Bonyolultságelmélet

Thursday 1st December, 2016, 22:20

Tárbonyolultság

A futásidő mellett a felhasznált tárterület a másik fontos erőforrás.

Tárbonyolultság

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.

Tárbonyolultság

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.

Tárbonyolultság

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).

Tárbonyolultság

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

Tárbonyolultság

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.

Tárbonyolultság

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) $= 8$.

Tárbonyolultság

Tárigény egy adott inputon

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

Tárbonyolultság

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 címzéssel, írásra vagy olvasásra), használtnak minősül.

Tárbonyolultság

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 címzéssel, írásra vagy olvasásra), használtnak minősül.
Kivéve a következő esetet:

Tárbonyolultság

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 címzéssel, írásra vagy olvasásra), használtnak minősül.
Kivéve a következő esetet:

- ha a program az input regisztereket csak olvassa,

Tárbonyolultság

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 címzéssel, írásra vagy olvasásra), használtnak minősül.
Kivéve a következő esetet:

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

Tárbonyolultság

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 címzéssel, írásra vagy olvasásra), használtnak minősül.
Kivéve a következő esetet:

- ha a program az input regisztereket csak olvassa,
- az output regiszterekbe 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.)

Tárbonyolultság

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 címzéssel, írásra vagy olvasásra), használtnak minősül.
Kivéve a következő esetet:

- ha a program az input regisztereket csak olvassa,
- az output regiszterekbe 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.)

Tárbonyolultság

A program tárigénye

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

Tárbonyolultság

A program tárigénye

A program tárigénye az $f(n) : \mathbb{N} \to \mathbb{N}$ 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.

Tárbonyolultság

A program tárigénye

A program tárigénye az $f(n) : \mathbb{N} \to \mathbb{N}$ 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 $\textsc{Space}(f(n))$ jelöli.

Tárbonyolultság

A program tárigénye

A program tárigénye az $f(n) : \mathbb{N} \to \mathbb{N}$ 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 $\textsc{Space}(f(n))$ jelöli.

Példa

$\textsc{Elérhetőség}$ eldönthető lineáris tárban:

Tárbonyolultság

A program tárigénye

A program tárigénye az $f(n) : \mathbb{N} \to \mathbb{N}$ 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 $\textsc{Space}(f(n))$ jelöli.

Példa

$\textsc{Elérhetőség}$ eldönthető lineáris tárban:

- a szélességi keresés algoritmusa egy N-csúcsú gráf esetén két, egyenként legfeljebb N méretű csúcshalmazt tárol

Tárbonyolultság

A program tárigénye

A program tárigénye az $f(n) : \mathbb{N} \to \mathbb{N}$ 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 $\textsc{Space}(f(n))$ jelöli.

Példa

$\textsc{Elérhetőség}$ eldönthető lineáris tárban:

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

Alapvető különbség tár és idő közt

A tár újrafelhasználható!

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 $\text{TSP}(E)$ probléma is, hasonlóképp
- ...eldönthető a 3-SZÍNEZÉS probléma is: ugyanazt a területet újrahasználva generáljuk a csúcsok összes 3-színezését
- ...

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 $\text{TSP}(E)$ 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 $\text{SPACE}(n)$ közt mi az összefüggés.

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 $\text{TSP}(E)$ probléma is, hasonlóképp
- ...eldönthető a 3-SZÍNEZÉS probléma is: ugyanazt a területet újrahasználva generáljuk a csúcsok összes 3-színezését
- ...

Megjegyzés

Nem ismert, hogy $\mathbf{NP}$ és $\text{SPACE}(n)$ közt mi az összefüggés.
Csak annyit tudunk biztosan, hogy $\mathbf{NP} \neq \text{SPACE}(n)$.

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 $\text{TSP}(E)$ probléma is, hasonlóképp
- ...eldönthető a 3-SZÍNEZÉS probléma is: ugyanazt a területet újrahasználva generáljuk a csúcsok összes 3-színezését
- ...

Megjegyzés

Nem ismert, hogy $\mathbf{NP}$ és $\text{SPACE}(n)$ közt mi az összefüggés.
Csak annyit tudunk biztosan, hogy $\mathbf{NP} \neq \text{SPACE}(n)$.
($\mathbf{NP}$ zárt a visszavezetésre, $\text{SPACE}(n)$ pedig – mint látni fogjuk – nem az.)

Nemdeterminisztikus tárbonyolultság

Nemdeterminisztikus program időbonyolultságát a leghosszabb szál hosszaként definiáltuk.

Nemdeterminisztikus tárbonyolultság

Nemdeterminisztikus program időbonyolultságát a leghosszabb szál hosszaként definiáltuk.

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

Nemdeterminisztikus tárbonyolultság

Nemdeterminisztikus program időbonyolultságát a leghosszabb szál hosszaként definiáltuk.

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

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

Nemdeterminisztikus tárbonyolultság

Nemdeterminisztikus program időbonyolultságát a leghosszabb szál hosszaként definiáltuk.

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

A nemdeterminisztikus program tárbonyolultsága 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 $\mathrm{NSpace}(f(n))$ jelöli.

Példa

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

$u := s$

while $u \neq t$ **do**

$v :=$ ND$(1 \ldots N)$ $//v$ értéke nemdet. 1 és N közti egész

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

$u := v$

return TRUE

Példa

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

$u := s$

while $u \neq t$ **do**

$v :=$ ND$(1 \ldots N)$ $//v$ értéke nemdet. 1 és N közti egész

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

$u := v$

return TRUE

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

Nemdeterminisztikus tárbonyolultság

Példa

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

$u := s$

while $u \neq t$ **do**

$v :=$ ND$(1 \ldots N)$ $//v$ értéke nemdet. 1 és N közti egész

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

$u := v$

return TRUE

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

Eldönti az ELÉRHETŐSÉG problémát.

Példa

function ND-ELER($G[1 \ldots N][1 \ldots N]$,$s$,$t$)
  $u := s$
  while $u \neq t$ **do**
    $v :=$ ND$(1 \ldots N)$ $//v$ értéke nemdet. 1 és N közti egész
    if $G[u][v] == 0$ **then return** FALSE
    $u := v$
  return TRUE

Tárigény: nemdeterminisztikus logaritmikus: $\mathcal{O}(\log n)$.
Eldönti az ELÉRHETŐSÉG problémát.
(Plusz egy, N-ig menő ciklusszámlálóval a végtelen ciklus elkerülhető, továbbra is logaritmikus tárat használva.)

A nevesített bonyolultsági osztályok

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

Alapvető összefüggések

Alapvető összefüggések bonyolultsági osztályok közt

Alapvető összefüggések

Alapvető összefüggések bonyolultsági osztályok közt

i) $\mathrm{TIME}(f(n)) \subseteq \mathrm{NTIME}(f(n))$,
$\mathrm{SPACE}(f(n)) \subseteq \mathrm{NSPACE}(f(n))$

Alapvető összefüggések

Alapvető összefüggések bonyolultsági osztályok közt

i) $\mathrm{TIME}(f(n)) \subseteq \mathrm{NTIME}(f(n))$,
$\mathrm{SPACE}(f(n)) \subseteq \mathrm{NSPACE}(f(n))$

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

Alapvető összefüggések

Alapvető összefüggések bonyolultsági osztályok közt

i) $\mathrm{TIME}(f(n)) \subseteq \mathrm{NTIME}(f(n))$,
$\mathrm{SPACE}(f(n)) \subseteq \mathrm{NSPACE}(f(n))$

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

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

Alapvető összefüggések

Alapvető összefüggések bonyolultsági osztályok közt

i) $\mathrm{TIME}(f(n)) \subseteq \mathrm{NTIME}(f(n))$,
$\mathrm{SPACE}(f(n)) \subseteq \mathrm{NSPACE}(f(n))$

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

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

Ötletek

Az i) pont világos.

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

Tehát egy $f(n)$ időkorlátos N nemdeterminisztikus programot lehet szimulálni $f^2(n)$ tárkorlátos determinisztikus programmal.

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

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

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

Tehát egy $f(n)$ időkorlátos N nemdeterminisztikus programot lehet szimulálni $f^2(n)$ tárkorlátos determinisztikus programmal.
Feltehetjük, hogy N-nek mindig pontosan két választása van megállásig, a $0.$ és az $1.$
A determinisztikus algoritmus:

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

Tehát egy $f(n)$ időkorlátos N nemdeterminisztikus programot lehet szimulálni $f^2(n)$ tárkorlátos 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

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

Tehát egy $f(n)$ időkorlátos N nemdeterminisztikus programot lehet szimulálni $f^2(n)$ tárkorlátos 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).

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

Tehát egy $f(n)$ időkorlátos N nemdeterminisztikus programot lehet szimulálni $f^2(n)$ tárkorlátos 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):

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

Tehát egy $f(n)$ időkorlátos N nemdeterminisztikus programot lehet szimulálni $f^2(n)$ tárkorlátos 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;

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

Tehát egy $f(n)$ időkorlátos N nemdeterminisztikus programot lehet szimulálni $f^2(n)$ tárkorlátos 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;

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

Tehát egy $f(n)$ időkorlátos N nemdeterminisztikus programot lehet szimulálni $f^2(n)$ tárkorlátos 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.

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

Tehát egy $f(n)$ időkorlátos N nemdeterminisztikus programot lehet szimulálni $f^2(n)$ tárkorlátos 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.)

Alapvető összefüggések

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

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

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

$\mathrm{NTIME}(f(n)) \subseteq \mathrm{SPACE}(f^2(n))$, 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.

$\mathrm{NTIME}(f(n)) \subseteq \mathrm{SPACE}(f^2(n))$, 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.

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

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

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

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

- N offline;
- elfogadás előtt közvetlenül kinullázza munkaregisztereit.

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

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 $(a_1, \ldots, a_n)$ 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

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

$\mathrm{NSPACE}(f(n)) \subseteq \mathrm{TIME}(k^{f(n)})$, 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.

$\mathrm{NSPACE}(f(n)) \subseteq \mathrm{TIME}(k^{f(n)})$, 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.

$\mathrm{NSPACE}(f(n)) \subseteq \mathrm{TIME}(k^{f(n)})$, 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.)

$\mathrm{NSPACE}(f(n)) \subseteq \mathrm{TIME}(k^{f(n)})$, 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 $\mathrm{NTIME}(f(n)) \subseteq \mathrm{TIME}(k^{f^2(n)})$, de $\mathrm{NTIME}(f(n)) \subseteq \mathrm{TIME}(k^{f(n)})$ is igaz.

A „nevesített” osztályok sorrendje

Következmény

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

A központi kérdés, hogy $\mathbf{P} = \mathbf{NP}$ vagy sem,

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

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

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

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

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

$\mathbf{PSPACE} = \mathrm{SPACE}(n^k) \subseteq \mathrm{TIME}(2^{n^k}) = \mathbf{EXP}$ (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 $f : \mathbb{N} \to \mathbb{N}$ monoton növő 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.

A hierarchia tételek

Megengedett függvények

Az $f : \mathbb{N} \to \mathbb{N}$ monoton növő 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

$$\mathrm{TIME}\Big(o\big(\frac{f(n)}{\log f(n)}\big)\Big) \subsetneq \mathrm{TIME}(f(n)).$$

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

A hierarchia tételek

Megengedett függvények

Az $f : \mathbb{N} \to \mathbb{N}$ monoton növő 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

$$\mathrm{TIME}\Big(o\big(\frac{f(n)}{\log f(n)}\big)\Big) \subsetneq \mathrm{TIME}(f(n)).$$

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

Tárhierarchia tétel

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

$$\mathrm{SPACE}\Big(o(f(n))\Big) \subsetneq \mathrm{SPACE}(f(n)).$$

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

Összefoglalás

- Megismertük egy RAM program tárigény-fogalmát, az offline programot és a $\mathrm{SPACE}(f(n))$, $\mathrm{NSPACE}(f(n))$ osztályokat.
- Néztünk néhány $\mathrm{SPACE}(n)$-beli problémát.
- Megneveztük az **L**, **NL**, **PSPACE**, **NPSPACE** és **EXP** osztályokat.
- Láttuk, hogy Elérhetőség $\in$ **NL**.
- Megmutattuk, hogy $\mathrm{TIME}(f(n)) \subseteq \mathrm{NTIME}(f(n))$.
- Megmutattuk, hogy $\mathrm{SPACE}(f(n)) \subseteq \mathrm{NSPACE}(f(n))$.
- Megmutattuk, hogy $\mathrm{NTIME}(f(n)) \subseteq \mathrm{SPACE}(f^2(n))$.
- Megmutattuk, hogy $\mathrm{NSPACE}(f(n)) \subseteq \mathrm{TIME}(k^{f(n)})$.
- 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.