Abbiamo mostrato PATH $\in$ SPACE$(\log^2 n)$

Facciamo vedere che la stessa dimostrazione implica PSPACE = NPSPACE.

THM $NL \subseteq P$, SPACE$(\log^2 n)$.

DIM. Applicare gli algoritmi che abbiamo studiato per PATH usando un grafo particolare.

Sia $A \in NL$, allora $\exists$ NTM N t.c. $L(M) = A$ e N ha complessità di spazio $O(\log n)$.

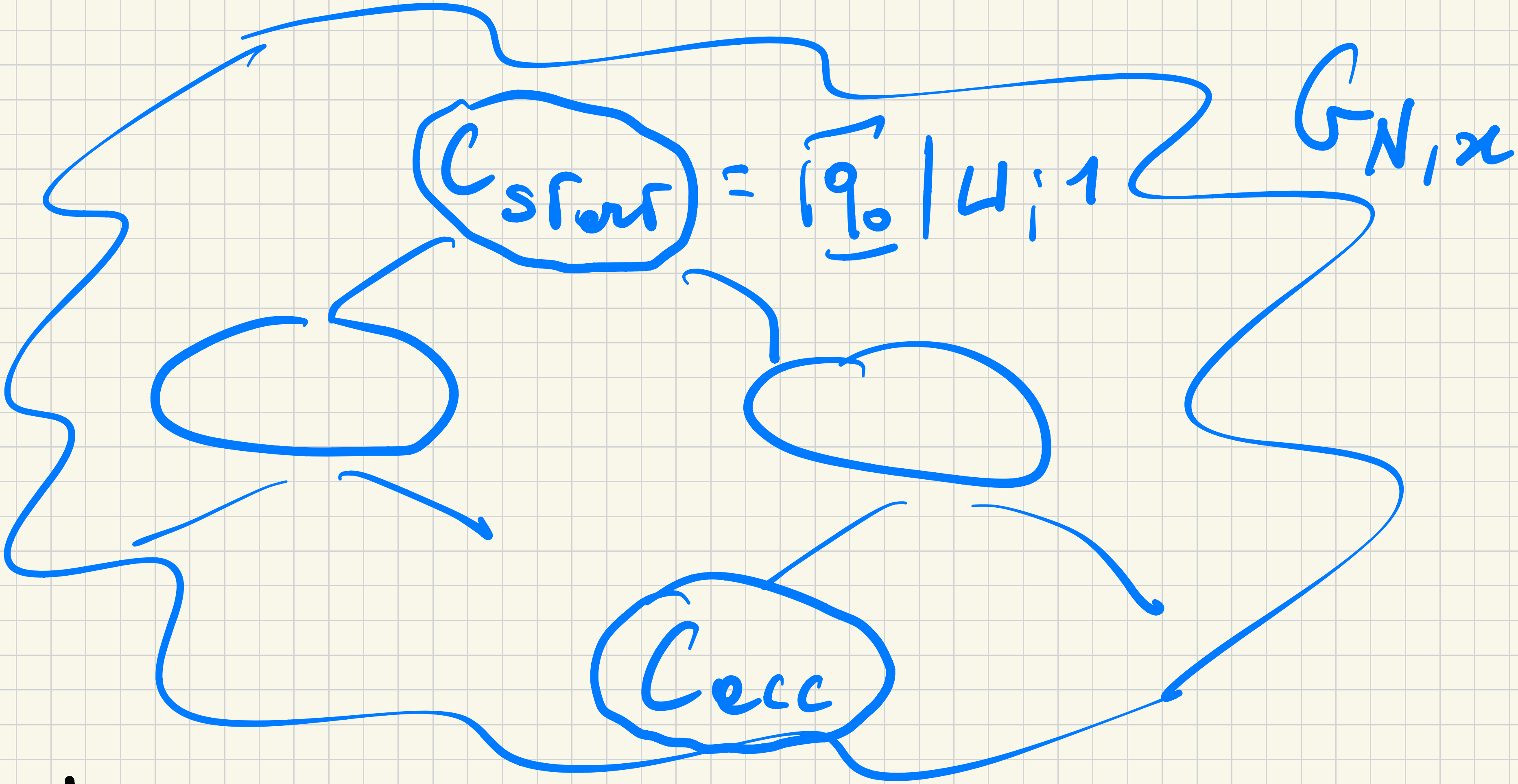
Ricordiamo le configurazioni di N su x.

$C = $ work tape $\boxed{q_7}$ pos. input

Siccome la complessità di spazio di N è $f(n) = O(\log n)$ allora:

$$\# \text{config.} = 2^{O(f(n))} = 2^{O(\log n)} = poly(n)$$

Osservazione: Una computazione non-det. di N su x ha associato un grafo $G_{N,x} = (V, E)$ i cui vertici sono configurazioni e c'è un arco tra $(C, C') \in E$ sse C' segue da C utilizzando una delle scelte non-det. di N.



$\Rightarrow N(x) = ACC$ sse $\exists\ C_{start} \rightsquigarrow C_{acc}$

in $G_{N,x}$. Per risolvere questo una

rispetto di |x|. C'è una unica configurazione = stato accettante C_{acc}:

$$C_{acc} = \overline{|q_{acc}|} \sqcup ; 1$$

Wlog. N prima di accettare cancella il contenuto del nastro di lavoro e muove la testina a sx. Questo non modifica la complessità di spazio.

A questo punto:

- $A \in NL \Rightarrow A \in P$. Basta considerare la TM M che su input <N, x> scrive esplicitamente sul nastro di lavoro in poly(n) time $G_{N,x}$, C_{start}, C_{acc} poi controlla $\langle G_{N,x}, C_{start}, C_{acc} \rangle \in PATH$ in tempo poly(n).

- $A \in NL \Rightarrow A \in SPACE(\log^2 n)$.

Stessa cosa di prima, ma ora M non può scrivere $G_{N,x}$ sul nastro di lavoro. Fortunatamente, si può verificare che l'algoritmo che abbiamo studiato per

PATH $\in$ SPACE$(\log^2 n)$ non deve conoscere $G_{N,x}$ per intero.

In realtà basta contare $|V| = n$ e enumerare tutti i nodi alla ricerca del MIDPOINT in $O(\log n)$ spazio.

Inoltre dato (C, C') devo poter controllare in $O(\log n)$ spazio se $(C, C') \in E$. Basta recuperare dato $C = \text{work}\boxed{q}|\text{tape}, i$ il carattere x_i

e controllare cosa dice $\delta_N(q, x_i)$ $\blacksquare$

In generale:

THM $NSPACE(f(n)) \subseteq DTIME(2^{O(f(n))})$

$SPACE(f^2(n))$.

COR Sia $f(n) = n^k$, $k > 0$.

$$\bigcup_k NSPACE(n^k) \subseteq \bigcup_k DTIME(2^{O(n^k)})$$

$$\bigcup_k SPACE(n^{2k})$$

$NPSPACE \subseteq EXP$, $PSPACE$ ∎

NL-COMPLETEZZA

Voglio mostrare che in qualche modo PATH è completo.

DEF B è NL-completo se:

(I) $B \in NL$

(II) $\forall A \in NL$, A "$\leq$" B ???

$\leq_L$

$\leq_m$

Devo stare attento, perché voglio far sì che:

$$A \leq B,\ B \in L \Rightarrow A \in L$$
$$B \in NL \Rightarrow A \in NL$$

Ma non posso usare "$\leq_m^p$" perché la riduzione può distruggere la complessità di spazio.

DEF $A \leq_m^L B$ se $\exists\ R : \{0,1\}^* \to \{0,1\}^*$ calcolabile in $O(\log n)$ spazio t.c. $\forall x \in \{0,1\}^*$, $x \in A$ sse $R(x) \in B$.

Difficoltà intrinseca: $R(x)$ può essere poly(n) e R non ha spazio per scriverlo.

TM con output: aggiungo nastro di output "WRITE ONCE".

Vedremo perché questa definizione va bene.

THM Se P, Q sono funzioni calcolabili in log-space, allora $R(x) = Q(P(x))$ pure calcolabile in log-space.

Dim. L'abbiamo visto per poly-time.

Ora è più difficile perché $R(x)$ non può calcolare $f(x)$ e scriverlo sul nastro di lavoro / output.

Devo progettare $R(x)$ che ha x su nastro input, sposta $\log |x|$ su nastro lavoro e deve scrivere $Q(f(x))$ su nastro output "write-once".

$R(x)$ sarà inefficiente in tempo e "pretenderà" di avere $y = f(x)$ su nastro read-only:

- Tiene traccia della posizione i sul nastro di input di Q usando $O(\log n)$ spazio.
- Lancia la computazione di $P(x)$ solo per recuperare y_i.
- Al prossimo passo ricomincia dall'inizio.

$\boxtimes$

Conseguente:

Cor $A \leq_m^L B$, $B \in L \Rightarrow A \in L$

COR $A \leq_m^L B,\ B \in NL \Rightarrow A \in NL.$

COR $A \leq_m^L B,\ B \leq_m^L C \Rightarrow A \leq_m^L C.$

Possiamo anche dimostrare che PATH è NL-completo.

THM PATH è NL-completo.

DIM. Da una parte è facile vedere che PATH è NL-hard. Sia $A \in NL$: $\exists$ NTM N con spazio $O(\log n)$ t.c. $N(x)$ accetta sse $x \in A$.

già abbiamo visto : $N(x) = Acc$ sse
$\exists$ un $G_{N,x}$ il cammino $C_{start} \to C_{acc}$.
Considero $R : \{0,1\}^* \to \{0,1\}^*$ che dato
x ritorna $\langle G_{N,x}, C_{start}, C_{acc} \rangle$.
Così :

$x \in A$ sse $N(x) = Acc$
sse $R(x) \in PATH$

$R(x)$ scrive $G_{N,x}$, C_{start}, C_{acc}
esplicitamente nel output usando $O(\log n)$

spazio sul nastro di lavoro.

Inoltre PATH $\in$ NL:

*) Su input G, s, t se $s = t$ accetta.

*) Calcola in $O(\log n)$ spazio $n = |V|$

*) curNode $= s$. For $i = 1, \dots, n$

- Indovina non-det. un nodo $u \in V$
- Se $(curNode, u) \in E$, allora curNode $= u$. Se curNode $= t$, ACCETTA.

- Se $(currNode, u) \notin E$, allora RIFIUTA.

*) RIFIUTA

Alcuni fatti importanti che non dimostreremo:

- Ci sono anche linguaggi P-completi e PSPACE-completi

  * CIRCUIT-EVAL è P-completo.
    Questo richiede di mostrare che Cook-Levin funziona sotto log-space reductions.

* TQBF è PSPACE COMPLETO

$SAT = \{ \langle \phi \rangle : \phi$ è soddisfacibile $\}$

↳ NP-completo.

$= \{ \langle \phi \rangle : \exists x_1 \dots x_n$ t.c. $\phi(x_1 \dots x_n) = 1 \}$

$TAUT = \{ \langle \phi \rangle : \forall x_1 \dots, x_n$ su cui $\phi(x) = 1 \}$

↳ coNP-completo

$TQBF = \{ \langle \phi \rangle : Q_1 x_1\, Q_2 x_2, \dots Q_n x_n$

t.c. $\phi(x) = 1 \}$

$Q_i \in \{\exists, \forall\}$

$\hookrightarrow$ PSPACE - completo.

- Abbiamo visto che non sappiamo se $NP \neq coNP$ ($\Rightarrow P \neq NP$).

Però $NL = coNL$. In altre parole $\overline{PATH} \in NL$.

TEOREMI DI GERARCHIA

Questi implicano ad es. $P \neq EXP$.
Vale sia per spazio che per tempo.
Ad alto livello:

THM Se "$t_1(n) << t_2(n)$" (e.g. $t_1(n) = n^2$ e $t_2(n) = n^6$). Allora esiste $L \in DTIME(t_2(n))$ ma

$$L \notin DTIME(t_1(n))$$

Fissiamo una codifica $[\cdot]_{TM} : \Sigma^* \to \{TM\}$

Voglio: $\forall$ TM M $\exists x \in \Sigma^*$ t.c.

$[x]_{TM} = M$. Se x non è una TM valida associo TM di default.

Costruisco una TM $D(x)$:

- Su input x, sia $M = [x]_{TM}$.
- Simulo $M(x)$ per $t_{1.5}(n)$ passi dove $t_1 << t_{1.5} << t_2$ (un po' più di t_1).

- Diagonalizzo: $M(x) = ACC \Rightarrow D(x) = REJ$ e $M(x) = REJ \Rightarrow D(x) = ACC$.
(Se $M(x)$ non ha ancora terminato allora $D(x) = ACC$.)

Sia $L = L(D)$. Sicuramente $L(D) \in DTIME(t_2(n))$.

Mostriamo: $\forall$ TM Q con tempo $\leq t_1(n)$ Q non può decidere L.
($L(D) \notin DTIME(t_1(n))$.)

Sia x t.c. $Q = [x]_{TM}$ e supponiamo $Q(x)$ termina in $t_1(|x|)$ passi ovvero $<< t_{1.5}(n)$ passi.

Conseguenza: $D(x)$ finisce la simulazione e poi fa l'opposto $\Rightarrow Q(x) \neq D(x)$ ovvero $Q(x)$ sbaglia a decidere "$x \in L$". Ovvero $L \notin DTIME(t_1(n))$. ∎

Ci sono alcune cose da chiarire per stabilire che significa "$t_1 << t_2$".

— Prima questione (tecnica): $DTIME(t_1)$ consente tempo $c \cdot t_1(n)$ per ogni $c \in \mathbb{N}$.
Per questo ho bisogno che

$$t_{1.5}(n) = \omega(t_1(n))$$

Ma questo è una nozione asintotica, solo vera per n abb. grande.
In altre parole: non basta un singolo x t.c. $Q(x) \neq D(x)$ perché x potrebbe essere troppo corto.

Deve essere vero per un numero infinito di x ; $Q = [x]_{TM}$ per infiniti x. Risolvo questo aggiungendo alla codifica un numero arbitrario di junk "\$".

- Altra difficoltà: Necessità di TM universale che in tempo $O(t_2(n))$ può simulare una TM M di tempo $O(t_{1.5}(n))$ gestendo il suo alfabeto, i suoi nastri, etc.

Made with Goodnotes

FATTO Esiste una TM universale con tempo $O(T \log T)$ dove T è il tempo di M.

$\Rightarrow t_2 = O(t_{1.5} \log t_{1.5})$

- Tecnicalità: t_2, t_1 devono essere tempo-costruibili, altrimenti non posso contare fino a $t_1(n)$ o $t_2(n)$.

Abbiamo ottenuto:

T.H.T. Siano $t_1(n) \geq n$ e $t_2(n)$ t.c.

$t_2(n) = \omega(t_1(n) \log t_1(n))$ e tempo costruttibile. Allora: $\exists L \in DTIME(t_2(n))$ ma $L \notin DTIME(t_1(n))$.

S.H.T. Sia $s_1(n) \geq \log n$ e $s_2(n)$ t.c. $s_2(n) = \omega(s_1(n))$. Allora: $\exists L$ t.c. $L \in SPACE(s_2(n))$ ma

$L \notin SPACE(s_1(n))$