Calculabilitate & Complexități
Subiectul 4

Complexitate Timp

Ce tre să știi?

Nota 6:

- modelul de masina Turing pe care se face evaluarea masurii timp

- definitia masurii timp

- definirea claselor de complexitate timp

- comprimarea benzilor (enunturi)

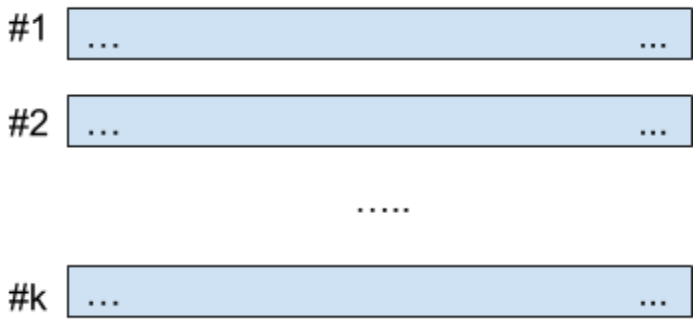
- eliminarea constantelor (enunturi)

- ierarhii de complexitate (enunturi)

Fiecare demonstratie, la alegere: 2p

Modelul de mașină Turing folosit

Vom folosi mașina Turing cu k benzi infinite la ambele capete. Poate să nu miște capul de citire-scriere la un moment dat. Mașinile considerate se opresc pe fiecare input.



Definiții

1. **$Time_M(n)$** = numărul maxim de pași pe care îi face mașina M pentru a decide o intrare de lungime n.
2. **$(D/N)TIME_k(f(n))$** = {L | există o mașină Turing M **deterministă/nedeterministă** cu **k** benzi astfel încât **L(M) = L** și există n_0 cu **$Time_M(n)$** <= f(n) pentru orice n >= n_0}
3. O funcție f(n) se numește **timp construibilă** dacă există o mașină Turing M și un n0 astfel încât **$Time_M(n) = f(n)$** pentru orice n > n0.
4. O funcție f(n) se numește **timp construibilă complet** dacă există o mașină Turing M astfel încât **$Time_M(n) = f(n)$** pentru orice n.

Teoreme

- **Comprimarea timpului de lucru cu un factor constant:**
    - $(D/N)TIME_k(f(n)) = (D/N)TIME_k(c\ f(n))$, unde c este o constanta pozitivă nenulă dacă
        1. k > 1
        2. f(n) / n tinde la infinit când n tinde la infinit
    - $(D/N)TIME_k(t\ n) = (D/N)TIME_k((1 + epsilon)\ n)$, pentru orice k > 1 și epsilon > 0.
- **Reducerea numarului de benzi:**
    - $((D/N)TIME_k(f(n)) \subseteq (D/N)TIME_1(f(n)^2)$, pentru orice k > 1 și orice f.
    - $((D/N)TIME_k(f(n)) \subseteq (D/N)TIME_2(f(n)\log(f(n)))$, pentru orice k > 1 și orice f.
- Oricare ar fi f(n) recursivă, există un limbaj recursiv L astfel încât $L \notin DTIME(f(n))$. (L nu aparține lui DTIME(f(n))).

Se aplică și pentru DSPACE, NTIME, NSPACE.

- Fie T1, T2 doua funcții, T2 timp construibila complet. Dacă limita când n tinde la infinit din T1(n)*log(T1(n)) / T2(n) tinde la 0, atunci există un L care aparține lui DTIME(T2(n)) și nu aparține lui DTIME(T1(n).

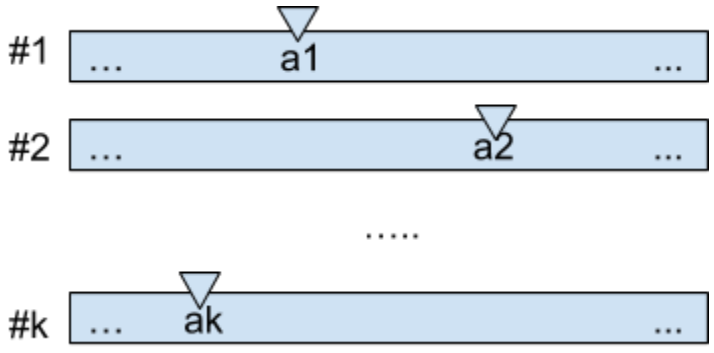
Ierarhie

- DTIME(f(n)) $\subseteq$ DSPACE(f(n))
- DSPACE(f(n)) $\subseteq$ DTIME($c^{f(n)}$), pentru f(n) $\geq$ log(n)
- NTIME(f(n)) $\subseteq$ NSPACE(f(n))
- NTIME(f(n)) $\subseteq$ DTIME($c^{f(n)}$)
- NSPACE(f(n)) $\subseteq$ DSPACE($f(n)^2$), pentru f(n) $\geq$ log(n) și f spațiu construibilă complet (Teorema lui Savitch)
- P = $\bigcup_k DTIME(n^k)$, NP = $\bigcup_k NTIME(n^k)$
    - DSPACE(log n) $\subseteq$ P $\subseteq$ NP $\subseteq$ NSPACE = PSPACE și DSPACE(log n) $\subset$ PSPACE

Demonstrații

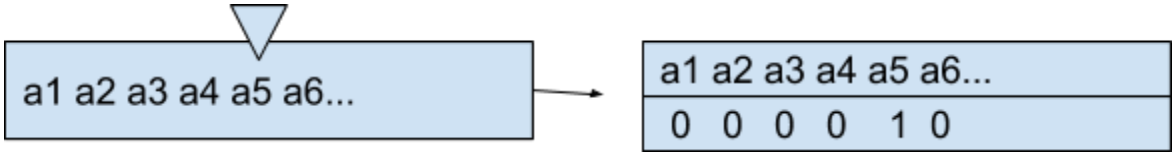
((D/N)TIME$_k$(f(n)) $\subseteq$ (D/N)TIME$_1$($f(n)^2$), pentru orice k > 1 și orice f.

Fie mașina Turing M cu TimeM(n) = f(n).



Și acum, construim mașina M' astfel:

- M’ are o singură bandă auxiliară, iar elementele ei vor fi **vectori cu 2k piste:**
    - Pe pista 2 * i - 1 se află conținutul benzii i a mașinii M
    - Pista 2 * i conține 0-uri mai puțin pe o poziție - are 1 unde se afla capul de citire-scriere al benzii i a mașinii M.



O pistă poate avea în cel mai rău caz f(n) celule ocupate (deoarece TimeM(n) = f(n), M nu are timp să ocupe mai mult de f(n) celule pe una dintre benzile ei).

Mașina M’:

- Citește conținutul benzii de la stânga la dreapta și memorează simbolurile de pe pistele 2 * i - 1 aflate imediat deasupra simbolurilor 1 de pe pistele 2 * i (maxim f(n) pași)
- Actualizează conținutul benzii de la dreapta la stânga:
    - Parcurgerea fiecărei celule: maxim f(n) pași:
        - Actualizarea simbolurilor de pe celula i: 1 pas
        - Dacă un cap de citire scriere aflat pe poziția i al mașinii M se mută la dreapta, trebuie actualizate pistele pare de pe celula din dreapta:
            - Un pas ca să ne mutăm la dreapta cu o poziție
            - Un pas ca să ne întoarcem
    - => 3f(n) pași

Deci, sunt f(n) + 3f(n) = 4f(n) pași care pot fi executați de maxim f(n) ori => 4f(n)) * f(n).

Nu știm dacă putem aplica teorema pentru eliminarea constantelor pentru că nu știm dacă funcția f este supraliniară. Dar, putem construi mașina M” cu L(M”) = L care face maxim f(n)/2 pași.

Atunci, mașina M’ poate simula în același mod mașina M” și va face $4\left(\frac{f(n)}{2}\right)^2 = f(n)^2$ pași.

Oricare ar fi f(n) recursivă, există un limbaj recursiv L astfel încât L $\notin$ DTIME(f(n)).

Fie L = { w $\in \{0, 1\}^*$ | w nu este acceptat de M în cel mult f(n) **pași**, unde $\widehat{M} = \widehat{w}$ }.

Notăm cu $\widehat{w}$ poziția lui w în mulțimea {0, 1, 00, 01, 10, 11, 000, ...}.

Codificărilor mașinilor Turing sunt peste alfabetul {0, 1, 2, (,), L, R}, iar $\widehat{M}$ reprezintă numărul de ordine al mașinii M în ordinea dată mai întâi de lungimea codificării și, în caz de egalitate, lexicografic.

Presupunem că există mașina M care acceptă L:

- M are ca input w
- Calculează lungimea lui |w| = n pe o bandă auxiliară
- Calculează f(n) pe aceeași bandă
- Găsește M' astfel încât $\widehat{w} = \widehat{M'}$
- Simulează mașina M' pe intrarea w, pentru maxim f(n) pași.
- Acceptă dacă M' se oprește și respinge sau dacă în cei f(n) pași pentru care a rulat simularea M' nu s-a oprit.

=> L = L(M). Mașina M e determinstă si se oprește pe fiecare intrare.

Am demonstrat că limbajul L e recursiv. Rămâne să demonstrăm că nu aparține lui DTIME(f(n)).

Alegem w cu $\widehat{M} = \widehat{w}$.

M acceptă în maxim f(n) pași => w nu e acceptat de M - contradicție.

M respinge în maxim f(n) pași => w ar trebui sa fie acceptat de M din definiția lui L - contradicție.

=> M trebuie sa accepte w în mai mult de f(n) pași.

=> TimeM(|w|) > f(|w|), deci L nu aparține lui DTIME(f(n)).