Concludiamo il nostro percorso nella complessità di spazio dimostrando alcuni risultati classici:

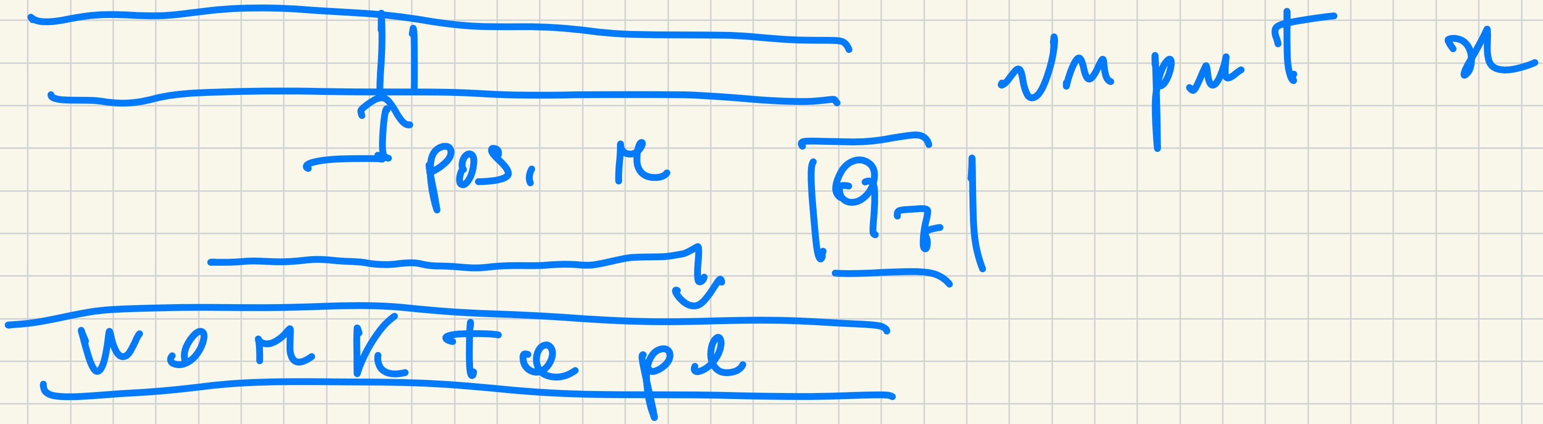
- NPSPACE = PSPACE.
- PATH è NL-completo.
- Esistono problemi PSPACE completi.
- NL = coNL
- Teorema di gerarchia per spazio / tempo.

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

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

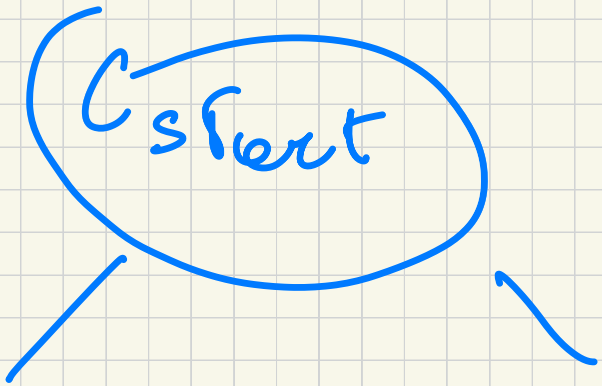
Per dimostrare il teorema ricordiamo il concetto di configurazione:

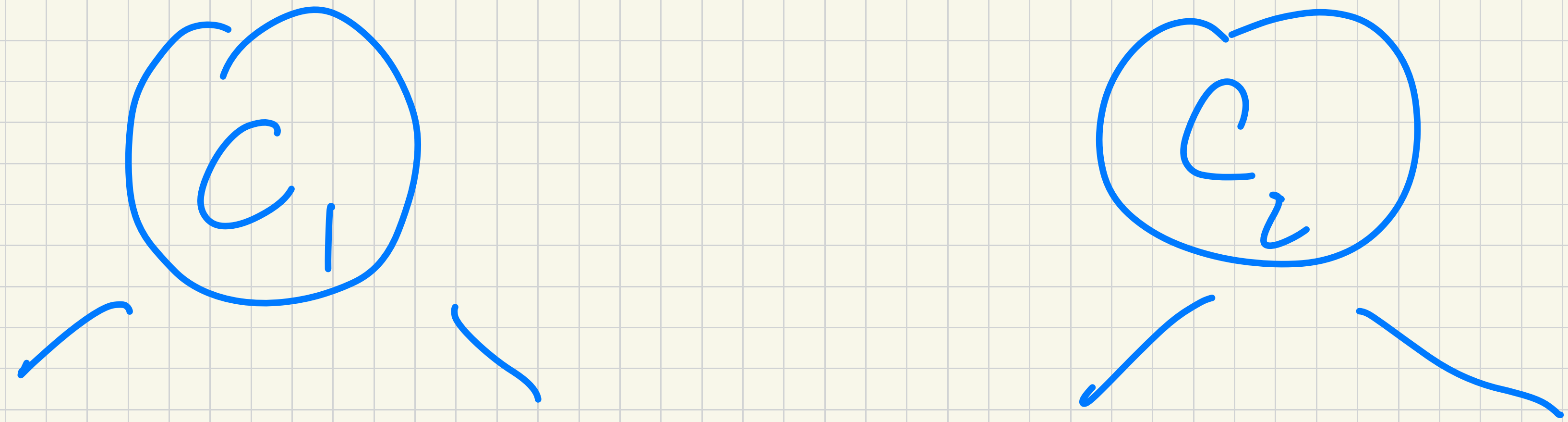
$$C = \text{workta} \, |q_7| \, \text{pe} \, ; \, x$$



Sappiamo che ci sono al più $\doteq 2^{O(\log n)}$ $= O(n^k)$ configurazioni.

Associamo alla computazione di N su x un grafo $G_{N,x}$ i cui nodi sono le configurazioni C come sopra. Gli archi:
$(C, C') \in E$ sse la configurazione C' può seguire dalla config. C secondo una delle transizioni non-det. dettate da δ.





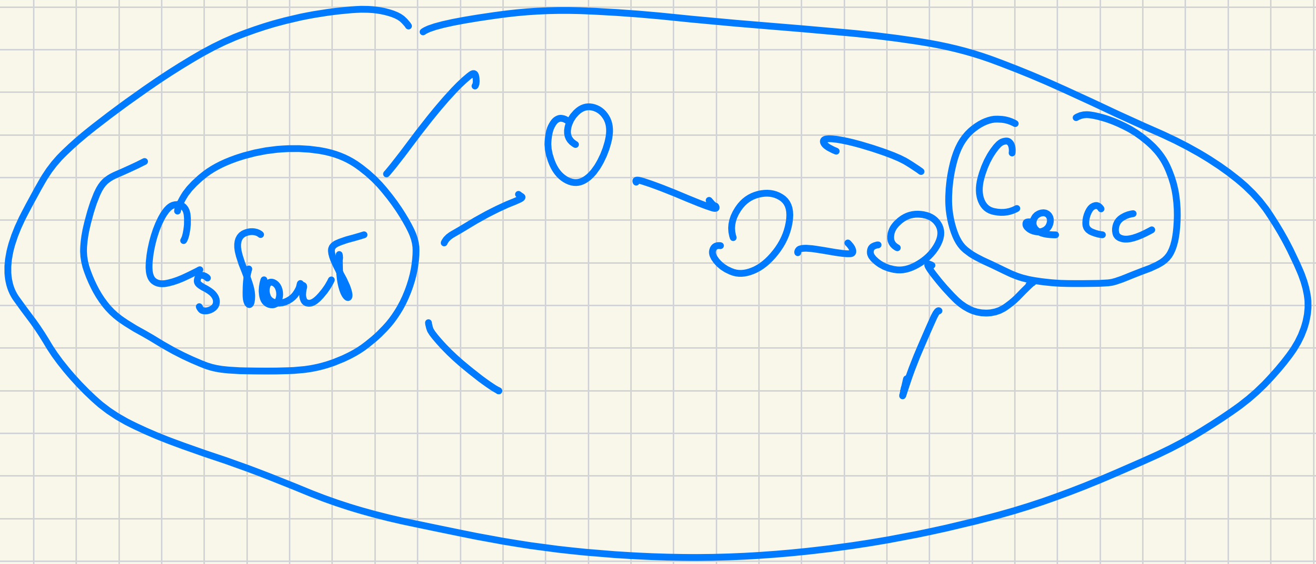
$$C_{start} = \boxed{q_0} \, \sqcup \, ; \, 1$$

$$C_1 = e \, \boxed{q_3} \, \sqcup \, ; \, 2$$

Ci sono configurazioni speciali, così dette accettanti, ovvero che contengono stato q_{acc}. Possiamo fare in modo che ci sia una UNICA configurazione accettante:

$$C_{ecc} = \lceil \underline{q_{ecc}} \rceil \sqcup ; 1$$

Ovvero, essendo che $N(x)$ prima di accettare cancella il contenuto del nastro di lavoro e muove le due testine tutte a sx.
Questa trasformazione preserva la complessità di spazio di N.



$\Rightarrow$ NPSPACE $\subseteq$ EXP, PSPACE

Siccome PSPACE $\subseteq$ NPSPACE

$\Rightarrow$ PSPACE $=$ NPSPACE.

Next: Vogliamo ora vedere che PATH è in qualche senso COMPLETO.

DEF B è NL-completo se:

(i) $B \in NL$

(ii) $\forall A \in NL : A \leq B$

$\hookrightarrow$???

La riducibilità che vogliamo considerare non può essere $\leq_m^P$ perché questo può non preservare la complessità di spazio. In particolare, la def. di $\leq$ dovrebbe garantire: Se $A \leq B$, allora $B \in L \Rightarrow A \in L$.

Definiamo una nuova riducibilità: $\leq_m^L$ riducibilità in spazio logaritmico.

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

$$\forall x \in \{0,1\}^*, \quad x \in A \text{ sse } R(x) \in B.$$

Made with Goodnotes

Siccome R(x) potrebbe avere dimensione poly(n) non possiamo scrivere l'output sul nastro di lavoro. Aggiungiamo al modello nastro di output che può essere scritto una volta solo.

THM PATH è NL-completo.

Dim. (i) PATH $\in$ NL; l'abbiamo già dimostrato.

(ii) $\forall A \in NL$ devo far vedere $A \leq_m^L$ PATH.

Sia N la NTM t.c. $L(N) = A$ e N ha comp. di spazio $O(\log n)$. Costruisco $R: \{0,1\}^* \to \{0,1\}^*$ che dato $x \in \{0,1\}^*$

informe $G_{N,x}$, C_{start}, C_{acc} come definiti nella prova del THM di SAVITCH.
In particolare:

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

sse $\exists\ C_{start} \rightsquigarrow C_{acc}$ in $G_{N,x}$

sse $R(x) = (G_{N,x}, C_{start}, C_{acc})$
$\in$ PATH.

In particolare $R(x)$ può scrivere $G_{N,x}$ usando le matrice di adiacenza sul nastro di output richiede $O(\log n)$ spazio sul nastro di lavoro. Questo richiede di enumerare tutte le configurazioni che è fattibile in spazio $O(\log n)$. Sempre con le stesse complessità di spazio posso enumerare config. C' e controllare se $(C, C') \in E$ $\blacksquare$

Vediamo che la def. di log-space reduction soddisfa le proprietà che ci aspettiamo:

- $A \leq_m^L B$: $B \in L \Rightarrow A \in L$

$B \in NL \Rightarrow A \in NL$

- $A \leq_m^L B$, $B \leq_m^L C$: $A \leq_m^L C$.

queste proprietà seguono dal:

THM. Se P, Q sono computabili in $\log(n)$ spazio, allora lo è anche

$$R(x) = Q(P(x)).$$

Dim. La difficoltà è che $P(x)$ può avere lunghezza polinomiale.

Posso dire: P ha tempo n^p e Q ha tempo n^q, ovvero entrambi polinomiali.

Devo definire $M(x)$ t.c. scrive sul nastro di output $R(x)$ usando $O(\log n)$ spazio sul nastro di lavoro. Non posso però scrivere $y = P(x)$ sul nastro di lavoro e poi calcolare $R(x) = Q(y)$.

La TM M tiene traccia della posizione corrispondente all'input tape di Q; questo richiede $O(\log n)$ spazio. Tutto quello che M deve fare è determinare $y[i]$ e simulare un passo della computazione di Q. Per farlo usando $O(\log n)$ spazio sarà tempo inefficiente: Ricalcola $P(x)$ fino a che non otteniamo $y[N]$ e butta via tutto il resto. Questo permette di simulare un passo di computazione di Q.

Ripeti per ogni passo, riusando lo spazio. ∎

COR $A \leq_m^L B$: $B \in L \Rightarrow A \in L$.

DIM. Sia P la log-space riduzione da A a B e Q la TM con spazio log. t.c. $Q(x) = Acc$ sse $x \in B$. Poi usiamo il teorema di sopra ∎

COR $A \leq_m^L B$: $B \in NL \Rightarrow A \in NL$

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

Finora abbiamo visto esempi di problemi:

- NP - completi
- NL - completi.
- coNP - completi

Vediamo ora esempi di problemi:

- P - completi
- PSPACE - completi.

Def C è P-completo se:

(i) $C \in P$

(ii) $\forall A \in P$, $A \leq C$

$\hookrightarrow$??? $\leq_m^L$

Perché questo è utile: se è vero che $C \in L$, allora $P \subseteq L$. Questo è un problema aperto.

Un problema che è P-completo:

$$\text{CIRCUIT-EVAL} = \{ \langle C, x \rangle : C(x) = 1 \}$$

THM. CIRCUIT-EVAL è P-completo.

Dim. Ovviamente CIRCUIT-EVAL $\in$ P.

Poi bisogna far vedere: $\forall A \in P$ allora

$A \leq_m^L$ CIRCUIT-EVAL.

Questa riduzione richiede di convertire una TM in un circuito $C_M(x)$ t.c.

$\hookrightarrow M(x)$

$$M(x) = Acc \text{ sse } C_M(x) = 1$$

Si può fare riutilizzando la dimostrazione del teorema di Cook-Levin. Ma non lo vedremo

Vediamo invece qualche dettaglio su PSPACE completezza: TQBF (Totally Quantified Boolean formula).

SAT $= \{ \langle \phi \rangle : \phi$ una formula soddisfacibile $\}$.

↳ NP - completo

$(\exists x_1, \exists x_2, \dots, \exists x_n$ t.c. $\phi(x) = 1)$.

TAUT $= \{ \langle \phi \rangle : \forall x_1, \forall x_2, \dots, \forall x_n$ t.c. $\phi(x) = 1 \}$

↳ coNP - completo

$TQBF = \{ \langle \phi \rangle : Q_1 x_1, Q_2 x_2, \dots, Q_m x_m$

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

Ad es. $Q_1 = \forall$, $Q_2 = \exists$, $Q_3 = \exists$, ...

THM $TQBF \in PSPACE$.

DIM. Facciamo algoritmo ricorsivo:

$IsTrue?(Q_1 x_1, Q_2 x_2, \dots, Q_m x_m, \phi(x))$:

- Se $m = 0$, ϕ è fatta solo di costanti e ritorna la valutazione di ϕ.
- Else if $Q_1 = \exists$, allora ritorna:

$\text{IsTrue?}(Q_2 x_2, \ldots, Q_n x_n ; \phi(0, x_2, \ldots, x_n))$

$$\vee$$

$\text{IsTrue?}(Q_2 x_2, \ldots, Q_n x_n, \phi(1, x_2, \ldots, x_n))$

- Else if $Q_1 = \forall$, alors retourne :

$\text{IsTrue?}(Q_2 x_2, \ldots, Q_n x_n, \phi(0, x_2, \ldots, x_n))$

$$\wedge$$

$\text{IsTrue?}(Q_2 x_2, \ldots, Q_n x_n, \phi(1, x_2, \ldots, x_n))$

L'algoritmo è sicuramente corretto. La profondità della ricorsione è n e lo spazio per ogni chiamata ricorsiva è lineare nella dimensione dell'input.
Spazio totale $O(n^2)$ ovvero polinomiale.