Lezione 11 – misure di complessità

Lezione del 10/04/2024

Misure di complessità

- Siamo alla dispensa 6, paragrafo 6.1
- Una **misura di complessità** è una funzione **c** che associa un valore numerico ad una macchina di Turing T e ad un suo input x
  - **c(T,x)** intende rappresentare il "costo" della computazione T(x)
- Affinché **c** possa essere considerata una misura di complessità, essa deve soddisfare le due seguenti proprietà, note come *assiomi di Blum*:
- 1) **c** è definita per tutte e sole le computazioni che terminano
  - se una computazione T(x) non termina, non ha senso considerare che tale computazione abbia come costo un valore finito;
- 2) **c** deve essere una funzione calcolabile,
  - ossia, deve esistere una macchina di Turing M che, ricevendo in input una macchina di Turing T ed un suo input x, calcola **c**(T,x) ogniqualvolta **c**(T,x) è definita (cioè, ogniqualvolta *T* (x) termina)
  - intuitivamente, questo significa che, il costo di una computazione T(x) (che termina) dobbiamo poterlo calcolare effettivamente.

Misure deterministiche

- Iniziamo con le misure di complessità che si riferiscono a computazioni deterministiche.
  - per ogni macchina di Turing deterministica T (riconoscitore o trasduttore), definita su un alfabeto Σ,
  - e per ogni $x \in \Sigma^*$
- definiamo le due funzioni seguenti associate alla computazione T(x):

$$\mathbf{dtime(T,x)} = \begin{cases} \text{numero di istruzioni eseguite da } T(x) & \text{se } T(x) \text{ termina} \\ \text{non definita} & \text{se } T(x) \text{ non termina} \end{cases}$$

$$\mathbf{dspace(T,x)} = \begin{cases} \text{numero di celle di memoria utilizzate da } T(x) & \text{se } T(x) \text{ termina} \\ \text{non definita} & \text{se } T(x) \text{ non termina} \end{cases}$$

- Osserviamo che dtime e dspace sono due funzioni parziali: non sono definite se T(x) non termina

Misure deterministiche

- Dimostriamo ora che le funzioni **dtime** e **dspace** soddisfano i due assiomi di Blum.
- 1) Facile: lo abbiamo già osservato!
  - Per ogni macchina di Turing deterministica T e per ogni $x \in \Sigma^*$, dtime(T,x) e dspace(T,x) sono definite se e solo se T (x) termina.
- 2) Dobbiamo mostrare che **dtime** e **dspace** sono calcolabili. Iniziamo da **dtime**:
  - consideriamo una modifica $\mathbf{U_{dtime}}$ della macchina di Turing universale U:
  - aggiungiamo ad U il nastro N_5 che fungerà da contatore del numero di istruzioni della computazione T(x)
  - U_{dtime}(T,x) si comporta come U(T,x) con l'unica differenza che, dopo avere eseguito una quintupla della macchina T su input x ed essersi preparata ad eseguire la quintupla successiva, scrive un 1 sul nastro N_5 e muove a destra la testina su tale nastro.
  - Al termine della computazione $U_{dtime}(T,x)$ (<u>se essa termina</u>) il nastro N_5 conterrà, codificato in unario, il numero di passi eseguiti dalla computazione *T*(*x*)
  - dunque, **dtime** è una funzione calcolabile.
  - La dimostrazione che **dspace** è una funzione calcolabile è simile e la fate come esercizio.
  - NB: RIGUARDATE LA MACCHINA UNIVERSALE!

Misure non deterministiche (1)

- Passiamo ora a definire le misure di complessità che si riferiscono a computazioni non deterministiche.
  - per ogni macchina di Turing non deterministica NT (riconoscitore, per forza!), definita su un alfabeto Σ,
  - e per ogni $x \in \Sigma^*$
  - tali che NT(x) **ACCETTA**,
- definiamo le due funzioni seguenti:
  - **ntime(NT,x)** = minimo numero di istruzioni eseguite da una computazione deterministica **accettante** di NT(x)
  - **nspace(NT,x)** = minimo numero di celle utilizzate da una computazione deterministica **accettante** di NT(x).
- Osservate che ntime e nspace sono due funzioni parziali – **molto** parziali, avendole definite *solo per computazioni* ***accettanti***!
  - Potremmo aggiungere: se NT(x) non accetta, anche quando NT(x) termina, allora
    - **ntime(NT,x)** non è definita
    - **nspace(NT,x)** non è definita

Misure non deterministiche

- Ma perché, nella definizione di **ntime** e **nspace**, si parla di computazioni che "accettano" invece che di computazioni che "terminano"?
- Ve la ricordate quella (dannata) asimmetria nelle definizioni di accettazione e di rigetto di una macchina non deterministica?
  - NT(x) accetta se ***esiste*** una sua computazione deterministica che accetta
  - NT(x) rigetta se ***tutte*** le sue computazioni deterministiche rigettano
- Perciò, se vogliamo estendere le definizioni di ntime e nspace a tutte le computazioni che terminano, dobbiamo dire che: per ogni macchina di Turing non deterministica NT, definita su un alfabeto Σ, e per ogni $x \in \Sigma^*$ tali che NT(x) **RIGETTA**,
  - **ntime(NT,x)** = **massimo** numero di istruzioni eseguite da una computazione deterministica rigettante di NT(x)
  - **nspace(NT,x)** = **massimo** numero di celle utilizzate da una computazione deterministica rigettante di NT(x).

Misure non deterministiche (2)

- Nel seguito di questo corso, faremo riferimento alla definizione delle funzioni **ntime** e **nspace** che tengono conto anche delle computazioni che rigettano
- ossia,

ntime(NT,x) =
- **min** # di istruzioni eseguite da una comp. det. accettante di NT(x), se NT(x) accetta
- **max** # di istruzioni eseguite da una comp. det. rigettante di NT(x), se NT(x) rigetta
- non definita, altrimenti

nspace(NT,x) =
- **min** # di celle utilizzate da una comp. det. accettante di NT(x), se NT(x) accetta
- **max** # di celle eseguite da una comp. det. rigettante di NT(x), se NT(x) rigetta
- non definita, altrimenti

- Anche con questa estensione, le funzioni **ntime** e **nspace** restano funzioni parziali
- Per esercizio, dimostrate che soddisfano gli assiomi di Blum

Relazioni fra spazio e tempo

- **Teorema 6.1** (caso deterministico): *Sia* T *una macchina di Turing* ***deterministica****, definita su un alfabeto* Σ *(non contenente il simbolo* □ *) e un insieme degli stati* Q*, e sia* x ∈ Σ* *tale che* T(x) *termina. Allora,*

  dspace(T,x) ≤ dtime(T,x) ≤ dspace(T,x) | Q | $(|\Sigma| + 1)^{\mathbf{dspace(T,x)}}$.

- **1) dspace(T,x) ≤ dtime(T,x)**
  - Facile: se T(x) utilizza dspace(T,x) celle di memoria, quelle celle deve almeno leggerle
  - e, per leggere ciascuna cella impiega un'istruzione (ossia, esegue una quintupla)
  - Fine.
  - Beh, in realtà, potrebbe utilizzare un input molto più lungo del necessario
  - pensate, ad esempio, se per sommare i due numeri 15 e 6, scrivessimo l'input nella forma: 000000000000000000000000000000000000000000000000000000000000=15+6
  - e poi posizionassimo la testina sul carattere '=' all'inizio della computazione: in questo caso non avremmo bisogno di leggere i tantissimi '0' iniziali
  - per questo la definizione di dspace è un po' diversa da quella che abbiamo visto
  - ma noi ci teniamo quella e non consideriamo questi casi "anomali"

Relazioni fra spazio e tempo

- **Teorema 6.1** (caso deterministico): *Sia* T *una macchina di Turing* ***deterministica****, definita su un alfabeto* Σ *(non contenente il simbolo* □ *) e un insieme degli stati Q, e sia* x ∈ Σ* *tale che* T (x) *termina. Allora,*
- **2) dtime(T,x) ≤ dspace(T,x)|Q|** $(|\boldsymbol{\Sigma}| + \mathbf{1})^{\mathbf{dspace(T,x)}}$.
  - Un po' meno facile...
  - Osserviamo che **dspace(T,x)|Q|** $(|\boldsymbol{\Sigma}| + \mathbf{1})^{\mathbf{dspace(T,x)}}$ è il numero di stati globali possibili di T nel caso in cui non più di dspace(T,x) celle del nastro vengano utilizzate dalla computazione T(x)
  - Infatti:
    - poiché ogni cella del nastro può contenere un simbolo di Σ oppure il blank, il numero di possibili configurazioni di dspace(T,x) celle del nastro è $(|\boldsymbol{\Sigma}| + \mathbf{1})^{\mathbf{dspace(T,x)}}$
    - poi, per ognuna di queste configurazioni
      - la testina può trovarsi su una qualsiasi delle dspace(T,x) celle
      - e la macchina può essere in uno qualsiasi dei |Q| stati interni
  - e questo è ben spiegato nella dispensa.
  - Chiamiamo k(T,x) questo valore: **k(T,x)** = **dspace(T,x)|Q|** $(|\boldsymbol{\Sigma}| + \mathbf{1})^{\mathbf{dspace(T,x)}}$

Relazioni fra spazio e tempo

- **Teorema 6.1** (caso deterministico): *Sia* T *una macchina di Turing* ***deterministica****, definita su un alfabeto Σ (non contenente il simbolo □) e un insieme degli stati Q, e sia* x ∈ Σ* *tale che* T(x) *termina. Allora,*
- **2) dtime(T,x) ≤ dspace(T,x)|Q| $(|\mathbf{\Sigma}| + \mathbf{1})^{\mathbf{dspace(T,x)}}$.**
  - Dunque: **k(T,x)** = **dspace(T,x)|Q|**$(|\mathbf{\Sigma}| + \mathbf{1})^{\mathbf{dspace(T,x)}}$ è il numero di stati globali possibili di T nel caso in cui non più di dspace(T,x) celle del nastro vengano utilizzate dalla computazione T(x)
  - Ora, ricordiamo che una computazione (deterministica) è una successione di stati globali tali che si passa da uno stato globale al successivo eseguendo una quintupla
  - se T(x) durasse più di **k(T,x)** passi (*senza uscire mai dalle dspace(T,x) celle*), allora sarebbe una successioni di stati globali contenente almeno due volte uno stesso stato globale – chiamiamolo SG_h:

$$SG_1 \rightarrow SG_2 \rightarrow \ldots \rightarrow SG_h \rightarrow SG_{h+1} \rightarrow SG_{h+2} \rightarrow \ldots \rightarrow SG_{k(T,x)}$$

Relazioni fra spazio e tempo

- **Teorema 6.1** (caso deterministico): *Sia* T *una macchina di Turing* ***deterministica****, definita su un alfabeto* Σ *(non contenente il simbolo* □ *) e un insieme degli stati* Q*, e sia* x ∈ Σ* *tale che* T(x) *termina. Allora,*
- **2) dtime(T,x) ≤ dspace(T,x)|Q| $(|\boldsymbol{\Sigma}| + \mathbf{1})^{\mathbf{dspace(T,x)}}$.**
  - Dunque: **k(T,x)** = **dspace(T,x)|Q|**$(|\boldsymbol{\Sigma}| + \mathbf{1})^{\mathbf{dspace(T,x)}}$ è il numero di stati globali possibili di T nel caso in cui non più di dspace(T,x) celle del nastro vengano utilizzate dalla computazione T(x)
  - se T(x) durasse più di **k(T,x)** passi (*senza uscire mai dalle dspace(T,x) celle*), allora sarebbe una successioni di stati globali contenente almeno due volte uno stesso stato globale – chiamiamolo SG_h:

$$SG_1 \rightarrow SG_2 \rightarrow \ldots \rightarrow SG_h \rightarrow SG_{h+1} \rightarrow SG_{h+2} \rightarrow \ldots \rightarrow SG_{k(T,x)}$$

  - ma T è deterministica; allora, a partire da SG_h è possibile eseguire un'unica quintupla (quella che porta nello stato globale SG_{h+1}) ed essa viene eseguita tutte le volte in cui T(x) si trova in SG_h
  - quindi, entrambe le volte, avviene una transizione verso lo stesso stato globale SG_{h+1}
  - e così via, e così via: T(x) sarebbe in loop (contro l'ipotesi che termina)
    - **studiatelo sulla dispensa (dove è descritto meglio!)**

Relazioni fra spazio e tempo

- **Teorema 6.1** (caso non deterministico): *Sia* NT *una macchina di Turing* ***non deterministica****, definita su un alfabeto* Σ *(non contenente il simbolo* □ *) e un insieme degli stati Q, e sia* x ∈ Σ* *tale che* NT(x) *accetta/termina. Allora,*

$$\text{nspace}(NT,x) \leq \text{ntime}(NT,x) \leq \text{nspace}(NT,x)\,|Q|\,(|\Sigma|+1)^{\text{nspace}(NT,x)}.$$

- Questa dimostrazione è sensibilmente più complessa di quella del caso deterministico, e ve la risparmio: non dovete studiarla
- Ma vi consiglio di guardarla: per vostra cultura, per capire come funziona il caso non deterministico

Verso le classi di complessità

- Sia $f : \mathbb{N} \to \mathbb{N}$ una funzione **totale calcolabile**.
- Sia Σ un alfabeto finito e sia $x \in \Sigma^*$: indichiamo con $|x|$ il numero di caratteri di x

- Un linguaggio $L \subseteq \Sigma^*$ è ***deciso in tempo (spazio) deterministico*** $\mathbf{f(n)}$ se
  - esiste una macchina di Turing deterministica T che decide L e tale che,
  - **per ogni $x \in \Sigma^*$**, $dtime(T,x) \leq f(|x|)$ ($dspace(T,x) = f(|x|)$).

- Un linguaggio $L \subseteq \Sigma^*$ è ***accettato in tempo (spazio) non deterministico*** $\mathbf{f(n)}$ se
  - esiste una macchina di Turing non deterministica NT che accetta *L* e tale che,
  - **per ogni $x \in L$**, $ntime(NT,x) \leq f(|x|)$ ($nspace(NT,x) \leq f(|x|)$)

- Un linguaggio $L \subseteq \Sigma^*$ è ***deciso in tempo (spazio) non deterministico*** $\mathbf{f(n)}$ se
  - esiste una macchina di Turing non deterministica NT che decide *L* e tale che
  - **per ogni $x \in \Sigma^*$**, $ntime(NT,x) \leq f(|x|)$ ($nspace(NT,x) \leq f(|x|)$).

Dall'accettazione alla decisione

- Osservate che
  - nel caso deterministico definiamo soltanto i linguaggi <u>decisi</u> in un certo tempo o spazio
  - nel caso non deterministico distinguiamo in linguaggi accettati in un certo tempo o spazio da quelli decisi nello stesso tempo o spazio

- In seguito a tale distinzione si potrebbe pensare che esistono linguaggi che sono accettabili in un certo tempo o spazio non deterministico, ma che non sono decidibili
  - ossia, il loro complemento non è accettabile
- In effetti, non è così: il prossimo teorema mostra che, *ogni qualvolta una funzione totale e calcolabile limita la quantità di risorse disponibili al fine di accettare le parole di un linguaggio, i concetti di accettabilità e di decidibilità coincidono*.
- Ossia, **la teoria della complessità computazionale si occupa solo di linguaggi decidibili**

Dall'accettazione alla decisione

- Osservate che
  - nel caso deterministico definiamo soltanto i linguaggi decisi in un certo tempo o spazio
  - nel caso non deterministico distinguiamo in linguaggi accettati in un certo tempo o spazio da quelli decisi nello stesso tempo o spazio
- In seguito a tale distinzione si potrebbe pensare che esistono linguaggi che sono accettabili in un certo tempo o spazio non deterministico, ma che non sono decidibili
  - ossia, il loro complemento non è accettabile
- In effetti, non è così: il prossimo teorema mostra che, *ogni qualvolta una funzione totale e calcolabile limita la quantità di risorse disponibili al fine di accettare le parole di un linguaggio, i concetti di accettabilità e di decidibilità coincidono*.
- Ossia, **la teoria della complessità computazionale si occupa solo di linguaggi decidibili**

Dall'accettazione alla decisione

- **Teorema 6.2** (tempo): *Sia* f : ℕ → ℕ *una funzione totale calcolabile.* *Se* L ⊆ Σ* *è accettato da una macchina di di Turing non deterministica* NT *tale che, per ogni* x ∈ L, ntime(NT,x) ≤ f (|x|)] *allora* L *è decidibile.*

- Poiché f è totale calcolabile, esiste una macchina T_f di tipo trasduttore tale che, per ogni n ∈ ℕ, T_f (n) termina con il valore f(n) scritto sul nastro di output
  - senza perdita di generalità, assumiamo che T_f scriva f(n) in unario sul nastro di output
  - perché possiamo assumerlo?
- Costruiamo una nuova macchina non deterministica NT', a tre nastri, che decide L: per ogni x ∈ Σ*
  - FASE 1) NT'(x) scrive |x| sul secondo nastro e invoca T_f(|x|): al termine della computazione sul terzo nastro si troverà scritto f(|x|) in unario
  - FASE 2) NT'(x) **simula** NT(x) e, per ogni quintupla eseguita da NT(x):
    - NT' "cancella" un '1' dal terzo nastro e, inoltre,
    - se NT(x) accetta allora anche NT'(x) accetta, se NT(x) rigetta allora anche NT'(x) rigetta;
  - se quando il terzo nastro di NT' è vuoto NT(x) non ha ancora terminato, allora NT'(x) rigetta

Dall'accettazione alla decisione

- **Teorema 6.2** (tempo): *Sia* f : ℕ → ℕ *una funzione totale calcolabile.* *Se* $L \subseteq \Sigma^*$ *è accettato da una macchina di di Turing non deterministica* NT *tale che, per ogni* $x \in L$, ntime(NT,x) ≤ f (|x|)] *allora* L *è decidibile.*

- Osserviamo, intanto, che le computazioni di NT' terminano sempre
  - se una computazione NT'(x) dura più di f(|x|) passi, la interrompiamo!
- Poi, NT' decide L, infatti:
  - se $x \in L$, allora NT(x) accetta in al più f(|x|) passi: e, quindi, NT'(x) accetta
  - se $x \notin L$, allora o NT(x) rigetta in al più f(|x|) passi e, quindi, NT'(x) rigetta, oppure NT(x) non termina entro f(|x|) passi e, quindi, NT'(x), ugualmente, rigetta
- **Ma quanto impiega NT' a rigettare $x \notin L$?**
  - Boh?! Che ne sappiamo quanto tempo impiega T_f a calcolare f(|x|)?
  - Sappiamo solo che T_f (|x|) termina, ma non in quanto tempo!
- **Per questo possiamo concludere che L è decidibile, ma non possiamo concludere che è deciso in tempo non deterministico f(n)**

Dall'accettazione alla decisione

- **Teorema 6.2** (spazio): *Sia* f : $\mathbb{N} \rightarrow \mathbb{N}$ *una funzione totale calcolabile.*
Se $L \subseteq \Sigma^*$ *è accettato da una macchina di di Turing non deterministica* NT *tale che, per ogni* $x \in L$, nspace(NT,x) $\leq$ f (| x |)] *allora L è decidibile.*

- La dimostrazione è analoga al caso di ntime
  - e ve la risparmio
  - non siete tenuti a studiarla
  - cioè, non siete tenuti a studiare le ultime 4 righe della dimostrazione del Teorema 6.2 sulla dispensa
  - tutto il resto della dimostrazione, però, siete tenuti eccome a studiarla!

Complessità e modelli di calcolo

- Siamo al paragrafo 6.2 della dispensa 6
- Qui si dimostra che che tutti i modelli di calcolo deterministici sono fra loro **polinomialmente correlati**
  - Macchine di Turing ad un nastro
  - Macchine di Turing a quanti nastri ci pare
  - Macchine di Turing su alfabeto binario
  - Macchine di Turing su alfabeti grandi quanto ci pare
- Ma che vuol dire che questi modelli sono fra loro polinomialmente correlati?
  - che per ogni macchina di Turing T di uno di questi tipi esistono una macchina di Turing T' di uno qualunque degli altri tipi ed un polinomio p tali che T' risolve lo stesso problema risolto da T e, per ogni x, dtime(T',x) ≤ p(dtime(T,x)) e dspace(T',x) ≤ p(dspace(T,x))
- E anche che il modello Macchina di Turing è polinomialmente correlato con il PascalMinimo

Complessità e modelli di calcolo

- Siamo al paragrafo 6.2 della dispensa 6
- Qui si dimostra che che tutti i modelli di calcolo deterministici sono fra loro polinomialmente correlati
- Ok, bello, ma che ce ne importa? cosa significa tutto ciò?
  - Che possiamo risolvere un problema utilizzando il modello che più ci aggrada
  - ad esempio, per risolvere un certo problema possiamo scrivere un algoritmo A in PascalMinimo (invece che stare lì a progettare quintuple di una macchina di Turing)
  - e se A trova la soluzione di una istanza x del problema eseguendo $f(|x|)$ istruzioni
  - allora esiste una macchina di Turing T ad un nastro che risolve lo stesso problema, ed esiste un polinomio p tale che $dtime(T,x) \leq p(f(|x|)$
- Chiara l'idea?
  - anche se ve l'ho raccontata come "risolvere problemi" invece che "decidere linguaggi"
  - e non vi ho detto cosa indica $|x|$ al di fuori del mondo delle macchine di Turing
  - ma voi lo intuite, vero?

Complessità e modelli di calcolo

- Siamo al paragrafo 6.2 della dispensa 6
- Qui si dimostra che che tutti i modelli di calcolo deterministici sono fra loro polinomialmente correlati
- Ok, bello, ma perché è così importante sapere che sono **polinomialmente** correlati?
  - Beh, perché se abbiamo un algoritmo in PascalMinimo che impiega un numero di istruzioni polinomiale nella lunghezza dell'input per risolvere il problema
  - sappiamo anche che esiste una macchina di Turing che risolve lo stesso problema eseguendo, anch'essa, un numero di istruzioni polinomiale nella lunghezza dell'input
- E (ve lo ricordate) un problema è trattabile se il tempo necessario a risolverlo è polinomiale (nella dimensione dell'input)
  - perciò, se un problema è trattabile rispetto ad un modello, è trattabile anche rispetto a tutti gli altri!
- Ma di questo parleremo a lungo, fra un po' di tempo

Complessità e modelli di calcolo

- Siamo al paragrafo 6.2 della dispensa 6
- Per il momento, vi basti sapere della correlazione polinomiale
  - e accontentatevi dell'idea (informale) di motivazione dell'importanza di questa correlazione che vi ho proposto
- Solo una questioncina merita di essere specificata: |x| lo leggiamo come ***lunghezza* di x**
  - qualunque sia il modello di calcolo che utilizzate, |x| rappresenta la quantità di memoria che occorre a rappresentare x in quel modello
  - Cioè? Il numero di celle di nastro di una macchina di Turing, il numero di bit di una RAM, ecc. ecc. ecc.
- Un'ultima cosa: i teoremi del paragrafo 6.2 non li dovete studiare
  - ma se avete voglia di guardarli, stanno là (e io sto qua per discuterne)