Subiectul 1

MAȘINI TURING

Definiții: 1) O <u>mașină Turing</u> se definește ca un 7-tuplu

$M = (Q, V, U, \delta, q_0, B, F)$, Q - mt. stărilor
V - alfabetul de input
U - alfabetul benzii, $V \subseteq U$
δ - funcția de tranziție parțial definită
q_0 - starea inițială
B - simbolul 'blank' (marchează sf. input)
F - mt. stărilor finale

- dacă mașina are <u>o bandă</u>:

$\delta : Q \times U \to (Q \times U \times \{L, R\})$

$\delta(q, a) = (s, b, X)$ are însemnătatea că, dacă mașina se află în starea q, trece în starea s, caracterul a este înlocuit cu b, iar capul de citire se deplasează în direcția indicată de X.

Mașina Turing s.n. <u>deterministă</u> dacă $|\delta(q, a)| \leq 1$, $\forall q \in Q$ și $a \in U$. Altfel, mașina Turing este <u>nedeterministă</u>.

- dacă mașina are <u>n-benzi</u>:

$\delta : Q \times U^n \to (Q \times U^n \times \{L, R\}^n)$

$\delta(q, a_1, \dots, a_m) = (s, b_1, \dots, b_m, X_1, \dots, X_n)$ are însemnătatea că, dacă mașina se află în starea q, trece în starea s, caracterul a_1 înlocuit cu b_1, a_2 cu b_2, etc., iar capul de citire al benzii i se deplasează în direcția indicată de X_i, $i = \overline{1,n}$.

Mașina Turing s.n. deterministă dacă $|\delta(q, a_1, \dots, a_m)| \leq 1$, $\forall q \in Q, (a_1, \dots, a_m) \in U^m$. Altfel, mașina Turing este nedeterministă.

2) Mașina Turing poate fi **DISPOZITIV DE ACCEPTARE** :

Input: $w \in U^*$

- mașina decide dacă $w \in L(M) = L$
- dacă M se oprește într-o stare finală, atunci w este acceptat ($w \in L(M)$). Altfel, w este respins ($w \notin L(M)$).

$\{ w \in V^* \mid w$ acceptat de $M \} = L(M) =$ limbajul acceptat de M

3) Mașina Turing poate fi **DISPOZITIV DE CALCUL** :

Avem funcția $f : \mathbb{N}^k \rightarrow \mathbb{N}$ - definită parțial sau total

- input: $(x_1, \dots, x_k) \in \mathbb{N}^k$, scrisă ca $1^{x_1+1} 0 1^{x_2+1} 0 \dots 0 1^{x_k+1}$
- dacă $f(x_1, \dots, x_k)$ este definit, mașina se oprește într-o stare finală și output: $1^{f(x_1, \dots, x_k)+1}$. Dacă nu e definit, mașina nu se oprește într-o stare finală
- dacă se poate găsi o astfel de mT, f se numește Turing calculabilă

4) RELAȚII ÎNTRE TIPURI DE MAȘINI TURING

- o mT cu n benzi este echivalentă cu o mT cu o singură bandă
- o mT nedeterministă cu o bandă este echivalentă cu o mT deterministă cu 3 benzi.
- o mT nedeterministă cu o bandă este echivalentă cu o mT deterministă cu o bandă

Subiectul 1

MAȘINI TURING

Demonstrații 1) Pt. M nedet. cu o bandă, există M' mT deterministă cu 3 benzi, echivalentă cu M.

Fie $M = (Q, V, U, \delta, q_0, B, F)$ - mT nedeterministă

Fie $E = \{(q, a, s, b, x)\}$: $q, s \in Q$ stări
$a, b \in U$, $b \neq B$
q nu e stare finală
$x \in L, R$

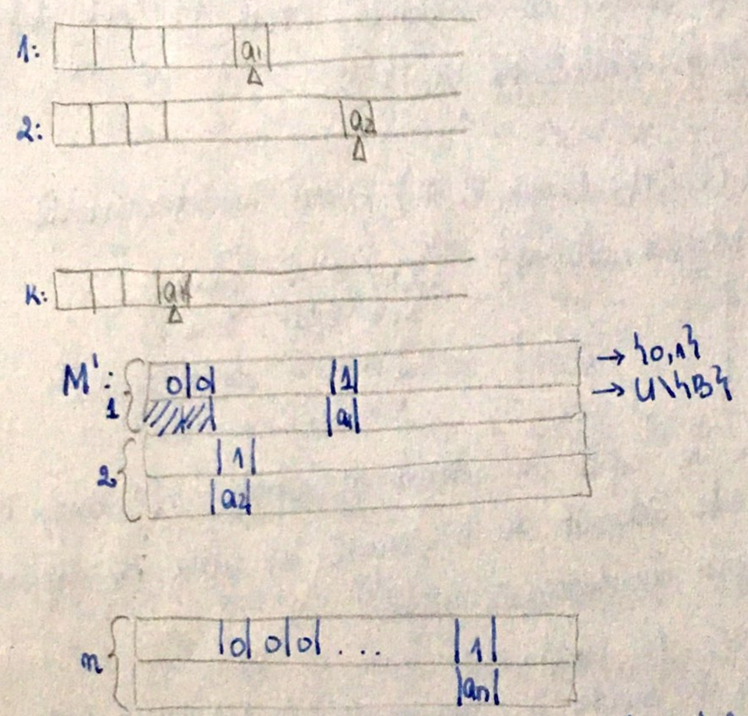
mașina se află în starea q, citește a, trece în starea s, a este înlocuit de b, capul de citire se deplasează după x.

M' are 3 benzi și lucrează astfel:

1. copiază w pe banda 3
2. generează pe banda 2 succesorul cuvântului din E^* înscris pe bandă
3. citește simbolul de pe banda 2: (q, a, s, b, x)
   a) verifică dacă mT se află în starea q. Dacă nu, GOTO 5
   b) verifică dacă simbolul citit de pe banda 1 este a. Dacă nu, GOTO 5
   c) scrie b peste a
   d) schimbă starea în s
   e) deplasează capul de citire de pe banda 1 în direcția x.
   f) deplasează capul de citire de pe banda 2 în dreapta
   g) dacă mai sunt simboluri pe bandă, GOTO 3. Altfel GOTO 4
4. Verificăm dacă starea obținută e finală. Dacă da, STOP, Altfel GOTO 5.
5. Copiază w de pe banda 3 pe 1. Merge în starea q_0 și GOTO 2

2) Pentru orice mTM cu k benzi, ∃ o mT M' cu o bandă, echivalentă cu M.
(în plus, dacă M deterministă ⇒ M' deterministă)

Avem M, o mașină cu k benzi:



M' are o singură bandă, iar elementele ei vor fi vectori de lungime k, având 2k piste.

- pe pista $2 \cdot i - 1$ conține 0-uri mai puțin pe o poziție unde are 1 (în locul în care se află capul de citire al benzii i a lui M)
- pista $2i$ conține banda i a mașinii M
- mașina M' citește conținutul benzii de la stânga la dreapta și memorează simbolurile de pe pistele pare, aflate dedesubtul simbolurilor de 1 de pe pistele impare.
- când ajunge la finalul benzii, simulează mișcările pe care le-ar fi făcut mașina M. Parcurge din nou banda, dar de la dreapta la stânga, și actualizează conținutul pistelor pare în conformitate cu simbolurile ce ar fi fost scrise de mașina M și conținutul pistelor impare în conformitate cu direcția în care s-ar fi deplasat fiecare cap de citire.

Subiectul 2

FUNCȚII RECURSIVE, CALCULABILE CU PROGRAME STANDARD TURING CALCULABILE

Definiții: 1) Funcții recursive - sunt de forma $f : \mathbb{N}^k \to \mathbb{N}$

- funcții elementare
  - funcții constante $C_p^{(x_1,\dots,x_k)} = p$
  - proiecții $\pi_r^{(x_1,\dots,x_k)} = x_r$
  - $succ(x) = x+1$
- operații

I) compunerea funcțională

$h : \mathbb{N}^m \to \mathbb{N}$, $g_i : \mathbb{N}^k \to \mathbb{N}$, $\forall\ 1 \le i \le m$

- formăm funcția f prin compunerea funcțională a lui h cu $g_1, \dots, g_m$

$$f(x_1,\dots,x_k) = h(g_1(x_1,\dots,x_k),\dots,g_m(x_1,\dots,x_k))$$

II) recurența primitivă

$f : \mathbb{N}^{k+1} \to \mathbb{N}$ este definită prin recurența primitivă de funcțiile $g : \mathbb{N}^k \to \mathbb{N}$, $h : \mathbb{N}^{k+2} \to \mathbb{N}$ dacă:

- $f(x_1,\dots,x_k,0) = g(x_1,\dots,x_k)$
- $f(x_1,\dots,x_k,t+1) = h(x_1,\dots,x_k,t,f(x_1,\dots,x_k,t))$

III) minimizarea mărginită

$f : \mathbb{N}^k \to \mathbb{N}$ se obține prin minimizarea mărginită de funcția $g : \mathbb{N}^{k+1} \to \mathbb{N}$ dacă $f(x_1,\dots,x_k) = \min\limits_y [g(x_1,x_2,\dots,x_k,y) = 0]$

2) Funcții Turing calculabile

O funcție $f : \mathbb{N}^k \to \mathbb{N}$ se numește Turing calculabilă dacă $\exists$ o mT ce are ca input $(x_1,\dots,x_k) \in \mathbb{N}^k$ și ca output $f(x_1,\dots,x_k)$

3) LIMBAJUL STANDARD

- programele standard calculează funcții de forma $f: N^K \to N$

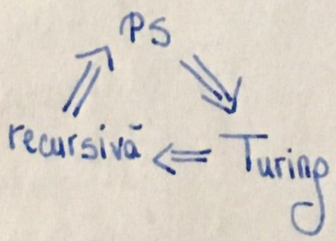
• VARIABILE
- de intrare: $X_1, X_2, X_3 \ldots$
- interne / de lucru / auxiliare: $Z_1, Z_2, Z_3, \ldots$
- de ieșire: Y

• INSTRUCȚIUNI
→ etichetate ex $A_1, A_2, \ldots B_1, \ldots$
→ neetichetate
- $V \leftarrow V$
- $V \leftarrow V+1$
- $V \leftarrow V \dot{-} 1$ $\begin{cases} V=0, \text{ nu face nimic} \\ V \leftarrow V-1, \text{ altfel} \end{cases}$
- IF $V \neq 0$ GOTO L, L etichetă

- un program standard se termină fie cu salt la eticheta E, fie prin salt la o etichetă care nu există, fie la finalul ultimei instrucțiuni
- funcția $f: N^K \to N$ pt. care $\exists$ un P.S. ce are ca input $x_1, \ldots, x_K$ și ca output $f(x_1, \ldots, x_K)$ se numește CALCULABILĂ cu PROGRAME STANDARD.

4) RELAȚII ÎNTRE ACESTE TIPURI DE FUNCȚII

- orice funcție calculabilă cu PS este Turing calculabilă.
- orice funcție recursivă este calculabilă cu PS.
- orice funcție Turing calculabilă este recursivă.



Subiectul 2

FUNCȚII RECURSIVE, CALCULABILE CU PROGRAME STANDARD, TURING CALCULABILE

Demonstrați: 1) Orice funcție calculabilă cu progr. standard este Turing calculabilă.

Fie $f: \mathbb{N}^k \to \mathbb{N}$ calculabilă cu PS.

Construim mașina M care are ca input $x_1, \ldots, x_k$.

$$\underbrace{1\,|\,1\,|\ldots|\,1}_{x_1+1}\,|\,0\,|\,\underbrace{1\,|\ldots|\,1}_{x_2+1}\,|\,0\,|\ldots|\,0\,|\,\underbrace{1\,|\ldots|\,1}_{x_n+1}\,|\,B\,B$$

Fixăm pe bandă pozițiile corespunzătoare variabilei Y și variabilelor $Z_1, \ldots, Z_m$. Deplasăm conținutul benzii la dreapta cu 2 poziții. Prima poziție va fi un θ, iar a doua un zero (separator). La dreapta inputului, fixăm câte un zero, urmat de θ pt. fiecare variabilă auxiliară.

$$\underbrace{\theta}_{Y+1}\,|\,0\,|\,\underbrace{1\,|\ldots|\,1}_{X_1+1}\,|\,0\,|\,\underbrace{1\,|\ldots|\,1}_{X_2+1}\,|\ldots|\,0\,|\,\underbrace{\bar{0}}_{Z_1+1}\,|\,0\,|\,\underbrace{\bar{0}}_{Z_2+1}\,|\ldots|\,B\,|\,B$$

$Y, Z_1, \ldots, Z_m$ inițial zero

O stare $\langle I_j \rangle$ a MT va fi o codificare a instrucțiunii I_j din PS care calculează f. CAZURI:

1) I_j: $V \leftarrow V$, trece în starea lui I_{j+1}, dacă $\exists$ altfel stare finală
2) I_j: $V \leftarrow V+1$, identifică poziția lui V pe bandă și crește cu o unitate nr. de 1 și trece în starea I_{j+1}
3) I_j: $V \leftarrow V \dot{-} 1$, găsește secvența de 1 asociată lui V; dacă secvența de 1 are lungime 1, trece în starea I_{j+1}, altfel se micșorează cu o unitate și trece în I_{j+1}

Scanned with CamScanner

I_j: IF $V \neq 0$ GOTO L, M identifică secvența de 1 asociată lui V

- dacă lungimea $= 1$ trece în stare I_{j+1}
- altfel, trece în eticheta I_{n+1}, dacă $L = E$ sau L nu există, sau în I_s, $s = \min\{t \mid I_t$ are eticheta $L\}$

În final, M se află în starea $\langle I_{n+1} \rangle$

2) Orice Turing calculabilă este recursivă.

~ preliminarii ~ toate funcțiile primitiv recursive ~

- funcția pereche $\langle , \rangle : \mathbb{N}^2 \to \mathbb{N}$, $\langle x, y \rangle = 2^x(2y+1) - 1$
- funcțiile $l, r : \mathbb{N} \to \mathbb{N}$

$$l(z) = x \text{ a.î. } \exists y \text{ cu } \langle x, y \rangle = z$$
$$r(z) = y \text{ a.î. } \exists x \text{ cu } \langle x, y \rangle = z$$

- al n-lea nr. prim $p_n : \mathbb{N} \to \mathbb{N}$
- Gödelizarea $z = [a_1, \dots, a_k] = p_1^{a_1} \cdot p_2^{a_2} \cdot \ldots \cdot p_k^{a_k}$

$$e_i(z) = a_i,\ lt(z) = k$$

Fie $f : \mathbb{N}^k \to \mathbb{N}$ Turing calculabilă $\Rightarrow \exists M = (Q, V, U, \delta, q_0, B, F)$ mT care calculează f

Renumerotăm stările $Q = \{q_0, \dots, q_r\} \to Q = \{0, \dots, r\}$

Renumerotăm alfabetul $U = \{0, 1, \dots, B, \dots\}$, $|U| = m \to U = \{0, 1, \dots, m\}$

Fie o configurație a mT formată din: starea q, poziția capului de citire și conținutul benzii - o secvență finită $s_1, \dots, s_m$,

O vom scrie drept $\langle \#(q), \langle p, [s_1, \dots, s_m] \rangle \rangle = z$

Fie $C_M(x, n)$, $x = (x_1, \dots, x_k)$, configurația lui M pe intrarea x, la pasul n, SCOPUL 1. Demonstrăm că C_M este recursivă.

$$C_M(x, 0) = \langle 0, \langle 0, [\underbrace{1, \dots, 1}_{x_1+1}, 0, \dots, \underbrace{1, 1, 1, \dots, 1}_{x_k+1}] \rangle \rangle$$

$$C_M(x, n+1) = \langle h_1(C_M(x, n)), \langle h_2(C_M(x, n)), h_3(C_M(x, n)) \rangle \rangle$$

$h_1(z)$ = numărul atașat stării în care trece mT din config. al cărei nr. este z, dacă z este asociat unei config. valide

$h_2(z)$ = poziția în care trece capul de citire scriere al lui M din config. al cărei nr. este z, dacă z este un nr. atașat unei config. valide

$h_3(z)$ = conținutul benzii în care ajunge mT din config. al cărei număr este z, dacă z este un nr. atașat unei config. valide

Dacă $C_M(x,n)$ este un nr. atașat unei config. finale:

$$C_M(x, n+1) = C_M(x,n)$$

Dacă h_1, h_2, h_3 recursive $\Longrightarrow$ C_M recursivă

Definim: $g_1, g_2, g_3 : \mathbb{N}^2 \to \mathbb{N}$

$g_1(a,b)$, unde a = nr. atașat unei stări
b = nr. atașat unui simbol

$g_1(a,b)$ = numărul stării în care trece mT, din starea a citind b

$g_2(a,b)$ = direcția în care se deplasează capul de citire, citind simbolul b, din starea a

$g_3(a,b)$ = numărul simbolului scris de mT fiind în starea a și citind simbolul b

$$h_1(z) = g_1\left(l(z), e_{l(r(z))}(r(r(z))\right)$$

$$h_2(z) = l(r(z)) + g_2\left(l(z), e_{l(r(z))}(r(r(z)))\right) - 1$$

$$h_3(z) = \frac{r(r(z))}{p_{l(r(z))}^{e_{l(r(z))}(r(r(z)))}} \cdot p_{l(r(z))}^{g_3\left(l(z),\, e_{l(r(z))}^{r(r(z))}\right)}$$

→ înmulțesc cu nr prim corespunzător literei care înlocuiește

↳ împart cu nr prim corespunzător literei înlocuite

h_1, h_2, h_3 recursive $\Longrightarrow$ $C_M(x,n)$ recursivă

Subiectul 3

MULȚIMI RECURSIVE, (REC)
RECURSIV ENUMERABILE, (RE)
NERECURSIV ENUMERABILE (NRE)

Definiții: 1) • Spunem că limbajul L este recursiv (REC) dacă una dintre următoarele condiții este îndeplinită:

fie $f(x) = (x \in L)? \begin{cases} 1, & x \in L \\ 0, & x \notin L \end{cases}$. Avem 3 variante:

I) f recursivă

II) f e calculabilă cu PS

III) $\exists$ o mT. M care se oprește pe fiecare intrare și acceptă L

• Spunem că limbajul L este recursiv enumerabil (RE) dacă $\exists$ o mT care acceptă L (dar care poate să nu se oprească pe fiecare intrare)

• Spunem că limbajul L este nerecursiv enumerabil (NRE) dacă $\nexists$ nicio mT care să accepte L.

$\begin{cases} REC = \{L \text{ limbaj recursiv}\} \\ RE = REC \cup \{L \text{ limbaj recursiv enumerabil}\} \\ NRE = RE \cup \{L \text{ limbaj nerecursiv enumerabil}\} \end{cases}$

Proprietăți și relații între ele:

- REC este închisă la reuniune, intersecție, complementară
- RE este închisă la $\cup, \cap$, nu e închisă la complementară
- $A \in REC \iff A$ și $C_A \in RE$
- $REC \subsetneq RE \subsetneq NRE$

Problema opririi: $HALT(x,y) \equiv$ programul cu nr. y se oprește pe x

(T) HALT nu este calculabilă cu PS.

Subiectul 3

MULȚIMI RECURSIVE (REC), RECURSIV ENUMERABILE (RE), NERECURSIV ENUMERABILE (NRE)

Demonstrații: 1) REC închisă la $\cap, \cup$, Complementară

Fie $L_1, L_2 \in REC$, $\exists$ mT M_1, M_2 cu $L(M_1) = L_1$ și $L(M_2) = L_2$ care se oprește pe fiecare input.

- Construim $M_\cap$ pentru $\cap$:
  Input: w
  Simulează pe M_1 intrarea w:
  - dacă M_1 respinge $\Rightarrow M_\cap$ respinge
  - dacă M_1 acceptă $\rightarrow$ simulează w pe M_2
    - M_2 respinge $\Rightarrow M_\cap$ respinge
    - M_2 acceptă $\Rightarrow M_\cap$ acceptă

Fie $w \in L(M_\cap) \Leftrightarrow w$ acceptat și de M_1 și de $M_2 \Leftrightarrow$
$\Leftrightarrow L \in L(M_1) \cap L(M_2) \Leftrightarrow w \in L_1 \cap L_2 \Rightarrow L_1 \cap L_2$ limbaj recursiv

- Construim $M_\cup$ pentru $\cup$:
  Input: w
  Simulează pe M_1 intrarea w:
  - dacă M_1 acceptă $\Rightarrow M_\cup$ acceptă
  - dacă M_1 respinge $\Rightarrow$ simulează w pe M_2
    - M_2 acceptă $\Rightarrow M_\cup$ acceptă
    - M_2 respinge $\Rightarrow M_\cup$ respinge

Fie $w \in L(M_\cup) \Leftrightarrow w \in L(M_1) \cup L(M_2) \Leftrightarrow w \in L_1 \cup L_2 \Rightarrow$ $L_1 \cup L_2$ limbaj recursiv

- Construim mT pentru C:
  Input: w
  Simulează pe M_1 intrarea w:
  - dacă M_1 acceptă $\Rightarrow M_C$ respinge
  - dacă M_1 respinge $\Rightarrow M_C$ acceptă

Fie $w \in L(M_C) \Leftrightarrow w$ nu e acceptat de $M_1 \Leftrightarrow w \notin L(M_1) \Rightarrow w \in C_{L_1}$
Deci $C(L_1)$ limbaj recursiv

2) RE este închisă la $\cap, \cup$.

Fie $L_1, L_2 \in RE$, atunci $\exists M_1, M_2$ cu $L(M_1) = L_1$ și $L(M_2) = L_2$,

2 mașini Turing care pot să nu se oprească pe inputurile respinse.

- Construim mașina Turing $M_\cap$:

  Input: w

  Simulează intrarea w pe mașina M_1

  M_1 se oprește și respinge $\Rightarrow M_\cap$ se oprește și respinge

  M_1 nu se oprește $\Rightarrow M_\cap$ nu se oprește

  M_1 acceptă $\Rightarrow$ simulează w pe M_2

  M_2 se oprește și respinge $\Rightarrow M_\cap$ respinge

  M_2 nu se oprește $\Rightarrow M_\cap$ nu se oprește

  M_2 se oprește și acceptă $\Rightarrow M_\cap$ acceptă

Fie $w \in L(M_\cap) \Leftrightarrow w \in L(M_1)$ și $w \in L(M_2) \Leftrightarrow L \in L(M_1) \cap L(M_2) \Leftrightarrow$

$\Leftrightarrow L \in L_1 \cap L_2 \Rightarrow L_1 \cap L_2$ este limbaj recursiv enumerabil

- Construim mașina Turing $M_\cup$:

  Input: w

  Simulează intrarea w alternativ pe M_1 și M_2:

  M_1 acceptă $\Rightarrow M_\cup$ acceptă

  M_2 acceptă $\Rightarrow M_\cup$ acceptă

  M_1 sau M_2 respinge $\Rightarrow$ așteaptă rezultatul celeilalte

  - cealaltă mașină acceptă $\Rightarrow M_\cup$ acceptă
  - cealaltă mașină respinge $\Rightarrow M_\cup$ respinge
  - cealalta mașina nu se oprește $\Rightarrow M_\cup$ nu se oprește

  Nici M_1 și nici M_2 nu se opresc $\Rightarrow M_\cup$ nu se oprește

Fie $w \in L(M_\cup) \Leftrightarrow w$ e acceptat de M_1 sau de $M_2 \Leftrightarrow$

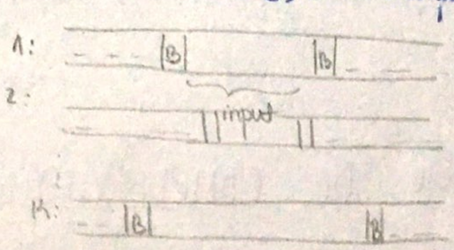
$w \in L(M_1) \cup L(M_2) \Leftrightarrow w \in L_1 \cup L_2$

$\Rightarrow L_1 \cup L_2$ limbaj recursiv enumerabil

Subiectul 4

CLASA DE COMPLEXITATE TIMP

Definiții: 1) Vom folosi mT cu k benzi infinite la ambele capete. Poate să nu miște capul de citire scriere, iar mașinile considerate se opresc la fiecare input.



- $Time_M(x)$ = nr. de pași până se oprește M pe intrarea x, dacă M deterministă
  = nr. minim — " — , dacă $x \in L(M)$, sau $x \notin L(M)$, când M nedeterministă
- $Time_M(n)$ = numărul maxim de pași pe care-l face M pt. a decide o intrare de lungime n.
- $(D)(N)\ Time_k(f(n)) = \{L \mid$ există o mT M deterministă/nedeterministă cu k benzi a.î. $L(M) = L$ și $\exists\ n_0$ cu $Time_M(n) \leq f(n)$ pt. orice $n \geq n_0\}$
- O funcție $f(n)$ se numește timp construibilă dacă $\exists$ o mT M și un n_0 a.î. $Time_M(n) = f(n)$, $\forall n \geq n_0$
- O funcție $f(n)$ se numește timp construibilă complet dacă $\exists$ o mT M și pt. $\forall n$, $Time_M(n) = f(n)$.

2) Teoreme:

- COMPRIMAREA TIMPULUI DE LUCRU CU UN FACTOR CT:
  - → $(D)(N) Time_k(f(n)) = (D)(N) Time_k(c \cdot f(n))$, $c \in \mathbb{N}^*$, $k > 1$ și $\lim_{n\to\infty} \frac{f(n)}{n} \to \infty$
  - → $(D)(N) Time_k(c \cdot n) = (D)(N)\ Time_k((1+\varepsilon)\, n)$, $\forall k > 1$ și $\varepsilon > 0$

- REDUCEREA NUMĂRULUI DE BENZI

→ $(D)(N)\,Time_k(f(n)) \subseteq (D)(N)\,TIME_1(f(n)^2)$, pentru orice $k \geq 1$ și $\forall f$

→ $(D)(N)\,Time_k(f(n)) \subseteq (D)(N)\,Time_2(f(n)\log(f(n)))$, $\forall k \geq 1$ și $\forall f$

- Oricare ar fi f recursivă, $\exists$ un limbaj recursiv L a.î. $L \notin DTIME(f(n))$.

// se aplică și pentru DSPACE, NSPACE, NTIME.

RELAȚII ÎNTRE CLASE DE COMPLEXITATE

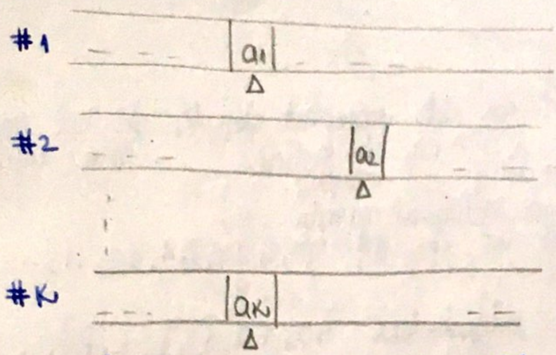
1) $(D)(N)\,Time(f(n)) \subseteq (D)(N)\,Space(f(n))$
2) $f(n) \geq \log(n)$, $\forall L \in DSPACE(f(n))$, $\exists c_L$ a.î. $L \in DTIME(c_L^{f(n)})$
3) $\forall f, \forall L \in NTIME(f(n))$, $\exists c_L$ a.î. $L \in DTIME(c_L^{f(n)})$
4) $\left.\begin{array}{l} f(n) \text{ spațiu construibilă complet} \\ f(n) \geq \log(n) \end{array}\right\} \Rightarrow NSPACE(f(n)) \subseteq DSPACE(f^2(n))$

Subiectul 4

CLASA DE COMPLEXITATE TIMP

Demonstrații: 1) $(D)(N)Time_k(f(n)) \subseteq (D)(N)Time_1(f(n)^2)$, $\forall\, k > 1$ și $\forall f$

Fie mT M cu $Time\, M(n) = f(n)$



Construim mașina M' astfel:

- M' are o singură bandă auxiliară (cu 2k piste), iar elementele ei vor fi vectori ~~cu 2k piste~~:
  - → pe pista $2i-1$ se află conținutul benzii i al mașinii M
  - → pe pista $2i$ se află 0-uri, mai puțin în locul în care se află capul de citire/scriere pe banda i, unde avem 1

a1	a2	a3	a4

→

..	a1	a2	a3	a4	a5	a6	...
	0	0	1	0	0	0	

O pistă poate avea în cel mai rău caz $f(n)$ celule ocupate ($Time_M(n) = f(n)$), astfel mașina M':

→ citește conținutul benzii de la stânga la dreapta și memorează simbolurile de pe pistele $2i-1$, aflate imediat deasupra simbolurilor de 1 de pe pistele $2\cdot i$ (maxim $f(n)$ pași) - ca să ajungă de

→ actualizează de la dreapta la stânga pt. fiecare celulă la 1 la caracter (parcurgerea fiecărei celule: $f(n)$)

1) actualizarea simbolurilor de pe celula i: 1 pas - $f(n)$

2) dacă un cap de citire/scriere aflat pe poziția i al mT M se mută la dreapta/stânga, trebuie actualizate celulele din dreapta: un pas ca să ne mutăm pe celula din dreapta, un pas ca să revenim $\Rightarrow 2\cdot f(n)$

$\Rightarrow 3 \cdot f(n)$ pași

În total avem $4 \cdot f(n)$ pași ce pot fi executați de maxim $f(n)$ ori $\Rightarrow$
$4 \cdot f(n)^2$ pași, astfel eliminând ct $\Rightarrow f(n)^2$

Deci $(D)(N)\mathrm{Time}_k(f(n)) \subseteq (D)(N)\mathrm{Time}_1(f(n)^2)$

2) Oricare ar fi $f(n)$ recursivă, $\exists$ un limbaj recursiv L a.î. $L \notin DTIME(f(n))$.

Fie $L = \{w \in \{0,1\}^* \mid w$ nu este acceptat de M în cel mult $f(n)$ pași, unde $\hat{M} = \hat{w}\}$ — nr. de ordine al cuv.; n e lungimea cuvântului

$\hat{w}$ { Notăm cu $\hat{w}$ poziția lui w în mț.: $\{0, 1, 00, 01, 10, 11, \dots\}$ — nr. ordine al mașinii

$\hat{M}$ { Codificăm mT M peste alfabetul $\{0,1,2,(,),L,R\}$, iar $\hat{M}$ reprezintă numărul de ordine al lui M, în ordinea dată mai întâi de lungimea codificării, iar la egalitate lexicografic.

Presupunem că $\exists$ M care acceptă L: M are ca input w, calculează lungimea lui $|w| = n$ pe o bandă, calculează $f(n)$ pe aceeași bandă.

M' { Găsește M' a.î. $\hat{w} = \hat{M'}$ și simulează w pe M' pentru maxim $f(n)$ pași. Acceptă dacă M' se oprește și respinge ~~sau~~ sau dacă M' ~~se oprește sau~~ dacă nu dă un răspuns în cei $f(n)$ pași.

$\Rightarrow L = L(M)$. M e deterministă și se oprește pe fiecare intrare

Deci L e recursiv și vrem să arătăm că $L \notin DTIME(f(n))$.

Fie w cu $\hat{M} = \hat{w}$

M acceptă în max $f(n)$ pași $\Rightarrow w$ nu $\in L$ (din ipoteză) contradicție

M respinge în max $f(n)$ pași $\Rightarrow w$ ar trebui să fie acceptat din definiția lui ~~L~~ – contradicție

$\Rightarrow M$ acceptă în mai mult de $f(n)$ pași

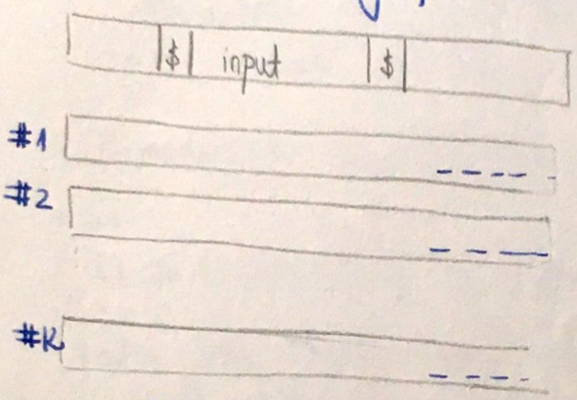
$\Rightarrow \mathrm{Time}_M(|w|) > f(|w|)$, deci $L \notin DTIME(f(n))$

Scanned with CamScanner

Subiectul 5

COMPLEXITATE SPAȚIU

Definiții: 1) Modelul de mT folosit: folosim mT offline, are o bandă pentru input read-only și K benzi auxiliare, infinite doar la un capăt.



- $Space_M(n)$ = numărul maxim de celule utilizate de către M pe una dintre auxiliare pentru a decide orice intrare de lungime n
- $(D)(N)Space_K(f(n)) = \{ L \mid$ există o mașină Turing M deterministă/nedeterministă cu K benzi a.î. $L(M) = L$ și $\exists$ n_0 cu $Space_M(n) <= f(n)$ pentru orice $n \geq n_0 \}$
- Funcția $f(n)$ se numește spațiu construibilă dacă $\exists$ o mT M și un n_0 astfel încât $Space\,M(n) = f(n)$, $\forall$ $n > n_0$
- Funcția $f(n)$ se numește spațiu construibilă complet dacă $\exists$ o mT M a.î. $Space_M(n) = f(n)$, $\forall$ n.

TEOREME:

// comprimarea timpului de lucru cu factor ct.

- $(D)(N)\,SPACE_K(f(n)) = (D)(N)\,Space_K(c \cdot f(n))$, c constantă pozitivă nenulă

// ~~tim~~ reducerea nr. de benzi

- $(D)(N)SPACE_K(f(n)) = (D)(N)Space_1(f(n))$, $\forall$ $K > 1$ și orice f

- Oricare ar fi $f(n)$ recursivă, există un limbaj recursiv L a.î. $L \notin DTIME(f(n))$.

// se aplică și pentru DSPACE, NTIME, DTIME.

- Fie S_1, S_2 două funcții $S_i(n) \geq \log(n)$; $i = \overline{1,2}$ și S_2 spațiu construibilă complet.

  Dacă $\lim_{n \to \infty} \frac{S_1(n)}{S_2(n)} = 0$, atunci $\exists$ $L \in DSPACE(S_2(n)) \setminus DSPACE(S_1(n))$

Subiectul 5

COMPLEXITATE SPAȚIU

Demonstrații: 1) $(D)(N)SPACE_k(f(n)) = (D)(N)SPACE_k(c \cdot f(n))$, unde c este o ct. > 0

Presupunem fără a restrânge generalitatea că $c > 1$. Dacă $c = 1$, evident, iar dacă $c < 1$, considerăm $1/c$.

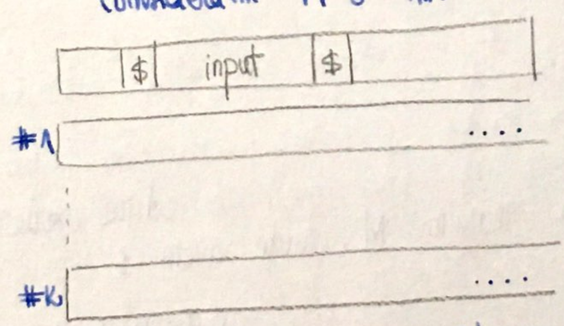
Demonstrăm prin dublă incluziune.

„$\subseteq$"

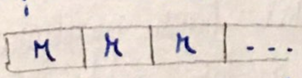
$(D)(N)SPACE_k(f(n)) \subseteq (D)(N)SPACE_k(cf(n))$, întrucât pentru orice funcție care ocupă $f(n)$, putem ocupa în plus $c \cdot f(n)$ celule pe o bandă.

„$\supseteq$"

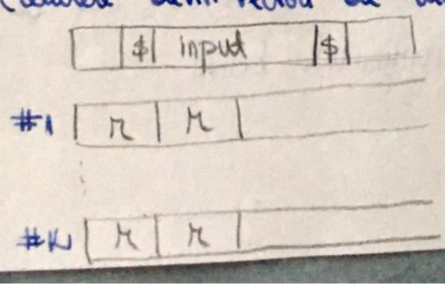
Considerăm M o mT:



Alegem un nr. $r \in \mathbb{N}$ cu prop că $r \geq 2c$. Mai întâi, împărțim benzile de la 1 la k în segmente de câte r celule.



Construim mașina M': fiecare celulă a benzilor de lucru corespunde câte unui grup de r simboluri de pe benzile de lucru ale lui M. (celulele devin vectori de lungime r)



Subiectul 6

Ierarhii de clase de complexitate

TIMP

- O funcție $f(n)$ se numește timp construibilă, dacă $\exists$ o mT și un n_0 a.î. $Time_M(n) = f(n), \forall n > n_0$.
- O funcție $f(n)$ se numește timp construibilă complet, dacă $\exists$ o mT a.î. $Time_M(n) = f(n), \forall n$.
- Fie T_1, T_2 două funcții, T_2 timp construibilă complet.
  Dacă $\lim_{n \to \infty} \frac{T_1(n) \log(T_1(n))}{T_2(n)} = 0 \Rightarrow \exists$ un $L \in DTIME(T_2(n))$ și $L \notin DTIME(T_1(n))$.

SPAȚIU

- O funcție $f(n)$ se numește spațiu construibilă, dacă $\exists$ o mT și un n_0 a.î. $Space_M(n) = f(n), \forall n > n_0$
- O funcție $f(n)$ se numește timp construibilă complet, dacă $\exists$ o mT a.î. $Space_M(n) = f(n), \forall n$
- Fie S_1, S_2 două funcții, $S_1, S_2(n) \geq \log(n)$ și S_2 spațiu construibilă complet.
  Dacă $\lim_{n \to \infty} \frac{S_1(n)}{S_2(n)} = 0$, atunci $\exists$ un $L \in DSPACE(S_2(n))$ și $L \notin DSPACE(S_1(n))$

- Oricare $f(n)$ recursivă, $\exists$ un limbaj recursiv L a.î. $L \notin DTIME(f(n))$ / DSPACE / NTIME / NSPACE.

RELAȚII CLASE:

1) $(D)(N)TIME(f(n)) \subseteq (D)(N)SPACE(f(n))$
2) Fie $f(n)$, cu $f(n) \geq \log(n)$, pt. $\forall L \in DSPACE(f(n))$, $\exists c_L$ a.î. $L \in DTIME(c_L^{f(n)})$
3) $\forall f, \forall L \in NTIME(f(n))$, $\exists c_L$ a.î. $L \in DTIME(c_L^{f(n)})$
4) $\left.\begin{array}{l} f(n) \text{ spațiu construibilă complet} \\ f(n) \geq \log(n) \end{array}\right\} \Rightarrow NSPACE(f(n)) \subseteq DSPACE(f(n)^2)$

Scanned with CamScanner