



Lezione29 - caratterizzazione di NP

Lezione del 16/05/2025

Da una condizione sufficiente...

- Abbiamo visto che tutti i problemi decisionali tali che
 - 1) il predicato ha la forma $\pi(x, S(x)) = \text{esiste } y \in S(x) \text{ tale che } \eta(x,y)$
 - 2) la scelta di un elemento y di $S(x)$ richiede tempo non deterministico polinomiale in $|x|$
 - 3) la verifica che y soddisfi il predicato η , richiede tempo polinomiale in $|x|$
- appartengono ad NP
- Ossia: 1), 2) e 3) sono condizioni sufficienti per poter affermare che un problema appartiene ad NP
- Sono condizioni facili da verificare
 - certo, più facili che non verificare l'esistenza di un algoritmo non deterministico che operi in tempo polinomiale
- Però, è chiaro, non è mica detto che tutti i problemi in NP soddisfino queste 3 condizioni!
 - Ossia, magari esiste un problema che **non** soddisfa 1) e 2) e 3)
 - **e che, tuttavia, appartiene ad NP!** ... Sarà forse così?!
- In effetti no: non è così! Infatti, c'è un teorema che...

... ad una nuova caratterizzazione ...

- **Teorema 9.1:** Un linguaggio $L \subseteq \Sigma^*$ appartiene ad NP **se e soltanto se**
 - esistono una macchina di Turing deterministica T e due costanti $h, k \in \mathbb{N}$ tali che,
 - per ogni $x \in \Sigma^*$,
 - $x \in L \leftrightarrow \exists y_x \in \{0,1\}^* : |y_x| \leq |x|^k$
 $\wedge T(x, y_x)$ accetta
 $\wedge \text{dtime}(T, x, y_x) \in O(|x|^h)$.
- Ma che vuol dire?!
 - Tipico caso di teorema più difficile da enunciare che da dimostrare...

... di cui capire il significato

- **Teorema 9.1:** Un linguaggio $L \subseteq \Sigma^*$ appartiene ad NP **se e soltanto se**
 - esistono una macchina di Turing deterministica T e due costanti $h, k \in \mathbb{N}$ tali che,
 - **per ogni $x \in \Sigma^*$,**
 - **$x \in L \leftrightarrow \exists y_x \in \{0,1\}^* : |y_x| \leq |x|^k \wedge T(x, y_x) \text{ accetta} \wedge \text{dtime}(T, x, y_x) \in O(|x|^h)$.**
- Ma che vuol dire?!
- Intanto, osserviamo che il teorema è una **condizione necessaria e sufficiente** per poter dire “ L appartiene ad NP”
- E, siccome è una condizione necessaria e sufficiente, dobbiamo scomporlo in due parti
- Se partiamo dall'ipotesi che L appartiene a NP, il teorema, ci indica **una condizione necessaria e sufficiente per poter affermare che $x \in L$:**
 - **per ogni $x \in \Sigma^*$, $x \in L \leftrightarrow \dots$ ecc. ecc.**

Che vuol dire?

- **Teorema 9.1:** Un linguaggio $L \subseteq \Sigma^*$ appartiene ad NP **se e soltanto se**
 - esistono una macchina di Turing deterministica T e due costanti $h, k \in \mathbb{N}$ tali che,
 - per ogni $x \in \Sigma^*$,
 - $x \in L \leftrightarrow \exists y_x \in \{0,1\}^* : |y_x| \leq |x|^k \wedge T(x, y_x) \text{ accetta} \wedge \text{dtime}(T, x, y_x) \in O(|x|^h)$.
- Se partiamo dall'ipotesi che L appartiene a NP, il teorema, ci indica una **condizione necessaria e sufficiente** per poter affermare che $x \in L$: per ogni $x \in \Sigma^*$, $x \in L \leftrightarrow \dots$ ecc. ecc.
- Cerchiamo, ora, di capire qual è questa **condizione**!
- a) $\exists y_x \in \{0,1\}^* :$ - ci dice che, per poter affermare che $x \in L$, dobbiamo trovare una parola da associare ad $x \dots$
- b) $|y_x| \leq |x|^k$... che non sia troppo lunga ...
- c) $\wedge T(x, y_x) \text{ accetta}$... e che induca *una certa macchina deterministica T* ad accettare...
- d) $\wedge \text{dtime}(T, x, y_x) \in O(|x|^h)$... e ad accettare in tempi brevi!
- **MA CHI È QUESTA CERTA MACCHINA DETERMINISTICA T????**

"Se avessi un piccolo aiuto"

- **Teorema 9.1:** Un linguaggio $L \subseteq \Sigma^*$ appartiene ad NP **se e soltanto se**
 - esistono **una macchina di Turing deterministica T** e due costanti $h, k \in \mathbb{N}$ tali che,
 - per ogni $x \in \Sigma^*$,
 - $x \in L \leftrightarrow \exists y_x \in \{0,1\}^* : |y_x| \leq |x|^k \wedge T(x, y_x) \text{ accetta} \wedge \text{dtime}(T, x, y_x) \in O(|x|^h)$.
- Se partiamo dall'ipotesi che appartiene ad NP, il teorema, ci indica una **condizione necessaria e sufficiente** per poter affermare che $x \in L$: per ogni $x \in \Sigma^*$, $x \in L \leftrightarrow \dots$ ecc. ecc.
- Meglio: il teorema ci dice che **se $L \in \text{NP}$ allora esiste una macchina deterministica T** tale che,
 - se le do in input due parole x e y , **con y scelta da me e non troppo lunga**,
 - $T(x,y)$, in tempo polinomiale in $|x|$, **accetta se e soltanto se $x \in L$ e io ho scelto la y giusta**
 - perché, mi dice il teorema, **una parola y_x che possa convincere T ad accettare riesco a trovarla se $x \in L$, ma non riesco a trovarla se $x \notin L$**
- Allora, **se trovo qualcuno in grado di suggerirmi, per ogni $x \in L$, la parola y_x giusta**
- io le parole di L riesco ad accettarle in tempo deterministico polinomiale!

Il ritorno del Genio

- **Teorema 9.1:** Un linguaggio $L \subseteq \Sigma^*$ appartiene ad NP se e soltanto se
 - esistono **una macchina di Turing deterministica T** e due costanti $h, k \in \mathbb{N}$ tali che,
 - per ogni $x \in \Sigma^*$,
 - $x \in L \leftrightarrow \exists y_x \in \{0,1\}^* : |y_x| \leq |x|^k \wedge T(x, y_x) \text{ accetta} \wedge \text{dtime}(T, x, y_x) \in O(|x|^h)$.
- Mi dice il teorema che, se $L \in \text{NP}$, **se trovo qualcuno in grado di suggerirmi, per ogni $x \in L$, la parola y_x giusta**, io le parole di L riesco ad accettarle in tempo deterministico polinomiale!
- Ma, a pensarci bene, questo già lo sapevamo...
- Se al Genio, invece di chiedere una quintupla alla volta, chiedo: "Ehilà, Genio! Ho questa parola x che potrebbe appartenere ad un linguaggio $L \in \text{NP}$ che è accettato da una macchina non deterministica NT; mi dici la sequenza di quintuple di NT che costituiscono una computazione accettante di $\text{NT}(x)$?"
- quello, il Genio, mi comunica una certa parola – chiamiamola y_x
 - **che descrive la sequenza di quintuple che egli afferma costituire una computazione accettante di $\text{NT}(x)$**
- Ma, io, del Genio mica mi fido! Devo verificare che mi abbia detto la verità ...

Un Genio con i suoi limiti

- **Teorema 9.1:** Un linguaggio $L \subseteq \Sigma^*$ appartiene ad NP se e soltanto se
 - esistono **una macchina di Turing deterministica T** e due costanti $h, k \in \mathbb{N}$ tali che,
 - per ogni $x \in \Sigma^*$,
 - $x \in L \leftrightarrow \exists y_x \in \{0,1\}^* : |y_x| \leq |x|^k \wedge T(x, y_x) \text{ accetta} \wedge \text{dtime}(T, x, y_x) \in O(|x|^h)$.
- Se al Genio, invece di chiedere una quintupla alla volta, chiedo: “mi dici la sequenza di quintuple di NT che costituiscono una computazione accettante di $NT(x)$?”
- quello, il Genio, mi comunica una certa parola – chiamiamola y_x
- Ma, io, del Genio mica mi fido! Devo verificare che mi abbia detto la verità
 - devo verificare, innanzi tutto, che y_x sia una sequenza di quintuple di NT
 - poi che y_x sia una sequenza di quintuple che NT può eseguire su input x – ossia, che corrisponda ad una computazione deterministica di $NT(x)$
 - infine, devo verificare che y_x corrisponda ad una computazione accettante.
 - Se tutte e tre le prove hanno esito positivo, allora posso concludere che $x \in L$
 - il Genio mi ha detto la verità!
- Che seccatura questa verifica, però!

Dalla verifica al verificatore

- **Teorema 9.1:** Un linguaggio $L \subseteq \Sigma^*$ appartiene ad NP se e soltanto se
 - esistono **una macchina di Turing deterministica T** e due costanti $h, k \in \mathbb{N}$ tali che,
 - per ogni $x \in \Sigma^*$,
 - $x \in L \leftrightarrow \exists y_x \in \{0,1\}^* : |y_x| \leq |x|^k \wedge T(x, y_x) \text{ accetta} \wedge \text{dtime}(T, x, y_x) \in O(|x|^h)$.
- Quasi quasi, costruisco una macchina deterministica T che esegua la verifica al posto mio!
 - E, poiché T mi serve a far verifiche, la chiamo **verificatore**
- E quanto impiega T(x, y_x) ad eseguire la verifica?
 - innanzi tutto, poiché $L \in \text{NP}$, se $x \in L$ allora esiste una computazione deterministica accettante di NT lunga $|x|^k$ passi - dove $\text{ntime}(\text{NT}, z) \leq |z|^k$ per ogni $z \in L$
 - perciò, $|y_x| \leq |x|^k$ (in realtà $O(|x|^k)$, ma facciamola facile...)
 - per verificare che y_x sia una sequenza di quintuple di NT, T impiega $O(|y_x|)$ passi
 - per verificare che y_x corrisponda ad una computazione accettante di NT(x), T deve simulare l'esecuzione delle quintuple descritte in y_x (un po' come farebbe la macchina universale!) – e, dunque, simula $|x|^k$ passi di NT e impiega $O(|x|^k \cdot |y_x|) \subseteq O(|x|^{2k})$ passi
- Ossia, T impiega tempo polinomiale in $|x|$ per verificare che il genio mi ha detto la verità

Una caratterizzazione ... geniale

- **Teorema 9.1:** Un linguaggio $L \subseteq \Sigma^*$ appartiene ad NP se e soltanto se
 - esistono **una macchina di Turing deterministica T** e due costanti $h, k \in \mathbb{N}$ tali che,
 - per ogni $x \in \Sigma^*$,
 - $x \in L \leftrightarrow \exists y_x \in \{0,1\}^* : |y_x| \leq |x|^k \wedge T(x, y_x) \text{ accetta} \wedge \text{dtime}(T, x, y_x) \in O(|x|^h)$.
- Ricapitoliamo:
 - se $L \in \text{NP}$ – e, quindi, è accettato da una macchina non deterministica NT tale che, per ogni $x \in L$, $\text{ntime}(\text{NT}, x) \leq |x|^k$
 - **se ho un Genio in grado di suggerirmi, per ogni $x \in L$, la parola y_x che corrisponde ad una computazione accettante di NT(x) ,**
 - allora **posso costruire** un **verificatore deterministico T** tale che,
 - se do in input a T una parola x e la parola y_x che mi ha suggerito il Genio,
 - $T(x, y_x)$ accetta se e solo se $x \in L$ e il Genio mi ha comunicato la parola y_x corretta
 - e lo fa in tempo polinomiale in $|x|$
 - **ATTENZIONE:** T è in grado di verificare che il Genio non ha mentito **solo se $x \in L$!**

Una caratterizzazione ... geniale

- **Teorema 9.1:** Un linguaggio $L \subseteq \Sigma^*$ appartiene ad NP se e soltanto se
 - esistono **una macchina di Turing deterministica T** e due costanti $h, k \in \mathbb{N}$ tali che,
 - per ogni $x \in \Sigma^*$,
 - $x \in L \leftrightarrow \exists y_x \in \{0,1\}^* : |y_x| \leq |x|^k \wedge T(x, y_x) \text{ accetta} \wedge \text{dtime}(T, x, y_x) \in O(|x|^h)$.
- Se $L \in \text{NP}$ – e, quindi, è accettato da una macchina non deterministica NT tale che, per ogni $x \in L$, $\text{ntime}(\text{NT}, x) \leq |x|^k$
- **se ho un Genio in grado di suggerirmi, per ogni $x \in L$, la parola y_x che corrisponde ad una computazione accettante di NT(x) ,**
- allora posso costruire un **verificatore deterministico T** che, per ogni $x \in L$, accetta (x, y_x) solo se il Genio non ha mentito
- ATTENZIONE: T è in grado di verificare che il Genio non ha mentito solo se $x \in L$!
- **Se $x \notin L$, non c'è verso: il povero Genio una parola che corrisponda ad una computazione accettante di NT(x) non può trovarla!**
- Ma, per come abbiamo costruito T,
se $x \notin L$, qualunque parola y ci venga indicata dal Genio, T(x,y) rigetta

Una nuova caratterizzazione (lato $\rightarrow$)

- **Teorema 9.1:** Un linguaggio $L \subseteq \Sigma^*$ appartiene ad NP se e soltanto se
 - esistono **una macchina di Turing deterministica** T e due costanti $h, k \in \mathbb{N}$ tali che,
 - per ogni $x \in \Sigma^*$,
 - $x \in L \leftrightarrow \exists y_x \in \{0,1\}^* : |y_x| \leq |x|^k \wedge T(x, y_x) \text{ accetta} \wedge \text{dtime}(T, x, y_x) \in O(|x|^h)$.
- Ri-Ri-capitoliamo
- Se $L \in \text{NP}$ – e, quindi, è accettato da una macchina non deterministica NT tale che, per ogni $x \in L$, $\text{ntime}(NT, x) \leq |x|^k$
- allora posso costruire un **verificatore deterministico** T che prende due input:
per ogni $x \in \Sigma^*$, accetta (x, y_x) se e solo se y_x è la codifica di una computazione accettante di $NT(x)$
- Quindi: **se $x \in L$ allora esiste una parola y_x tale che $T(x, y_x)$ accetta**
- Ma **se $x \notin L$ non esiste alcuna parola y tale che $T(x, y)$ accetta**
- Infine, siccome per ogni $x \in L$, $\text{ntime}(NT, x) \leq |x|^k$,
allora posso fare in modo che, per ogni $x \in L$ e per ogni y tale che $|y| \leq |x|^k$,
 $\text{dtime}(T, x, y) \leq |x|^{hk}$

Una nuova caratterizzazione (lato $\rightarrow$)

- **Teorema 9.1:** Un linguaggio $L \subseteq \Sigma^*$ appartiene ad NP se e soltanto se
 - esistono una macchina di Turing deterministica T e due costanti $h, k \in \mathbb{N}$ tali che,
 - per ogni $x \in \Sigma^*$,
 - $x \in L \leftrightarrow \exists y_x \in \{0,1\}^* : |y_x| \leq |x|^k \wedge T(x, y_x) \text{ accetta} \wedge \text{dtime}(T, x, y_x) \in O(|x|^h)$.
- E così, ridendo e scherzando
... e giocando col Genio
- abbiamo seriamente dimostrato la prima parte del teorema:
- se $L \in \text{NP}$
allora esistono una macchina di Turing deterministica T e due costanti $h, k \in \mathbb{N}$ tali
che, per ogni $x \in \Sigma^*$,
 $x \in L \leftrightarrow \exists y_x \in \{0,1\}^* : |y_x| \leq |x|^k \wedge T(x, y_x) \text{ accetta} \wedge \text{dtime}(T, x, y_x) \in O(|x|^h)$.
 - e questa parte la trovate (più formalizzata) alle pag. 8 e 9 (fino al riga 11) della dispensa 6
- Prima di passare alla dimostrazione della seconda parte, una paio di questioncine...

Un paio di precisazioni

- Prima di passare alla dimostrazione della seconda parte, paio di questioncine:
- 1) Intanto, nell'enunciato del teorema si parla dell'esistenza di una " $y_x \in \{0,1\}^*$ ", ma la y_x che abbiamo tirato fuori nella dimostrazione mica è una parola in $\{0,1\}^*$
 - ma questa cosa la sappiamo gestire: abbiamo parlato un sacco di volte di codifiche (binarie)!
 - e di come trasformare una macchina di Turing definita su un alfabeto generico in una macchina di Turing definita sull'alfabeto $\{0,1\}$
 - e in modo tale che le due macchine siano polinomialmente correlate
 - ed è quello che si dice alle righe 12-18 a pag. 9
- 2) Poi, in effetti, quel che chiediamo al Genio è: " x appartiene ad L ?"
- E, se $x \in L$, il Genio risponde: "sì!".
- Ma noi non gli crediamo... E, allora, il poverino, per *dimostrarci* che ha detto il vero, ci comunica la parola y_x – che poi noi verifichiamo
- Per questo, se $x \in L$, y_x prende il nome di **dimostrazione** o di **certificato** per x

Una nuova caratterizzazione (lato $\leftarrow$)

- **Teorema 9.1:** Un linguaggio $L \subseteq \Sigma^*$ appartiene ad NP se e soltanto se
 - esistono una macchina di Turing deterministica T e due costanti $h, k \in \mathbb{N}$ tali che,
 - per ogni $x \in \Sigma^*$,
 - $x \in L \leftrightarrow \exists y_x \in \{0,1\}^* : |y_x| \leq |x|^k \wedge T(x, y_x) \text{ accetta} \wedge \text{dtime}(T, x, y_x) \in O(|x|^h)$.
- Dobbiamo dimostrare la seconda parte del teorema:
- dato un linguaggio L
 - se esistono una macchina di Turing deterministica T e due costanti $h, k \in \mathbb{N}$ tali che,
 - per ogni $x \in \Sigma^*$,
 - $x \in L \leftrightarrow \exists y_x \in \{0,1\}^* : |y_x| \leq |x|^k \wedge T(x, y_x) \text{ accetta} \wedge \text{dtime}(T, x, y_x) \in O(|x|^h)$
- dobbiamo dimostrare che $L \in \text{NP}$
 - ossia, dobbiamo dimostrare che esistono una macchina di Turing non deterministica NT e un intero a tale che
 - per ogni $x \in L$, $NT(x)$ accetta e $\text{ntime}(NT, x) \in O(|x|^a)$
 - per ogni $x \notin L$, $NT(x)$ non accetta

Una nuova caratterizzazione (lato $\leftarrow$)

- Dato un linguaggio L , sappiamo che
 - esistono una macchina di Turing deterministica T e due costanti $h, k \in \mathbb{N}$ tali che,
 - per ogni $x \in \Sigma^*$,
 - $x \in L \leftrightarrow \exists y_x \in \{0,1\}^* : |y_x| \leq |x|^k \wedge T(x, y_x) \text{ accetta} \wedge \text{dtime}(T, x, y_x) \in O(|x|^h)$
- dobbiamo dimostrare che esistono una macchina di Turing non deterministica NT e un intero a tale che, per ogni $x \in L$, $NT(x)$ accetta e $\text{ntime}(NT, x) \in O(|x|^a)$
- E come facciamo a dimostrare che esistono NT e a ?
 - 1) costruiamo NT
 - sfruttando quello che sappiamo sulle parole in L e usando T
 - 2) dimostriamo che NT accetta L
 - 3) dimostriamo che, sulle parole di L , NT opera in tempo polinomiale
- Ed è più semplice di quanto si possa pensare!

Una nuova caratterizzazione (lato $\leftarrow$)

- 1) costruiamo NT
 - sfruttando quello che sappiamo sulle parole in L e usando T
- Cosa sappiamo sulle parole di L ?
 - $x \in L \leftrightarrow \exists y_x \in \{0,1\}^* : |y_x| \leq |x|^k \wedge T(x, y_x) \text{ accetta} \wedge \text{dtime}(T, x, y_x) \in O(|x|^h)$
 - *dove T, h e k li conosciamo*
 - infatti, le nostre ipotesi sono: “dato L , esistono una macchina di Turing deterministica T e due costanti $h, k \in \mathbb{N}$ tali che ...”
- Allora, costruiamo una macchina NT che opera in due fasi: con input x
 - FASE 1: NT sceglie non deterministicamente una parola binaria y di lunghezza $|y| \leq |x|^k$
 - FASE 2: NT invoca $T(x, y)$ e, se $T(x, y)$ accetta entro $O(|x|^h)$ passi allora NT accetta
- Vediamo, ora, in dettaglio le due fasi

Una nuova caratterizzazione (lato $\leftarrow$)

- 1) costruiamo NT - sfruttando quello che sappiamo sulle parole in L e usando T
- NT opera in due fasi: con input x
 - FASE 1: NT sceglie non deterministicamente una parola binaria y di lunghezza $|y| \leq |x|^k$
 - FASE 2: NT invoca $T(x,y)$ e, se $T(x,y)$ accetta entro $O(|x|^h)$ passi allora NT accetta
- OSSERVAZIONE: $f(n)=n^k$ è una funzione time-constructible – sia T_f il trasduttore che la calcola, in unario, con $\text{dtime}(T_f, n) \in O(n^k)$
- Vediamo, ora, in dettaglio la FASE 1: con (ricordiamo) input x

```
B ← Tf(|x|);           calcola la lunghezza della parola che deve scegliere
i ← 1;
while ( i ≤ B ) do begin
    scegli y[i] nell'insieme {0,1};
    i ← i+1;
end
y ← y[1]y[2] ... y[B];
```

Una nuova caratterizzazione (lato $\leftarrow$)

- 1) costruiamo NT - sfruttando quello che sappiamo sulle parole in L e usando T
- NT opera in due fasi: con input x
 - FASE 1: NT sceglie non deterministicamente una parola binaria y di lunghezza $|y| \leq |x|^k$
 - FASE 2: NT invoca $T(x,y)$ e, se $T(x,y)$ accetta entro $O(|x|^h)$ passi allora NT accetta
- Assumiamo che, se $x \in L$, T accetti entro $c|x|^h \in O(|x|^h)$ passi
anche $g(n) = c n^h$ è una funzione time-constructible – sia T_g il trasduttore che la calcola, in unario, con $\text{dtime}(T_g, n) \in O(c n^h)$
- Vediamo, ora, in dettaglio la FASE 2: con input x e y

```
A ← Tg(|x|);      calcola la lunghezza della computazione che deve simulare
i ← 1;
while ( i ≤ A) do begin
    simula l'esecuzione della i-esima istruzione eseguita da T(x,y);
    if (T è entrata in qA) then accetta e termina;
    else i ← i+1;
end
```

Una nuova caratterizzazione (lato $\leftarrow$)

- 2) dimostriamo che NT accetta L
- Sappiamo che $x \in L \leftrightarrow \exists y_x \in \{0,1\}^* : |y_x| \leq |x|^k \wedge T(x, y_x) \text{ accetta} \wedge \text{dtime}(T, x, y_x) \in O(|x|^h)$
- Se $x \in L$ allora esiste $y_x \in \{0,1\}^* : |y_x| \leq |x|^k \wedge T(x, y_x) \text{ accetta}$
 - allora, **esiste una sequenza di scelte** nella FASE 1 che genera proprio y_x
 - allora, nella FASE 2, $T(x, y_x)$ accetta entro $c |x|^h$ passi
 - allora, anche la computazione deterministica di $NT(x)$ corrispondente alla sequenza di scelte che ha generato y_x accetta
- Questo dimostra che, se $x \in L$, allora $NT(x)$ accetta**
- Se $x \notin L$ allora non esiste alcuna $y_x \in \{0,1\}^* : |y_x| \leq |x|^k \wedge T(x, y_x) \text{ accetta}$
 - allora, **qualunque sia la sequenza di scelte** nella FASE 1 per generare una parola y
 - nella FASE 2, $T(x, y)$ non accetta
 - questo significa che nessuna computazione deterministica di $NT(x)$ (corrispondente ciascuna ad una diversa sequenza di scelte nella FASE 1) accetta
- Questo dimostra che, se $x \notin L$, allora $NT(x)$ non accetta**

Una nuova caratterizzazione (lato $\leftarrow$)

- 3) dimostriamo che, sulle parole di L, NT opera in tempo polinomiale
- FASE 1:
 - $B \leftarrow T_f(|x|);$ calcola la lunghezza della parola che deve scegliere
 $i \leftarrow 1;$
while ($i \leq B$) **do begin**
 scegli $y[i]$ nell'insieme $\{0,1\};$
 $i \leftarrow i+1;$
end
 $y \leftarrow y[1]y[2] \dots y[B];$
- calcolare B richiede $O(|x|^k)$ passi – come già osservato
- il ciclo **while** esegue $|x|^k$ iterazioni, in ciascuna delle quali
 - sceglie un valore in un insieme di dimensione costante – e, quindi, impiega un numero costante di operazioni
 - incrementa di 1 una variabile – e *possiamo assumere* che questo abbia costo costante
 - anche se non è proprio così
- quindi, complessivamente, il ciclo **while** esegue $O(B) = O(|x|^k)$ operazioni
- e la FASE 1 termina in $O(|x|^k)$ operazioni

Una nuova caratterizzazione (lato $\leftarrow$)

- 3) dimostriamo che, sulle parole di L , NT opera in tempo polinomiale
- FASE 2:
 - $A \leftarrow T_g(|x|);$ *calcola la lunghezza della computazione che deve simulare*
 $i \leftarrow 1;$
while ($i \leq A$) **do begin**
 simula l'esecuzione della i -esima istruzione eseguita da $T(x,y);$
 if (T è entrata in q_A) **then accetta** e termina;
 else $i \leftarrow i+1;$
end
 - calcolare A richiede $O(|x|^h)$ passi – come già osservato
 - il ciclo **while** esegue $c|x|^h$ iterazioni, in ciascuna delle quali
 - simula l'esecuzione di una istruzione della computazione $T(x,y)$ – e questo richiede un numero costante di operazioni
 - confronta lo stato in cui è entrata T con q_A – e questo ha costo costante
 - e, se non è q_A , incrementa di 1 una variabile – e possiamo assumere che questo abbia costo costante
 - quindi, complessivamente, il ciclo **while** esegue $O(|x|^h)$ operazioni
 - e la FASE 2 termina in $O(|x|^h)$ operazioni

Una nuova caratterizzazione (lato $\leftarrow$)

- Riassumendo:
- Partendo dall'**ipotesi**
 - **esistono una macchina di Turing deterministica T e due costanti $h, k \in \mathbb{N}$ tali che,**
 - **per ogni $x \in \Sigma^*$,**
 - **$x \in L \leftrightarrow \exists y_x \in \{0,1\}^* : |y_x| \leq |x|^k \wedge T(x, y_x) \text{ accetta} \wedge \text{dtime}(T, x, y_x) \in O(|x|^h)$**
- abbiamo costruito una macchina NT
 - che accetta L
 - e tale che, per ogni $x \in L$, $\text{ntime}(NT, x) \in O(|x|^k + |x|^h)$
- Ossia, partendo dall'**ipotesi abbiamo dimostrato che $L \in NP$**
- Il **teorema 9.1** è completamente dimostrato

Due definizioni equivalenti

- Il Teorema 9.1 è una **caratterizzazione alternativa** della classe NP
 - come stiamo ripetendo dall'inizio della lezione
- Una caratterizzazione alternativa che, a seguito della discussione che abbiamo portato avanti dimostrando il Teorema 9.1, possiamo esprimere nel modo seguente

un problema (decisionale) Γ è in NP se e soltanto se le sue istanze sì ammettono certificati verificabili in tempo polinomiale

- Ma che significa **caratterizzazione alternativa**?
- Significa che possiamo dimostrare che un problema (decisionale) Γ è in NP in due modi differenti
- **MODO 1) progettiamo un algoritmo non deterministico che accetta Γ e dimostriamo che quell'algoritmo accetta le istanze sì di Γ in tempo polinomiale**
- **MODO 2) dimostriamo che le istanze sì di Γ ammettono certificati di lunghezza polinomiale nella loro dimensione, e che quei certificati sono verificabili in tempo polinomiale**

Come si usa la caratterizzazione alternativa

- Lo abbiamo già visto! Nel primo lucido di questa lezione ...
- Ogni problema
 - il cui predicato ha la forma $\pi(x, S(x)) = \text{esiste } y \in S(x) \text{ tale che } \eta(x, y)$
 - in cui la lunghezza di un elemento y di $S(x)$ è polinomiale in $|x|$
 - in cui la verifica che y soddisfi il predicato η , richiede tempo deterministico polinomiale in $|x|$**
- appartiene ad NP
- Utilizzando la notazione che abbiamo imparato in questa lezione,
un elemento y di $S(x)$ è un **certificato** per l'istanza x del problema
- e dire che *sceglierlo richiede tempo non deterministico polinomiale in $|x|$* è
equivalente a dire **che ha lunghezza polinomiale in $|x|$**
 - infatti, ricordate la FASE 1 della macchina NT che decide un linguaggio che soddisfa le ipotesi del Teorema 9,1 (lucido 18):
 - dobbiamo scegliere non deterministicamente ciascuno dei suoi bit
 - e se il loro numero fosse più che polinomiale in $|x|$ non ce la faremmo in tempo polinomiale!

Come si usa la caratterizzazione alternativa

- Ogni problema
 - il cui predicato ha la forma $\pi(x, S(x)) = \text{esiste } y \in S(x) \text{ tale che } \eta(x, y)$
 - in cui la lunghezza di un elemento y di $S(x)$ è polinomiale in $|x|$
 - in cui **la verifica che y soddisfi il predicato η , richiede tempo deterministico polinomiale in $|x|$**
- appartiene ad NP
- **ATTENZIONE**: è fondamentale che **la verifica che il certificato soddisfi il predicato η , richieda tempo deterministico polinomiale in $|x|$**
- se questo non è vero, non possiamo affermare che il problema appartenga ad NP!!!
- E, se non controllate che questa seconda condizione sia soddisfatta, potete incorrere in **errori clamorosi!**
 - come andiamo ad illustrare ...

ATTENZIONE!!!

- Consideriamo il seguente problema - che assomiglia a SAT
- dati due insiemi X_1 e X_2 di variabili booleane ed un predicato f , in forma congiuntiva normale, definito sulle variabili in $X_1 \cup X_2$, decidere se esiste una assegnazione a_1 di valori in {vero, falso} alle variabili in X_1 tale che per ogni assegnazione a_2 di valori in {vero, falso} alle variabili in X_2 , risulti $f(a_1(X_1), a_2(X_2)) = \text{vero}$
- Questo problema prende il nome di 2QBF, ed è così formalizzato:
 - $\mathfrak{I}_{2\text{QBF}} = \{ \langle X_1, X_2, f \rangle : X_1 \cup X_2 \text{ è un insieme di variabili booleane} \wedge f \text{ è un predicato in CNF su } X_1 \cup X_2 \}$
 - $\mathcal{S}_{2\text{QBF}}(X_1, X_2, f) = \{ a_1 : X_1 \rightarrow \{\text{vero}, \text{falso}\} \}$
 - $\pi_{2\text{QBF}}(X_1, X_2, f, \mathcal{S}_{3\text{SAT}}(X_1, X_2, f)) = \exists a_1 \in \mathcal{S}(X_1, X_2, f) : \forall a_2 : X_2 \rightarrow \{\text{vero}, \text{falso}\} [f(a_1(X_1), a_2(X_2)) = \text{vero}]$
- Un **certificato** per una istanza $\langle X_1, X_2, f \rangle$ è una assegnazione di verità a_1 per le variabili in X_1 – ed ha **lunghezza** $|X_1|$
 - tranquillamente **polinomiale in** $|\langle X_1, X_2, f \rangle|$!
- Allora, possiamo concludere che 2QBF appartiene ad NP?

ATTENZIONE!!!

- Il problema 2QBF è così formalizzato:
 - $\mathfrak{I}_{2QBF} = \{ \langle X_1, X_2, f \rangle : X_1 \cup X_2 \text{ è un insieme di variabili booleane} \wedge f \text{ è un predicato su } X_1 \cup X_2 \}$
 - $\mathcal{S}_{2QBF}(X_1, X_2, f) = \{ a_1 : X_1 \rightarrow \{\text{vero}, \text{falso}\} \}$
 - $\pi_{2QBF}(X_1, X_2, f, \mathcal{S}_{3SAT}(X_1, X_2, f)) = \exists a \in \mathcal{S}(X_1, X_2, f) : \forall a_2 : X_2 \rightarrow \{\text{vero}, \text{falso}\} [f(a_1(X_1), a_2(X_2)) = \text{vero}]$
- Un **certificato** per una istanza $\langle X_1, X_2, f \rangle$ è una assegnazione di verità a_1 per le variabili in X_1 – ed ha **lunghezza polinomiale in $|\langle X_1, X_2, f \rangle|$**
- Allora, possiamo concludere che 2QBF appartiene ad NP?
- COL CAVOLO!**
- In questo caso, il predicato η è: $\forall a_2 : X_2 \rightarrow \{\text{vero}, \text{falso}\} [f(a_1(X_1), a_2(X_2)) = \text{vero}]$
- Quindi, verificare che una assegnazione di verità a_1 per le variabili in X_1 soddisfi η significa verificare che, **comunque scegliamo $a_2 : X_2 \rightarrow \{\text{vero}, \text{falso}\}$** , risulta $f(a_1(X_1), a_2(X_2)) = \text{vero}$
- E il numero di assegnazioni di verità a_2 alle variabili in X_2 è $2^{|X_2|}$
- Col cavolo** che riusciamo a verificare η in tempo **polinomiale in $|\langle X_1, X_2, f \rangle|$** !