

Bonyolultságelmélet

Thursday 1st December, 2016, 22:21

Tárbonylultság

A futásidő mellett a felhasznált tárterület a másik fontos erőforrás.

Ismét igaz, hogy egy RAM-program esetében ha csak a használt cellák **számát** vennénk alapul, irreálisan alacsony tárigény-fogalomhoz jutnánk.

Cellánként

Egy memóriacella igénye egy adott inputon: a benne a program futása során tárolt **legnagyobb** szám bináris alakjának a **hossza**. (Azaz $1 + \lfloor \log a \rfloor$, ha ez a maximális érték a , kivéve, ha $a == 0$, mely esetben 1).

PLUSZ a cella **címének** bináris alakjának a hossza.

Példa

Ha a $W[7]$ regiszterben a program futása során a legnagyobb érték **20**, akkor ennek a regiszternek a tárigénye 5 (a $20 = 10100_2$ string hossza) plusz 3 (a $7 = 111_2$ string hossza) = **8**.

Tárigény egy adott inputon

A teljes program tárigénye egy adott inputon: az összes, a futás során **használt** cella tárigényének összege.

A két memóriamodell

Amint egy cellát megcímez a program (direkt vagy indirekt címmel, írásra vagy olvasásra), használnak minősül.

Kivéve a következő esetet:

- ha a program az **input** regisztereket **csak olvassa**,
- az **output** regiszterekben pedig **csak ír**, ráadásul stream módban: először $O[1]$ kap értéket, majd $O[2]$, $O[3]$ stb.

akkor az input és output regiszterek tárigényét nem kell bevenni a tárigénybe. (Ez az ún. „**offline**” vagy „lyukszalagos” eset.)
(Ez megfelel annak a memóriamodellnek, mikor a hívó fél biztosítja az I/O területet: az inputot nem írhatjuk át, az outputot pedig egy output stream formájában kapjuk.)

Tárbonylultság

A program tárigénye

A program tárigénye az $f(n) : \mathbb{N} \rightarrow \mathbb{N}$ függvény, ha bármely, legfeljebb n méretű inputon legfeljebb $f(n)$ tárat használ.

A probléma tárigénye

Egy probléma **eldönthető $\mathcal{O}(f(n))$ tárban**, ha van őt eldöntő, $\mathcal{O}(f(n))$ tárigényű program. Ezen problémák osztályát $\text{SPACE}(f(n))$ jelöli.

Példa

ELÉRHETŐSÉG eldönthető **lineáris** tárban:

- a szélességi keresés algoritmus a egy N -csúcsú gráf esetén két, egyenként legfeljebb N méretű csúcshalmazt tárol (a már elért ill. elért és kifejtett csúcso két), ez pl. N hosszú bitvektor alkalmazásával egy-egy regiszterben $\mathcal{O}(N)$ tárban megoldható.

Alapvető különbség tár és idő közt

A tár újrafelhasználható!

Lineáris tárban...

- ... eldönthető a HAMILTON-ÚT probléma is: ugyanazt a területet újrahasználva generáljuk a csúcsok összes permutációját
- ... eldönthető a $TSP(E)$ probléma is, hasonlóképp
- ... eldönthető a 3-SZÍNEZÉS probléma is: ugyanazt a területet újrahasználva generáljuk a csúcsok összes 3-színezését
- ...

Megjegyzés

Nem ismert, hogy NP és $SPACE(n)$ közt mi az összefüggés.

Csak annyit tudunk biztosan, hogy $NP \neq SPACE(n)$.

(NP zárt a visszavezetésre, $SPACE(n)$ pedig – mint látni fogjuk – nem az.)

Nemdeterminisztikus tárnyolultság

Nemdeterminisztikus program **idő**nyolultságát a **leghosszabb** szál hosszaként definiáltuk.

Nemdeterminisztikus program **tár**nyolultságát egy adott inputon hasonló módon: a **legtöbb tárat használó szál** tárigényeként definiáljuk.

A nemdeterminisztikus program tárnyolultsága szintén akkor $f(n)$, ha tetszőleges, legfeljebb n méretű inputon legfeljebb $f(n)$ tárat használ.

A nemdeterminisztikus programmal $\mathcal{O}(f(n))$ tárban eldönthető **problémák** osztályát $\text{NSPACE}(f(n))$ jelöli.

Példa

```
function ND-ELER( $G[1 \dots N][1 \dots N], s, t$ )
```

```
   $u := s$ 
```

```
  while  $u \neq t$  do
```

```
     $v := \text{ND}(1 \dots N)$  //  $v$  értéke nemdet. 1 és  $N$  közti egész
```

```
    if  $G[u][v] == 0$  then return FALSE
```

```
     $u := v$ 
```

```
  return TRUE
```

Tárigény: nemdeterminisztikus **logaritmus**: $\mathcal{O}(\log n)$.

Eldönti az ELÉRHETŐSÉG problémát.

(Plusz egy, N -ig menő ciklusszámlálóval a végtelen ciklus elkerülhető, továbbra is logaritmus tárat használva.)

A nevesített bonyolultsági osztályok

- **R, RE** – eldönthető ill. felismerhető problémák
- $\text{TIME}(f(n))$, $\text{NTIME}(f(n))$ – det/nemdet. $\mathcal{O}(f(n))$ időben eldönthető problémák
- $\text{SPACE}(f(n))$, $\text{NSPACE}(f(n))$ – det/nemdet. $\mathcal{O}(f(n))$ tárban eldönthető problémák
- $\mathbf{P} = \text{TIME}(n^k) = \bigcup_{k \geq 0} \text{TIME}(n^k)$ – polinomidőben eldönthető problémák
- $\mathbf{NP} = \text{NTIME}(n^k)$ – nemdet. polinomidőben eldönthető problémák
- $\mathbf{L} = \text{SPACE}(\log n)$ – logaritmikus tárban eldönthető problémák
- $\mathbf{NL} = \text{NSPACE}(\log n)$ – nemdet. logtárban eldönthetőek
- $\mathbf{PSPACE} = \text{SPACE}(n^k)$ – polinom tárban eldönthetőek
- $\mathbf{NPSPACE} = \text{NSPACE}(n^k)$ – nemdet. polinom tárban...
- $\mathbf{EXP} = \text{Time}(2^{n^k}) \dots$

Alapvető összefüggések

Alapvető összefüggések bonyolultsági osztályok közt

- i) $\text{TIME}(f(n)) \subseteq \text{NTIME}(f(n))$,
 $\text{SPACE}(f(n)) \subseteq \text{NSPACE}(f(n))$
- ii) $\text{NTIME}(f(n)) \subseteq \text{SPACE}(f^2(n))$
- iii) $\text{NSPACE}(f(n)) \subseteq \text{TIME}(k^{f(n)})$

Ötletek

Az i) pont világos.

Alapvető összefüggések

$$\text{NTIME}(f(n)) \subseteq \text{SPACE}(f^2(n))$$

Tehát egy $f(n)$ **idő**korlátos N nemdeterminisztikus programot lehet szimulálni $f^2(n)$ **tár**korlátos determinisztikus programmal. Feltehetjük, hogy N -nek mindig **pontosan két** választása van megállásig, a 0. és az 1.

A determinisztikus algoritmus:

- Futtatunk egy k számlálót 1-től fölfelé (ennyi lépésig szimuláljuk a nemdeterminisztikus programot).
- k minden értékére leszimuláljuk az összes k hosszú választási sorozatot (ez 2^k darab):
 - ha van köztük elfogadó, elfogadjuk az inputot;
 - ha mind elutasító, elvetjük az inputot;
 - különben növeljük k -t és folytatjuk a ciklust.

(Kimaradt részlet: t utasítás végrehajtása alatt egy RAM program $\mathcal{O}(t^2)$ memóriát használ – ez pl. ismét annak a következménye, hogy a szorzás **nem** elemi művelet, az összeadás nem növelheti „túlzott ütemben” a változók értékét)

$\text{NTIME}(f(n)) \subseteq \text{SPACE}(f^2(n))$, tovább

A szimulálást a tárterület újrafelhasználásával tesszük (feltehetjük, hogy N offline, így az inputot a szimuláció nem rontja el).

A k . menetben egy k hosszú bitvektorban tartjuk nyilván az aktuális választási sorozatot, ezt növeljük mondjuk binárisan 0-ról indítva.

Mivel $f(n)$ lépésben N is csak $f^2(n)$ tárat használ, a szimulációhoz elég $f^2(n)$ tár; mivel $k \leq f(n)$, a bitvektornak is elég $f(n)$ tár.

$$\text{NSPACE}(f(n)) \subseteq \text{TIME}(k^{f(n)})$$

Tehát egy $f(n)$ tárkorlátos N nemdeterminisztikus programot akarunk determinisztikusan szimulálni.

Feltehetjük, hogy

- N offline;
- elfogadás előtt közvetlenül kinullázza munkaregisztereit.

Nevezzük **konfigurációnak** a RAM program teljes leírását a számítás egy pillanatában, miután az n hosszú (a_1, \dots, a_n) inputon elindítjuk:

- a programszámláló értéke: k_1 konstans sok lehetőség
- az összes nemnulla munkaregiszter aktuális tartalma $(i, W[i])$ párok listájának formájában: $\mathcal{O}(f(n))$ bit hosszú lista, $2^{\mathcal{O}(f(n))}$ -féle lehetőség

Összesen $k_1 \cdot 2^{\mathcal{O}(f(n))} = k^{f(n)}$ konfiguráció.

Alapvető összefüggések

$\text{NSPACE}(f(n)) \subseteq \text{TIME}(k^{f(n)})$, tovább

Az adott inputhoz tartozó **konfigurációs gráf**: a konfigurációk a csúcsok, C_1 -ből akkor vezet él C_2 -be, ha N átléphet C_1 -ből C_2 -be.

A **determinisztikus** algoritmus: elkészítjük az inputhoz tartozó konfigurációs gráfot és azon lineáris időben megoldunk egy ELÉRHETŐSÉG problémát a kezdőkonfigurációból a (kinullázás miatt egyértelmű) elfogadó konfigurációba.

(A gráfot nem kell előre elkészítenünk és letárolnunk, on-the-fly is számíthatjuk.)

Megjegyzés

Az eddigiekből $\text{NTIME}(f(n)) \subseteq \text{TIME}(k^{f^2(n)})$, de $\text{NTIME}(f(n)) \subseteq \text{TIME}(k^{f(n)})$ is igaz.

A „nevesített” osztályok sorrendje

Következmény

$$\mathbf{L} \subseteq \mathbf{NL} \subseteq \mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{PSPACE} \subseteq \mathbf{EXP} (\subsetneq \mathbf{R})$$

Bizonyítás

$$\mathbf{L} \subseteq \mathbf{NL} \quad (\text{i})$$

$$\mathbf{NL} = \mathbf{NSPACE}(\log n) \subseteq \mathbf{TIME}(k^{\log n}) \subseteq \mathbf{P} \quad (\text{iii})$$

$$\mathbf{P} \subseteq \mathbf{NP} \quad (\text{i})$$

$$\mathbf{NP} = \mathbf{NTIME}(n^k) \subseteq \mathbf{SPACE}(n^k) = \mathbf{PSPACE} \quad (\text{ii})$$

$$\mathbf{PSPACE} = \mathbf{SPACE}(n^k) \subseteq \mathbf{TIME}(2^{n^k}) = \mathbf{EXP} \quad (\text{iii})$$

A **központi** kérdés, hogy $\mathbf{P} = \mathbf{NP}$ vagy sem, de a fentiek közül **egyiket** se tudjuk, hogy egyenlőség-e!

A hierarchia tételek

Megengedett függvények

Az $f : \mathbb{N} \rightarrow \mathbb{N}$ monoton növekvő függvény **megengedett**, ha létezik olyan program, ami az $1^n \mapsto 1^{f(n)}$ függvényt számítja ki (n darab egyesből $f(n)$ darab egyest készít) $\mathcal{O}(f(n))$ tárban és $\mathcal{O}(n + f(n))$ időben.

Időhierarchia tétel

Ha $f(n) > n$ megengedett függvény, akkor

$$\text{TIME}\left(o\left(\frac{f(n)}{\log f(n)}\right)\right) \subsetneq \text{TIME}(f(n)).$$

Emiatt pl. $\mathbf{P} \neq \mathbf{EXP}$.

Tárhierarchia tétel

Ha $f(n) > \log n$ megengedett függvény, akkor

$$\text{SPACE}\left(o(f(n))\right) \subsetneq \text{SPACE}(f(n)).$$

Emiatt pl. $\mathbf{L} \neq \mathbf{PSPACE}$.

Összefoglalás

- Megismertük egy RAM program tárigény-fogalmát, az offline programot és a $\text{SPACE}(f(n))$, $\text{NSPACE}(f(n))$ osztályokat.
- Néztünk néhány $\text{SPACE}(n)$ -beli problémát.
- Megneveztük az **L**, **NL**, **PSPACE**, **NPSPACE** és **EXP** osztályokat.
- Láttuk, hogy $\text{ELÉRHETŐSÉG} \in \text{NL}$.
- Megmutattuk, hogy $\text{TIME}(f(n)) \subseteq \text{NTIME}(f(n))$.
- Megmutattuk, hogy $\text{SPACE}(f(n)) \subseteq \text{NSPACE}(f(n))$.
- Megmutattuk, hogy $\text{NTIME}(f(n)) \subseteq \text{SPACE}(f^2(n))$.
- Megmutattuk, hogy $\text{NSPACE}(f(n)) \subseteq \text{TIME}(k^{f(n)})$.
- Ezekhez megismertük egy program adott inputhoz definiált konfigurációs gráfját.
- Megmutattuk, hogy $\mathbf{L} \subseteq \mathbf{NL} \subseteq \mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{PSPACE} \subseteq \mathbf{EXP}$.
- Kimondtuk az idő- és a tárhierarchia tételt.