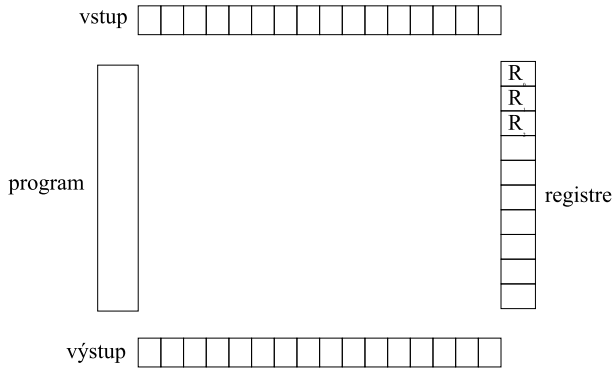


1 Paralelné RAM - PRAM

1.1 Základné definície

Označenie RAM je z "Random Access Machine". Vyzerá nasledovne:



Program je postupnosť inštrukcií, ktoré vyzerajú nasledovne:

COPY R_j $\langle R_j \rangle \rightarrow \langle R_0 \rangle$
 STORE R_j $\langle R_0 \rangle \rightarrow \langle R_j \rangle$
 IFZERO i if $\langle R_0 \rangle = 0$ then jump na i -ty riadok
 GOTO i jump na i -ty riadok
 ADD R_j $\langle R_j \rangle + \langle R_0 \rangle \rightarrow \langle R_0 \rangle$
 SUB R_j $\langle R_0 \rangle - \langle R_j \rangle \rightarrow \langle R_0 \rangle$
 CONST c $c \rightarrow \langle R_0 \rangle$
 HALT ACCEPT
 HALT REJECT
 READ
 WRITE
 MULT c $c \langle R_0 \rangle \rightarrow \langle R_0 \rangle$
 DIV c $\langle R_0 \rangle / c \rightarrow \langle R_0 \rangle$

Môžeme používať aj nepriamu adresáciu.

Definícia 1.1.1 Program pre PRAM je konečná postupnosť takýchto inštrukcií.

Poznámka 1.1.2 Vlastnosti:

1. V polynomiálnom čase nevyrobí čísla superpolynomiálnej dĺžky.

2. Miery zložitosti:

- (a) jednotková $\left\langle \begin{array}{l} TIME - \text{počet vykonávaných inštrukcií} \\ SPACE - \text{počet používaných registrov} \end{array} \right.$
- (b) logaritmická - zohľadňuje veľkosť čísla, s ktorým RAM pracuje

Veta 1.1.3 RAM a DTS sú v polynomiálnom vzťahu vzhľadom na zložitosti.

PRAM má neohraničený počet procesorov synchronne pracujúcich ozn. P_0, P_1, P_2, \dots . Každý z nich je RAM s vlastnými registrami $R_{i0}, R_{i1}, R_{i2}, \dots$. Ďalej má nekonečnú sadu globálnych registrov C_0, C_1, C_2, \dots . Každý procesor má tiež svoje identifikačné číslo id a má priamy prístup ku svojim aj globálnym registrom. Vstup a výstup pre PRAM predpokladáme v globálnych registroch.¹ Tiež má inštrukciu IDENT, ktorá vloží id číslo procesora do jeho $\langle R_0 \rangle$. Otázkou je, že ktoré procesory sú aktívne. Z P_0 urobíme generála. P_0 má špeciálny register C_{-1} , ktorý je dostupný aj ostatným a do ktorého na základe vstupu vloží index (index aktívneho procesora). Aktívne sú všetky procesory s $id \leq \langle C_{-1} \rangle$. Tiež môžeme vyvolať inštrukciu FORK.

Súčasný prístup ku globálnym registrom:

1. **EREW-PRAM** (Exclusive Read Exclusive Write) - k registru môže pristupovať najviac jeden proces
2. **CREW-PRAM** (Concurrent Read Exclusive Write) - viacerí môžu naraz čítať, ale iba jeden môže zapisovať.
3. **CRCW-PRAM** (Concurrent Read Concurrent Write)
 - (a) *Priority* - stanovíme prioritu, pri zapisovaní uspeje procesor s najmenším id .
 - (b) *Common* - všetky zapisujúce procesory zapisujú rovankú hodnotu.
 - (c) *Arbitrary* - viacero procesorov sa snaží zapísať, zapíše sa len jedno z nich. Zvolenie toho, ktorý zapíše trvá $\log P(n)$, kde $P(n)$ je počet aktívnych procesorov na vstupe dĺžky n .

1.2 Miery zložitosti

TIME $T(N)$ čas výpočtu P_0 na vstupe dĺžky n
 PROCESSORS $P(n)$ počet aktívnych procesorov

Pre $T(n) \geq \log n$ žiadne číslo nemôže mať viac ako $O(T(n))$ bitov. Pre $P(n)$ procesorov je to $P(n)T(n)T(N)$ bitov, teda priestor je ohraničený a netreba ho uvažovať.

1.3 Príklady výpočtov PRAM

Príklad 1.3.1 Výpočet MAX na CRCW-PRAM. Vstup je v globálnej pamäti uložený nasledovne: v $C_0 = n$ je počet hodnôt a v C_1, \dots, C_n sú jednotlivé hodnoty. Výstup bude v C_0 , čo bude $\max\{C_1, \dots, C_n\}$. PRAM používa n^2 procesorov a procesor P_0 .

¹Napríklad dĺžka vstupu je v C_0 a jednotlivé bity v $C_1, C_2, C_3, \dots, C_n$

Algoritmus:

1. *Inicializácia:* C_{n+1}, \dots, C_{n+n} na 0. Napríklad P_{1j} inicializuje C_{n+j} .
2. *Procesor* P_{ij} zapíše $C_{n+i} = 1$, ak $C_i < C_j$
3. *Procesor* P_{i1} zapíše do C_0 hodnotu C_i , ak $C_{n+i} = 0$

Poznámka 1.3.2 Po kroku 2 je $C_{n+i} = 0 \iff x_i$ je maximálne. Všetky procesory zapisujú rovnakú hodnotu a používame konštantný čas. Algoritmus by fungoval aj na modeloch Priority a Arbitrary.

Príklad 1.3.3 Výpočet MAX na EREW-PRAM. Výška rozhodovacieho stromu bude $\log n$.

Príklad 1.3.4 Sčítanie n čísel. Analogicky $T(n) = \log n$.

Veta 1.3.5 Nech $\{C_n\}_{n=1}^\infty$ je UBC uniformná trieda boolovských obvodov počítajúca funkciu $f_n : \{0, 1\}^n \rightarrow \{0, 1\}^{m(n)}$. $DEPTH(C(n)) = (\log n)^{O(1)}$ a $SIZE(C_n) = n^{O(n)}$. Potom existuje CREW-PRAM taký, ktorý počíta f v čase $(\log n)^{O(1)}$ na $n^{O(1)}$ procesoroch.

Veta 1.3.6 Nech M je CREW-PRAM, ktorý počíta v polylog čase na $P(n) = n^{O(1)}$. Potom existuje konštanta k a BC uniformná postupnosť obvodov $\{C_i\}_{i=1}^\infty$ taká, že C_n počíta na vstupe x_1, \dots, x_n výstupy $y_{11}, y_{21}, \dots, y_{ij}$, kde y_{ij} je hodnota j -teho bitu registra C_i v čase $T(n)$ pre $1 \leq i \leq T(n)T(n)$ a je $1 \leq j \leq kT(n)$. Navyše $DEPTH(C_n) = (\log n)^{O(1)}$ a $SIZE(C_n) = n^{O(1)}$

Záver 1.3.7 PRAM-y sú v druhej počítačovej triede.

Záver 1.3.8 NC je PRAM TIMEPROC($(\log n)^{O(1)} n^{O(1)}$).

Príklad 1.3.9 Triedenie je v NC. Vieme, že $NP \subset P$.

1.4 Redukovateľnosť

Definícia 1.4.1 Many-One redukovateľnosť. $L_1 \leq_m L_2$ ak \exists funkcia, taká, že $x \in L_1 \iff f(x) \in L_2$. Pre použiteľnosť v otázkach rozhodnuteľnosti $f \in REC$ [TODO]

Definícia 1.4.2 L_1 a L_2 sú ekvivalentné, ak $L_1 \leq L_2$ a $L_2 \leq L_1$.

Lema 1.4.3 $L' \in P$ a $L \leq_m^P L'$, potom $L \in P$. $L' \in NC$ a $L \leq_m^{NC} L'$, potom $L \in NC$.

Poznámka 1.4.4 $\leq_{m1}, \leq_m^P, \leq_m^{NC}, \leq_m^{NC^*}$ sú tranzitívne.

Definícia 1.4.5 Turingova redukovateľnosť. $L_1 \leq_R L_2$, ak existuje DTS s orákulom L_2 akceptujúci L_1 .

2 Úplnosť

Definícia 2.0.6 Jazyk L je P -ťažký (P -hard), pri turingovej NC^2 redukovanosti, ak pre každý $L' \in P$ platí, že $L' \leq_T^{NC} L$.

Definícia 2.0.7 Jazyk L je P -úplný (P -complete), ak pri turingovej NC redukovanosti, ak L je P -ťažký a $L \in P$.

Poznámka 2.0.8 Otvorený problém je $NC \stackrel{?}{=} P$. Vieme však, že $NC \subseteq P$.

Lema 2.0.9 Ak nejaký P -úplný (pri NC redukcii) problém (jazyk) patrí do NC , potom $NC = P$.

Poznámka 2.0.10 Všeobecne panuje domnienka, že $NC \neq P$.

2.1 Základné problémy

Problém 2.1.1 Generický problém simulácie TS.

Vstup: Slovo w , kód DTS $\langle M \rangle$, číslo t (unárne kódované). (Máme teda reťazec $w\# \langle M \rangle \#t$).

Otázka: Akceptuje M slovo w v priebehu t krokov ?

Veta 2.1.2 Generický problém simulácie TS je P -úplný pri $\leq_m^{NC^1}$ redukcii.

Dôkaz:

1. $\in P$. Štandardná simulácia z konštrukcie univerzálneho TS prebieha v polynomiálnom čase vzhľadom na maximálnu dĺžku pásky $\leq D(t)$
2. je potrebné ukázať, že ľubovoľný $L' \in P$ vieme redukovať na generický problém simulácie. Nech M je DTS pracujúci v čase $p(n)$ pre polynóm n . Keď máme pre vstup w dané M , treba vyrobiť vstup pre generický problém simulácie. Vstup vyzerá $w\# \langle M \rangle \#^{p(|w|)}$. Stačí však $w\# \langle M \rangle \#^{f(|w|)}$, pre ľahko počítateľnú funkciu f , $p(n) \leq f(n)$. Napríklad $f(n) = 2^{k \log(|w|+2)}$. Transformácia sa dá urobiť z w na $w\# \langle M \rangle \#^{f(|w|)}$. Vyrobit ju takto: najprv w skopíruje bez zmeny, potom zapíše $\langle M \rangle$ čo je konštanta natvrdo, a potom vyrobí $f(|w|)$. *VyhodnotreieBoo*

□

Problém 2.1.3 Vyhodnotenie Booleovského obvodu (Circuit Value Problem - CVP)

Vstup: kód obvodu C_n , vstupy x_1, \dots, x_n a vyznačený výstupný vrchol y .

Otázka: Je výstup y pri vstupoch x_1, \dots, x_n TRUE ?

Veta 2.1.4 CVP je P -úplný pri $\leq_m^{NC^1}$ redukcii

²vždy treba spomenúť o ktorej redukovanosti sa bavíme

Dôkaz:

1. $CVP \in P$. viď simulácia bool. obvodu na DTS.
2. CVP je P-ťažký. Stačí ukázať, že obvod pre simuláciu DTS vieme zostrojiť v NC^1

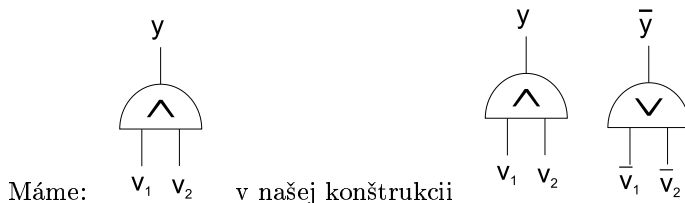
□

Veta 2.1.5 *Problém vyhodnotenia monotónnych boolovských obvodov³ je P-úplný.*

Dôkaz:

1. $\in P$ - zřejmé
2. CVP sa dá redukovť na monotónne CVP takto
 - (a) Ak sa dohodneme, že uvažujeme boolovské obvody s negáciou len na vstupe (tj. vstup $x_1, \bar{x}_1, \dots, x_n, \bar{x}_n$), tak je to zřejmé. Pre daný obvod a vstup x_1, \dots, x_n vyrobíme monotónny obvod C'_{2n} tak, že, zdvojíme vstupy.

vstup	1	0	1	1	0	1
monotónny vstup	10	01	10	10	01	10
 - (b) Ak trváme na všeobecnom použití negácie, tak vyrobíme pre každé hradlo tieňové hradlo počítajúce negáciu (opäť na zvojenom vstupe).



□

Problém 2.1.6 *Maximálna nezávislá množina (maximal independent set).*

Vstup: konečný graf

Výstup: množina vrcholov B taká, že žiadne dva nie sú spojené hranou a každý vrchol z $V - B$ je spojený hranou s niektorým z vrcholov v B.

Poznámka 2.1.7 *Maximum independent set je NP-úplný.*

Problém 2.1.8 *LFMIS (Lexicographically first maximal independent set)*

Vstup: Graf G a vrchol v

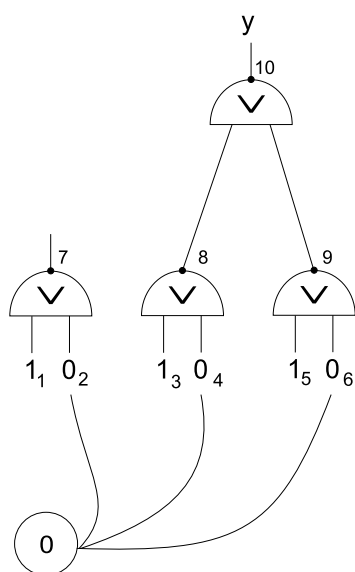
Otázka: Je v v LFMIS ?

³obvody len s \vee, \wedge a neobsahujúce negáciu

Veta 2.1.9 LFMIS je NP-úplný pri NC^1 redukcii.

Dôkaz:

1. Problém je v P. Funguje totiž greedy algoritmus.
2. Je P-ťažký. Ukážeme tak, že naň budeme redukovat CVP. Budeme uvažovať NOR obvody. Konštrukcia bude vyzerat tak, že očísľujeme hradlá od 1 topologicky, od vstupov začínajúc. Pridáme vrchol 0 a spojíame ho so vstupnými vrcholmi so vstupnou hodnotou 0. Za vrchol v LFMIS zvolíme výstupný vrchol obvodu. Tvrdíme, že okrem 0 bude v LFMIS vrchol i práve vtedy, keď jeho hodnota je 1.



□

Problém 2.1.10 Problém prázdnoti pre CFG (bezkontextové gramatiky)

Vstup: CFG gramatika G

Otázka: Je $L(G) = \emptyset$?

Veta 2.1.11 Problém prázdnoti pre CFG je P-úplný.

Dôkaz:

1. $\in P$ vid' FOJA 2. ročník.
2. je P-ťažký. Dokazujeme to redukciovou CVP na problém prázdnoti pre CFG. Pre daný obvod zostrojíme CFG takto:
 $N = \{i \mid i \text{ je hradlo v booleovskom obvode}\}$

$$T = \{a\}$$

$$P = \left\{ \begin{array}{ll} i \rightarrow jk & \text{ak } i \text{ je } \wedge \text{ hradlo so vstupmi } j, k \\ i \rightarrow j|k & \text{ak } i \text{ je } \vee \text{ hradlo so vstupmi } j, k \\ i \rightarrow a & \text{ak } i \text{ je vstup s hodnotou 1} \\ \sigma \rightarrow j & j \text{ je výstupný vrchol} \end{array} \right\}$$

□

3 Dodatok

3.1 Faq

Otázka 3.1.1 *Prečo asi platí $P \neq NC$?*

Odpoveď 3.1.2 *Jeden dôvod je, že všeobecné simulácie sú pomalé. Najlepšie známe simulácie sekvenčných modelov redukujú sekvenčný čas $T(n)$ na paralelný $T(n)/\log n$. Niekedy vieme urobiť $\sqrt{T(n)}$ v závislosti od modelu. Pritom sa používa $2^{T(n)^{O(1)}}$ procesorov. Rýchle simulácie sú známe len pre slabé modely⁴. Pre DTS platí*

1. Ak čas $T(n)$ je $n^{O(1)}$ a $SPACE(n) \leq \log n$ potom $\subseteq NC$.
2. Otvorený problém že čo sa nachádza tu.
3. Ak $T(n) = n^{O(1)}$ a $SPACE(n) = n^{O(1)} \supseteq P$

Pre PRAM platí:

1. Ak $n^{O(1)}$ procesorov a $\log^{O(1)} n$ čas, tak $\subseteq NC$.
2. Ak $n^{O(1)}$ procesorov a $n^{O(1)}$ čas, tak $\supseteq NC$

Otázka 3.1.3 *Kedy je paralelný algoritmus optimálny ?*

Odpoveď 3.1.4 *Optimálny je vtedy, ak platí $\underbrace{\text{paralelný čas} * \text{počet procesorov}}_{\text{práca, ktorú vynaloží paral. algoritmus}} \geq$ sekvenčný čas. Pre systém s P procesormi je najlepší možný čas rovný $\frac{\text{Sekvenčný čas}}{P}$, čo je to najlepšie, čo sa podarí. Keď je to $\frac{1}{2}$, tak je to dobré. Problém býva udržať všetky procesory v BUSY stave.*

Otázka 3.1.5 *Je NC dobre zvolená?*

Odpoveď 3.1.6 *Čo ak vezmeme polynomiálne zrýchlenie na polynomiálne veľa procesoroch...*

Otázka 3.1.7 *Čo tak pripustiť malú neefektívnosť, teda dovoliť aby práca bola rovná sekvenčný čas * $\log n$.*

⁴napríklad konečné automaty

3.2 Niektoré ďalšie modely

- PRAM-y ktorých komunikačný čas nie je zadaný
- Iteratívne a systolické polia. Je to mriežka a stav každého nódu v každom takte závisí od susedov.
 - von Neumann 4 susedia (hore, dole, vľavo, vpravo)
 - Moore 8 susedov (aj tie na diagonále)