Per esempio usiamo, suggeriscono il seguente "modello mentale" per una TM con spazio $\log(n)$:

- Posso assumere $n = |x|$ è noto.
- Posso gestire $O(1)$ variabili contatore $i, j, k \ldots$ da 1 a $\text{poly}(n)$.
- Posso prelevare simboli di input
- Posso eseguire semplici operazioni aritmetiche (tempo $n+1-i$)

Esempio ulteriore: Il prodotto $a \cdot b = c$.

Si può fare un L:

- Inizializzo $c = 0$
- For $i = 1, \dots, b$

  $c = c + 2$

Non è efficiente in tempo ma usa poco spazio (N.L. $O(\log n)$).

Ma ci sono problemi che non sappiamo essere in L? Sì. Ad esempio, PATH. L'algoritmo che abbiamo studiato deve memorizzare almeno 1 bit per ogni nodo marcato e quindi risulta in spazio $\Omega(n)$.

Vedremo che PATH $\in$ SPACE$(\log^2 n)$.

Altro esempio: 3SAT $\in$ PSPACE. Posso provare tutte gli assegnamenti una alla volta usando spazio poly(n) e riusando questo spazio. Pertanto, in maniera simile, NP $\subseteq$ PSPACE.
Vediamo la relazione tra spazio e tempo.

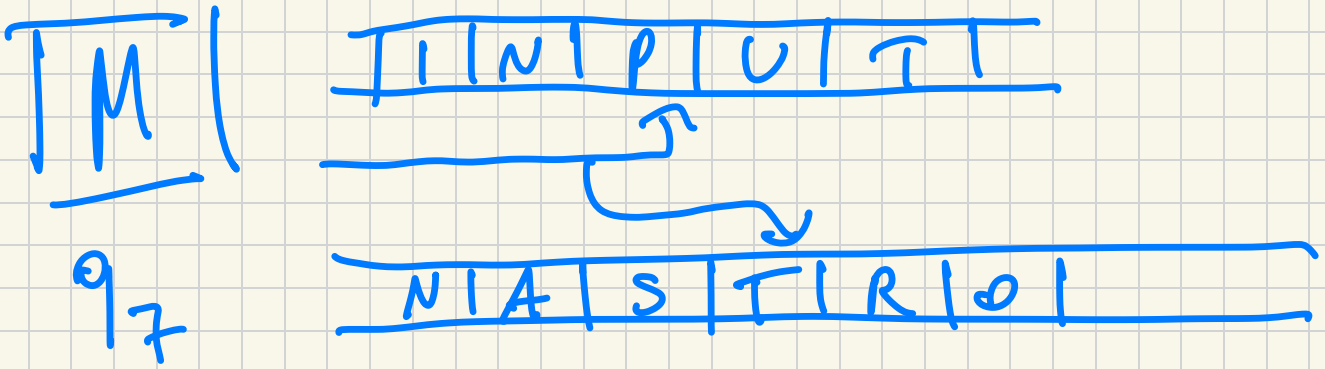
TEO DTIME(f(n)) $\subseteq$ SPACE(f(n))

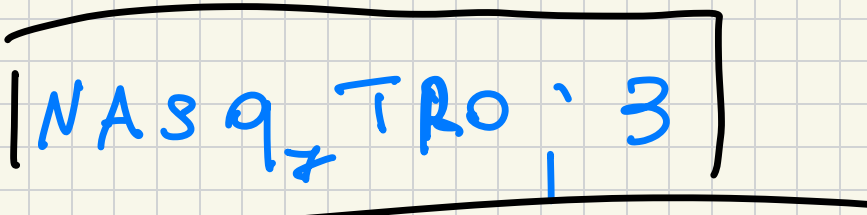
DIM. Perché una TM di tempo O(f(n)) può utilizzare al più O(f(n)) celle di memoria

TEO Per ogni $f(n) \geq \log(n)$, SPACE(f(n)) $\subseteq$ DTIME($2^{O(f(n))}$).

DIM. Intuizione: $2^{O(f(n))}$ è il numero massimo

di configurazioni per una TM di spazio $O(f(n))$.
Ricordiamo il concetto di configurazione





Osservazione: M è un decisore deterministico e quindi non ripete mai una configurazione.
Per questo motivo, la complessità di tempo di M è al più il # di configurazioni:

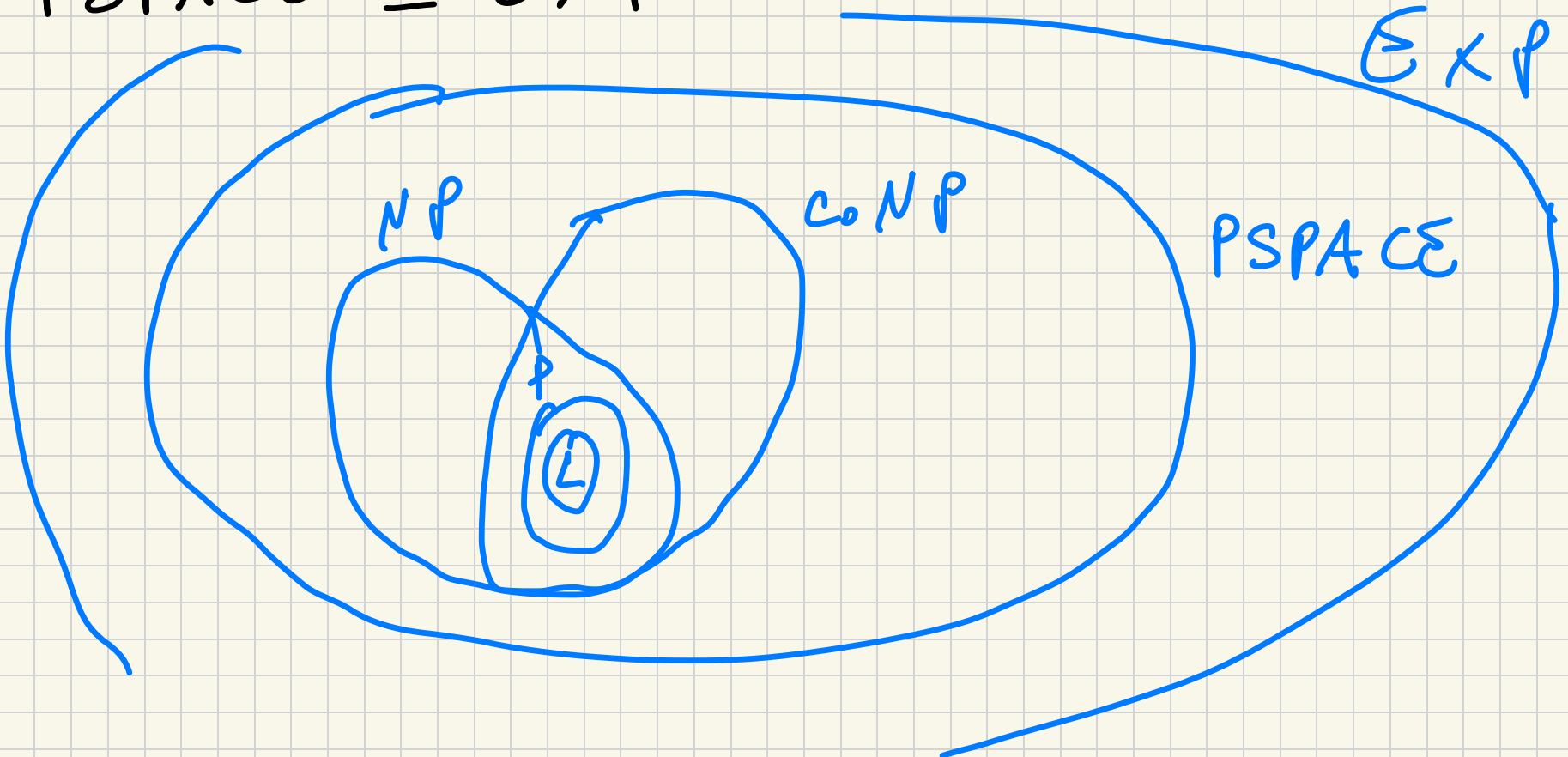
$$\#\text{ configurazioni} \leq |\Gamma|^{f(n)} \cdot |Q| \cdot n.$$

$$\leq |\Gamma|^{f(n)} \cdot |Q| \cdot 2^{f(n)}$$

$$= 2^{O(f(n))}$$

perché $|\Gamma|, |Q| \in O(1)$ ∎

Cor $PSPACE \subseteq EXP$



$$L \subseteq P \subseteq PSPACE \subseteq EXP$$

e $P \neq EXP$

(T. H. T.)

$\Rightarrow$ $P \neq PSPACE$ oppure $PSPACE \neq EXP$

L'altra cosa importante da capire è il ruolo del NON-DETERMINISMO. Vedremo che nel contesto di spazio il non-det. non fa molta differenza.

Il seguente risultato è cruciale:

TEO PATH $\in$ SPACE$(\log^2 n)$.

DIM. L'idea è di usare un algoritmo ricorsivo. La nostra TM M ha sul nastro di input il grafo $G = (V, E)$ e i due nodi s, t. Come detto possiamo assumere che M calcoli $M = \#V$ in spazio $O(\log n)$.

Se c'è un cammino tra s e t questo ha dimensione $\leq M$. L'algoritmo intuitivamente prova e trovare un nodo di mezzo tra s e t t.c. $s \rightsquigarrow u \rightsquigarrow t$. Ripete questo processo ricorsivamente.

PATH ? (x, y, k) : Retourne ACC ssi

$\exists$ x ⇝ y de longueur $\leq 2^k$.

La TM M, elle fait exécuter :

PATH ? (s, t, $\lceil \log n \rceil$)

PATH ? (x, y, k) :

If k = 0

Return ACC ssi x = y $\vee$ (x, y) $\in E$

Else (k > 0)

For each w $\in V$

If PATH ? (x, w, k-1) $\wedge$ PATH ? (w, y, k-1)

Return Acc

REJECT.

La correttezza è immediata. Per quanto riguarda la complessità di spazio, notiamo che questo può essere ricorsivo per ogni modo w; e per ogni w è necessario memorizzare un # costante di variabili, il che richiede spazio logaritmico.

La profondità della ricorsione è $O(\log n)$

Lo spazio è $O(\log^2 n)$ ∎

Vediamo le conseguenze di questo teorema.

DEF $NSPACE(f(n)) = \{ L : \exists \text{ NTM } N$ t.c. $L(N) = L$ e N ha spazio $O(f(n))\}$

dove la complessità di spazio per una NTM è il massimo su tutti i rami di computazione.

$$NPSPACE = \bigcup_k NSPACE(n^k)$$

$$NEXPSPACE = \bigcup_k NSPACE(2^{n^k})$$

$$NL = NSPACE(\log n).$$

Per NPSPACE esiste anche una alf. basata su Verificatore.

Ora mostriamo $NPSPACE = PSPACE$.

LEMMA $PATH \in NL$.

(Infatti vedremo che PATH è NL-completo)

DIM. L'idea è quella di memorizzare un cammino di lunghezza n e accettare se esiste.

- Su input G, s, t ; se $s = t$ accetta.
- Calcola $n := \#V$ deterministicamente.

- CURNODE = S. For $i = 1, \dots, M$:
  - Scelgo in modo non-det. un nodo $u \in V$.
  - If $(CURNODE, u) \in E$, allora CURNODE = u. Se CURNODE = t restituisco ACC
  - Else $(CURNODE, u) \notin E$, allora REJECT.

- REJECT.

Il numero di variabili è $O(1)$ su ogni

ramo di computazione e quindi $O(\log n)$ spazio.

Inoltre, è facile vedere che esiste $s\text{-}t$ sse esiste un ramo di computazione che accetta ∎