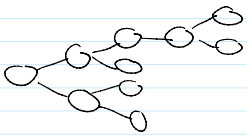
Complexité en espace et PSPACE

17 mars 2022 | 10:17

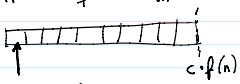
- Une MT dét. est en espace $O(f(n))$ si elle utilise tjrs au plus $c \cdot f(n)$ cellules de ruban distinctes. (c = constante)
- Une MT non-dét est en espace $O(f(n))$ si $\forall$ séq. de transitions possible, la MT accède à $\leq c \cdot f(n)$ cellules



- Espace $\leq$ Temps
- Temps $\leq 2^{\text{Espace}}$

Prop : si une MT M est en espace $O(f(n))$ et que M arrête sur toute entrée, alors M est en temps $O(2^{d \cdot f(n)})$, d = constante

- Si M n'a pas de boucle $\circlearrowleft$, alors M ne rencontre pas 2x la même config.
  Supposons que M utilise un espace $\leq c \cdot f(n)$



$(|\Sigma|+|Q|+1)$

Le # de cfg sur cet espace est $\leq (|\Sigma|+|Q|+1)^{c \cdot f(n)}$

Au pire, la MT visite chaque cfg une fois

$\Rightarrow$ le temps est $\leq (|\Sigma|+|Q|+1)^{c \cdot f(n)}$

$= \left(2^{\log(|\Sigma|+|Q|+1)}\right)^{c \cdot f(n)}$

$= (2^d)^{c \cdot f(n)} = 2^{d \cdot c \cdot f(n)}$

$d \cdot c$ = constante ∎

$\text{DSPACE}(f(n)) = \{L : \exists \text{ MT } M \text{ qui décide } L \text{ en espace } O(f(n))\}$

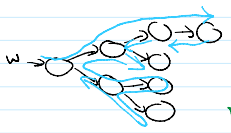
$\text{NSPACE}(f(n)) = \{L : \exists \text{ MT non-dét } M \text{ dont le langage est } L \text{ et qui prend un espace } O(f(n))\}$

$$\text{PSPACE} = \bigcup_{k=0}^{\infty} \text{DSPACE}(n^k)$$

$$\text{NPSPACE} = \bigcup_{k=0}^{\infty} \text{NSPACE}(n^k)$$

Prop : $P \subseteq \text{PSPACE}$

Prop : $NP \subseteq \text{PSPACE}$



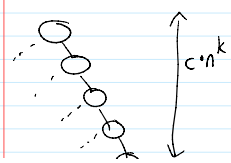
Soit $L \in NP$, $\exists$ MT M non-dét. avec langage L en temps $\leq c \cdot n^k$.

Pour décider L en espace poly:

```
ma_fct(w, cfg, nbsteps)
    si cfg est acceptante
        return true
    si nbsteps > c·n^k   (n=|w|)
        return false

    pour cfg2 pouvant suivre cfg dans M
        si ma_fct(w, cfg2, nbsteps+1) accepte
            return true
    return false
```

Cet algo explore un arbre de récursion de profondeur $\leq c \cdot n^k$



pour cfg2 pouvant suivre cfg dans M
  si ma_fct(w, cfg2, nbsteps+1) accepte
    return true
return fake

Appel init: ma_fct(w, cfg_init, 0)

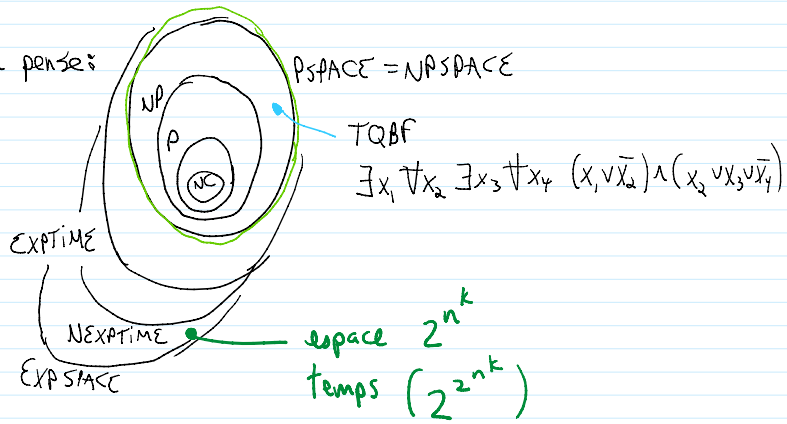
De plus, l'espace pour chaque appel est en espace $O(n^k)$ car on n'a qu'à stocker cfg et cfg2, et chaque cfg est $O(n^k)$ car M prend un temps $O(n^k)$.
Donc l'espace est $O($prof de l'arbre $\times$ espace par noeud$)$
$= O(n^k \cdot n^k) = O(n^{2k})$ ∎

Autre preuve :
On montre SAT $\in$ PSPACE.

sat(φ, vars $x_1, \dots, x_n$)
// espace $O(n)$ pour chaque assignation A des x :
// espace $O(|\varphi|)$ si A satisfait φ
    accepte
rejette

$\Rightarrow$ SAT $\in$ PSPACE $\rightarrow \forall L \in NP,\ L \leq_p$ SAT
$\Rightarrow \exists f$ en espace + temps poly tq.
$w \in L \Leftrightarrow f(w) \in$ SAT
$\Rightarrow$ Pour décider L, rouler sat($f(w)$) ∎

Ce qu'on pense :



PSPACE $\neq$ EXPSPACE

Autres classes d'espace

LOGSPACE = DSPACE($\log n$) = L

ex : expression bien parentésée : (()(()())) → +1 +1 -1 +1 +1 -1 ...

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

Q : $L \stackrel{?}{=} NL$

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

Q: $L \stackrel{?}{=} NL$

$PSPACE = NPSPACE$

Théorème de Savitch

- $NSPACE(f(n)) \subseteq DSPACE(f(n)^2)$

Ceci implique $PSPACE = NPSPACE$, car
- $PSPACE \subseteq NPSPACE$ (trivial)
- $NPSPACE \subseteq PSPACE$

  Soit $L \in NPSPACE$. Alors $\exists k$ tq $L \in NSPACE(n^k)$

  Par Savitch, $L \in DSPACE(n^{2k})$

  $\Rightarrow L \in PSPACE$

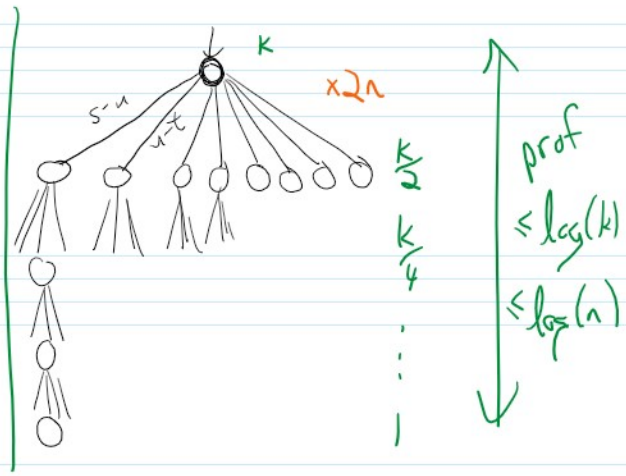
Pour prouver Savitch, on a besoin de:

Lemme: Soit $DPATH = \{\langle G, s, t, k\rangle$: G est un graphe orienté et $\exists$ un chemin de s à t avec $\leq k$ arêtes dans $G\}$

Alors, on peut décider DPATH en utilisant un espace $O((\log n)^2)$, où $n = |V(G)|$.

Preuve: voici un algo:

```
dpath(G, s, t, k)
    si k == 0, return (s == t)
    si k == 1, return ((s,t) ∈ E(G))
    // brancher sur le pt milieu u entre s et t
    // s --⌊k/2⌋--> u --⌈k/2⌉--> t
    pour u ∈ V(G)
        si dpath(G, s, u, ⌊k/2⌋) ∧ dpath(G, u, t, ⌈k/2⌉)
            return true
    return false
```



Cet algo explore un arbre de récursion de profondeur $O(\log n)$. Chaque appel prend un espace $O(\log n)$ pour stocker s, t, u, k si on suppose que les sommets de G sont représentés par $V(G) = \{1, 2, \ldots, n\}$.

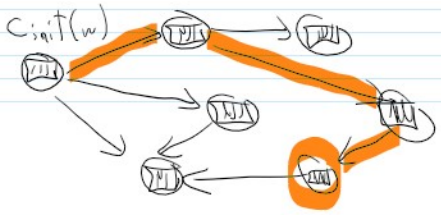
$\Rightarrow$ espace total $\in O(\log n \cdot \log n) = O((\log n)^2)$. ∎

sommets de G sont représentés par $\lceil \log c_1, c_2, \ldots, n \rceil$.

$\Rightarrow$ espace total $\in O(\log n \cdot \log n) = O((\log n)^2)$. ∎

Soit $L \in NSPACE$. $\exists M$ non-dét. avec langage L en espace $O(n^k)$.
Encore une fois, le # de configs possible est $O(2^{d \cdot n^k})$

Soit G_M le graphe où
- $V(G)$ = cfgs possibles de M
- $(c_1, c_2) \in E(G)$ si M permet d'aller de c_1 vers c_2.



On note que $w \in L$ ssi $\exists$ chemin de c_{init} vers un c_i acceptant.

Voici l'algo pour décider L en espace poly:

pour chaque c_i acceptante // $O(n^k)$
  si dpath$(G, c_{init}, c_i, 2^{dn^k})$
    accepter
rejeter

stocker l'entier 2^{dn^k}

L'espace requis est $O(n^k)$ pour c_i

$$O(\log(|V(G)|)^2 + \log(2^{dn^k}))$$
$$= O(\log(2^{dn^k})^2 + \log(2^{dn^k}))$$
$$= O(\log(2^{dn^k})^2)$$
$$= O((dn^k \cdot \log 2)^2)$$
$$= O(d^2 \cdot (n^k)^2)$$
$$= O((n^k)^2)$$ ∎