Space complexiteit

Theoretische Informatica II

1 Definitie van space complexiteit

Zie het boek, inleiding tot Hoofdstuk 8, alsook Paragraaf 8.4 voor Turing machines met read-only input tape.

2 Van NTIME naar SPACE

Moraal van volgende stelling is dat we space kunnen hergebruiken; met tijd gaat dat jammer genoeg niet!

Stelling. *Als* $f(n) \geqslant n$*, dan* $\text{NTIME}(f) \subseteq \text{SPACE}(O(f))$.

Bewijs: Zij A een probleem in $\text{NTIME}(f)$. Dus A wordt opgelost door een niet-deterministische TM M in tijd $f(n)$. Elke mogelijke run van M op een input w wordt volledig bepaald door de reeks keuzes $c_1, c_2, \ldots, c_{f(n)}$ die we maken voor de mogelijke transities in elke stap. Hier staat dus c_i voor het nummer van de transitie die we hebben gedaan in de i-de stap van de run. Er zijn maar een vast aantal transities in het programma van M (net zoals elk computerprogramma uit een vast aantal statements bestaat). Noemen we dit aantal k, dan is een keuzesequentie $c_1, c_2, \ldots, c_{f(n)}$ dus een string over het eindige alfabet $\{1, \ldots, k\}$.

We kunnen nu A alternatief oplossen d.m.v. volgend algoritme. Genereer alle mogelijke keuzesequenties één voor één, waarbij we de space waarin de sequenties geschreven staan telkens hergebruiken. Voor elke keuzesequentie doen we de overeenkomstige run van M, waarbij we opnieuw de space ingenomen door deze verschillende runs telkens hergebruiken. Als we een run vinden die aanvaardt, aanvaarden we ook. Als we er geen vinden die aanvaardt, rejecten we.

Het net beschreven algoritme kan geïmplementeerd worden in space $O(f(n))$: iedere keuzesequentie is maar $f(n)$ lang, en elke mogelijke run is ook maar $f(n)$ lang. *Vraag: waar gebruiken we nu het gegeven dat* $f(n) \geqslant n$*?*

3 De stelling van Savitch: van NSPACE naar SPACE!

Hier volgen we in ruime mate de presentatie in het boek. Om echter de complicatie die optreedt op het einde van het bewijs uit het boek te vermijden, herformuleren we de stelling als volgt:

Stelling. *Als* $f(n) \geqslant \log n$*, en als* $f(n)$ *berekenbaar is uit* 1^n *in space* $O(f(n))$*, dan*

$$\mathrm{NSPACE}(O(f)) \subseteq \mathrm{SPACE}(O(f^2)).$$

Het bewijs is gegeven in het boek Paragraaf 8.1, behalve dat de complicatie op het einde vermeden wordt, en vervolgt in Paragraaf 8.4.

4 Van NSPACE naar (exponentiëel meer) TIME

Lees om te beginnen Paragraaf 8.2. Lees dan verder in Paragraaf 8.4. Zonder ze expliciet te formuleren, wordt daar volgende stelling geargumenteerd:

Stelling. *Als* $f(n) \geqslant \log n$*, dan* $\mathrm{NSPACE}(f) \subseteq \mathrm{TIME}(2^{O(f)})$.

Een mooi gevolg van deze stelling is dat NL $\subseteq$ P.

5 De configuratiegraaf van een machine op een input

Zij M een TM, ze mag niet-deterministisch zijn, en w een input voor M. De configuratiegraaf van M op w heeft als knopen alle mogelijke configuraties waarin M zich kan bevinden als je M runt op input w. Er is een pijl van knoop K_1 naar knoop K_2 als M van configuratie K_1 kan overgaan in configuratie K_2 in 1 stap.

6 De stelling van Immerman–Szelepcsenyi

Lees eerst in het boek (bewijs van Theorem 8.22) het mooie bewijs dat $\overline{\mathrm{PATH}}$, het complement van PATH, in NL zit. Hieruit volgt meer dan enkel NL = coNL, namelijk:

Stelling. *Zij* f *zoals in de stelling van Savitch. Dan*

$$\mathrm{coNSPACE}(O(f)) = \mathrm{NSPACE}(O(f)).$$

Bewijs: Zij A in $\mathrm{coNSPACE}(O(f))$. Dit betekent dat het complement van A, noteer dit $\overline{A}$, in $\mathrm{NSPACE}(O(f))$ zit. Er is dus een niet-deterministische Turing machine (met read-only input string) M die $\overline{A}$ beslist in space

$O(f(n))$. Beschouw nu een willekeurige input w, en zij G de configuratiegraaf van M op w. We hebben nu het volgende:

$$\begin{aligned} w \in A &\Leftrightarrow w \notin \overline{A} \\ &\Leftrightarrow \text{er is } \textit{geen} \text{ pad in } G \text{ van de startconfiguratie} \\ &\quad\ \text{naar de accept-configuratie} \\ &\Leftrightarrow \langle G, \text{start}, \text{accept} \rangle \in \overline{\text{PATH}} \end{aligned}$$

Nu weten we dat $\overline{\text{PATH}}$ in NL (niet-deterministische logaritmische space) kan opgelost worden. De grootte van de input voor $\overline{\text{PATH}}$ in ons geval, namelijk G, is gelijk aan $2^{O(f(n))}$ (waarom?) De logaritme daarvan is $O(f(n))$. We concluderen dat we $w \in A$ kunnen testen in niet-deterministische space $O(f)$, zoals gewenst.

Er is wel één probleem met deze redenering: om het log-space algoritme voor $\overline{\text{PATH}}$ toe te passen op input G mogen we G niet expliciet construeren: dit zou $2^{O(f)}$ space vereisen, en we willen maar $O(f)$ space gebruiken. Het algoritme voor $\overline{\text{PATH}}$ (zie pagina 301 in boek) heeft haar inputgraaf echter maar op twee heel specifieke manieren nodig:

1. Door de knopen lopen. De knopen van G zijn de configuraties van M op w. We kunnen gemakkelijk al deze configuraties, de ene na de andere, genereren, telkens de space hergebruikend, in $O(f)$ space (inderdaad, elke configuratie is $O(f)$ groot).

2. Testen of er een pijl is van een knoop naar een andere. Dit betekent of we kunnen testen of, gegeven twee configuraties, M in 1 stap van de eerste naar de tweede kan gaan. Omdat we het programma van M kennen is dit een eenvoudige check.

We concluderen dat we de graaf G helemaal niet expliciet hoeven te construeren.

Waar hebben we de twee eisen op f gebruikt? Dat $f(n) \geqslant \log n$ is nodig opdat een configuratie $O(f)$ groot is (zie onderaan pagina 300 in het boek). Dat $f(n)$ berekenbaar is in space $O(f)$ is nodig bij het genereren van de configuraties van M op w.