimostrare né l'inclusione propria né la coincidenza delle due classi che la costituiscono.
- Ad esempio, sappiamo che
 - tutti i linguaggi che sono in PSPACE sono anche in EXPTIME
 - tutti i linguaggi che sono in P sono anche in NP
- Ma non sappiamo rispondere alle seguenti domande
 - non sarà forse che tutti i linguaggi in EXPTIME sono anche in PSPACE? Ossia, che $\text{PSPACE} = \text{EXPTIME}$?
 - Oppure, esiste almeno un linguaggio in EXPTIME che non può essere deciso in spazio polinomiale? Ossia, che $\text{PSPACE} \subset \text{EXPTIME}$
- Si tratta, se volete, di relazioni deboli
 - **e sarebbe tremendo se si dimostrasse che tutte quelle inclusioni improprie fossero, in effetti, delle uguaglianze!**
 - Non saremmo affatto in grado di classificare i problemi in “facili” e “difficili”

L'unica relazione di contenimento stretto!

- In effetti, uno strumento che dimostra l'inclusione stretta fra classi di complessità ce l'abbiamo: il Teorema di gerarchia temporale:
- **Teorema 6.15 [Teorema di gerarchia temporale]:** Siano $f : \mathbb{N} \rightarrow \mathbb{N}$ e $g : \mathbb{N} \rightarrow \mathbb{N}$ due funzioni tali che f è time-constructible e

$$\lim_{n \rightarrow \infty} \frac{g(n) \log g(n)}{f(n)} = 0$$

Allora, **$\text{DTIME}[g(n)] \subset \text{DTIME}[f(n)]$** ossia, esiste un linguaggio L tale che $L \in \text{DTIME}[f(n)]$ e $L \notin \text{DTIME}[g(n)]$.

- Come conseguenza del Teorema di gerarchia temporale, vale il seguente

Teorema 6.18: $P \subset \text{EXPTIME}$

(che noi non dimostriamo e la cui dimostrazione,
per gli interessati, si trova sulle dispense)

L'unica relazione di uguaglianza!

- La maggior parte delle relazioni fra classi complessità che abbiamo visto fino ad ora, sono **inclusioni improprie**
- A parte le inclusioni proprie che derivano dal Teorema di gerarchia temporale,
- del quale abbiamo dimostrato un caso particolare:
 - il **Teorema 6.18**: $P \subset EXPTIME$
- In effetti, esiste anche un teorema che va nella direzione opposta – che dimostra, cioè, l'uguaglianza di due classi
- una classe deterministica e una classe non deterministica:
- **Teorema 6.19**: $PSPACE = NPSPACE$
 - non studiamo, quest'anno, la dimostrazione di questo teorema
 - ma sono ben lieta di discuterne con chi la vuole guardare!