Ďalej pre jednoduchosť budeme uvádzať všetky algoritmy len pre prípad, že vstupné čísla sú nezáporné celé čísla. Úprava týchto algoritmov tak, aby mohli pracovať aj so zápornými číslami nie je ťažká, ale pracná a zbytočne zastiera ideu algoritmu.

Vo všetkých uvedených algoritmoch sa objavujú dve metódy (princípy) návrhu paralelných algoritmov.

Prvou metódou je **práca v tímoch**, keď každej variante riešenia úlohy je priradený celý tím procesorov, ktoré overia, či táto varianta je riešením zodpovedajúcim vstupu. Práve jediný tím zistí, že jeho riešenie zodpovedá vstupu a svoje riešenie potom zapíše na výstup (do dohodnutých buniek spoločnej pamäte).

Druhou metódou je metóda **rozdeľ a panuj** známa zo sekvenčných algoritmov, ktorá sa na paralelnom modele aplikuje tak, že sa všetky podúlohy riešia súčasne na mnohých procesoroch.

Počítanie funkcií $\lfloor \log n \rfloor$ *a* $\lceil \log n \rceil$

Jednoduchým príkladom algoritmu pracujúceho metódou tímovej práce je výpočet funkcie $\lfloor \log n \rfloor$ (resp. $\lceil \log n \rceil$), keď je vstupom číslo n. Na vyriešenie tejto úlohy v konštantnom čase stačí $\lfloor \log n \rfloor$ procesorov. i-ty procesor predstavuje tím (jednočlenný) overujúci, či $i = \lfloor \log n \rfloor$ tak, že si pomocou jediného aritmetického posunu spočíta hodnotu $v = \lfloor \frac{n}{2^i} \rfloor$. Ak je $v>0$ a $\lfloor \frac{v}{2} \rfloor = 0$, tak $i = \lfloor \log n \rfloor$. Procesor, pre ktorý platí táto podmienka, zapíše číslo i do nultej bunky spoločnej pamäte ako výstup.

Počítanie $\lceil \log n \rceil$ vyžaduje len malú modifikáciu, ak $n=2^i$, tak $\lceil \log n \rceil = i$, inak $\lceil \log n \rceil = i+1$.

Je zrejmé, že na výpočet $\lfloor \log n \rfloor$ (resp. $\lceil \log n \rceil$) uvedeným algoritmom v konštantnom čase stačí šírka slova $O(\log n)$.

Keď vieme počítať funkciu $\lceil \log n \rceil$, tak môžeme v konštantnom čase nájsť najmenšiu mocninu dvojky väčšiu alebo rovnú n - stačí spočítať $2^{\lceil \log n \rceil}$.

Počítanie maxima z n čísel

Maximum z n čísel sa dá pomocou CRCW siete spočítať v konštantnom čase s $O(n^2)$ procesormi.

Bez ujmy na všeobecnosti môžeme predpokladať, že n je mocninou dvojky (viz. predchádzajúci odstavec), na odhadoch potrebného času a pamäte sa to neprejaví.

Výpočet bude založený na tímovej práci - n^2 procesorov bude rozdelených do n tímov po n procesoroch.

Algoritmom z predchádzajúceho odstavca je možné v konštantnom čase spočítať $\log n$ a pomocou aritmetického posunu aj n^2 (pretože n je mocninou dvojky). Každý z procesorov s identifikačným číslom menším než n^2 sa zúčastní výpočtu. Každý procesor rozdelí svoje identifikačné číslo na dve čísla dĺžky $\log n$ bitov (pomocou aritmetických posunov a odčítania), dostane tak dvojicu čísel (i,j), kde $0 \le i,j < n$, a bude pracovať ako j-ty člen i-teho tímu.

i-ty tím bude na komunikáciu používať pamäťovú bunku $n+i$ (pripomeňme, že bunky 0 až $n-1$ obsahujú vstupné čísla). Nultý procesor i-teho tímu bude **manažérom** a na začiatku vynuluje bunku $n+i$. i-ty tím bude zisťovať, či je vstupné číslo x_i najväčšie zo všetkých. j-ty procesor i-teho tímu porovná x_i a

x_j (pre všetky $0 \le i,j < n$). Ak je x_i menšie ako x_j, tak do $n+i$ zapíše hodnotu 1. Nakoniec manažér tímu i prečíta hodnotu z bunky $n+i$, a ak je to 0, tak zapíše hodnotu x_i do pamäťovej bunky 0 ako výstup.

Je zrejmé, že uvedený algoritmus pracuje v konštantnom čase a vyžaduje šírku slova minimálne $2b+\log n$.

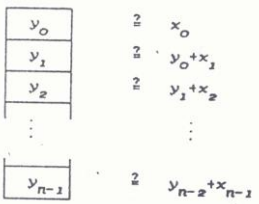
Sčítanie n b-bitových čísel

CRCW sieť so šírkou slova $O(n(b+\log n))$ môže sčítať n b-bitových celých čísel v konštantnom čase.

Bez ujmy na všeobecnosti budeme predpokladať, že n je mocninou dvojky, a že všetky vstupné čísla sú kladné.

Súčet n b-bitových čísel nebude mať viac ako $(b+\log n)$ bitov. Každý procesor potrebuje poznať toto číslo. Hodnota $\log n$ sa spočíta vyššie uvedeným postupom, postupom z predchádzajúceho odseku zistíme najväčšie vstupné číslo a b určíme ako hornú celú časť logaritmu tohto čísla.

Predpokladajme, že číslo $b+\log n$ je mocninou dvojky, inak ho upravíme na najbližšiu vyššiu mocninu dvojky postupom uvedeným vyššie. Potom sa procesory rozdelia na $2^{n(b+\log n)}$ tímov po n procesoroch. Každý tím interpretuje identifikačné číslo tímu ako n-ticu $(b+\log n)$-bitových čísel, $y_0, y_1, \ldots, y_{n-1}$. i-ty člen tímu , $1 \le i < n$, extrahuje y_i a y_{i-1}, 0-tý procesor extrahuje y_0. Na to musí i-ty procesor urobiť posun o $i(b+\log n)$ bitov. Pretože $b+\log n$ je mocninou dvojky, tak aj veľkosť posunu sa dá spočítať jediným aritmetickým posunom. i-ty procesor tímu , pre $1 \le i < n$, kontroluje, či $y_i = y_{i-1} + x_i$, a 0-tý procesor kontroluje, či $y_0 = x_0$ (viz obr. 2.1).



obr. 2.1.

Procesory, ktoré zistia nerovnosti, to ohlásia svojmu tímovému manažérovi prostredníctvom spoločnej pamäte. Práve jeden tím nezistí nerovnosť. Jeho manažér vie, že identifikačné číslo tohto tímu reprezentuje reťazec súčtov prefixov vstupnej postupnosti. Tento procesor extrahuje celkový súčet (posledné číslo n-tice), ktorý nakoniec zapíše do bunky 0 spoločnej pamäte ako výstup.

Všimnime si, že to, že sa jedná o b-bitové čísla ovplyvňuje len potrebnú šírku slova, ale nemá to vplyv na čas potrebný na sčítanie.

Násobenie čísel v konštantnom čase

CRCW sieť so šírkou slova $O(b^2)$ môže vynásobiť dve b-bitové celé čísla v

konštantnom čase.

Použije sa školská metóda násobenia. Pre jednoduchosť predpokladajme, že vstupné čísla x_1 a x_2 sú kladné. Najprv sa spočíta hodnota b ako logaritmus väčšieho vstupného čísla (v konštantnom čase). Potom i-ty procesor, $0 \le i < b$, pracuje nasledovne:

a) Extrahuje i-ty bit čísla x_2 (bity číslujeme od nuly zľava do prava) pomocou posunov a odčítania.

b) Ak hodnota získaná v kroku a) je nenulová, tak urobí aritmetický posun čísla x_1 o i bitov doľava (tj. vynásobí x_1 číslom 2^i) a výsledok zapíše do bunky $i+1$ spoločnej pamäte.

Súčet takto získaných čísel spočíta v konštantnom čase metódou uvedenou v predchádzajúcom odseku.

Odmocňovanie čísel v konštantnom čase

Nech $c>1$ je prirodzené číslo a n je vstupné celé číslo. CRCW sieť so šírkou slova $O(\log^2 n)$ môže v konštantnom čase spočítať $\lceil n^{1/c} \rceil$ takto:

Použijeme n tímov procesorov. Tím i, $0 \le i < n$, kontroluje, či $i = \lceil n^{1/c} \rceil$. Robí to tak, že spočíta i^c (pomocou $c-1$ násobení). K tomuto účelu má každý tím $2^{O(\log^2 n)}$ procesorov (podľa predchádzajúceho odseku). Ak $i^c \ge n$ a $i^{c-1} < n$, potom $i = \lceil n^{1/c} \rceil$.

Sčítanie čísel v konštatnom čase s menšou šírkou slova

Pre naše účely budeme potrebovať ukázať, že CRCW sieť s lineárnou šírkou slova môže sčítať n čísel s konštantnou bitovou dĺžkou v konštantnom čase. Avšak ukážeme silnejší výsledok.

Predpokladajme, že ako vstup máme n kladných celých čísel, každé s $n^{1-1/c}$ bitmi, pre nejaké celé číslo $c \ge 2$. Kvôli prehľadnosti v algoritme vynecháme operácie dolnej a hornej celej časti, ktoré zaručujú, že počítame stále s celými číslami. Súčet týchto čísel a každá čiastočná suma má maximálne $O(n^{1-1/c})$ bitov.

Sieť najprv určí n a spočíta $m = n^{\frac{1}{c+1}}$. Po tomto predvýpočte sa súčet spočíta v dvoch fázach. Použije sa metóda rozdeľ a panuj.

Fáza 1. Rozdeľ vstupné čísla do $\frac{n}{m}$ skupín po m čísel a urob súčet každej skupiny. Po c iteráciách tohto postupu nám zostane $\frac{n}{m^c}$ čiastočných súčtov.

Fáza 2. Sčítaj $\frac{n}{m^c}$ čiastočných súčtov.

Podľa predchádzajúceho odstavca predvýpočet vyžaduje konštantný čas a zanedbateľnú šírku slova (pre dostatočne veľké n). Pomocou starého postupu násobenia sa fázy 1 a 2 dajú urobiť v konštantnom čase. Šírka slova potrebná na prvú je úmerná

$$m n^{1-1/c} = n^{1-\frac{1}{c^2+c}},$$

a na druhú fázu je úmerná

$$\frac{n}{m^c} n^{1-1/c} = n^{1-\frac{1}{c^2+c}}.$$

Teda $n\, n^{1-1/c}$ -bitových čísel sa dá sčítať v konštantnom čase so šírkou slova $O\left(n^{1-\frac{1}{c^2+c}}\right)$.

Teda pre ľubovoľné $0 < \lambda \le \frac{1}{2}$ existuje $\mu > 0$ také, že CRCW sieť so šírkou slova

$n^{1-\mu}$ môže sčítať n $n^{1-\lambda}$-bitových čísel v konštantnom čase.

Tento postup predstavuje ďalšiu z metód návrhu, resp. vylepšovania paralelných algoritmov. Náš prvý algoritmus sčítania n čísel vyžadoval šírku slova $O(n(b+\log n))$, kde b je dĺžka zápisu najväčšieho vstupného čísla. Použitím tohto algoritmu na skupiny "malej" veľkosti sa nám v prvej fáze podarilo zredukovať veľkosť úlohy, a pritom neprekročiť lineárnu šírku slova, a v druhej fáze sme s týmto algoritmom na redukovanej úlohe taktiež vystačili so sublineárnou šírkou slova.

Počítanie funkcií prev a last

Nech $x_i \in \mathbb{N}$, $1 \le x_i \le n$, pre $1 \le i \le n$. Definujme *prev*: $\mathbb{Z}^n \to \mathbb{Z}^n$ nasledovne:
$prev(x_1,\ldots,x_n) = \langle y_1,\ldots,y_n\rangle$, kde $y_i = j$, ak $x_j = x_i$, $j<i$ a $x_k \neq x_i$ pre $j<k<i$, a $y_i = 0$, ak takéto j neexistuje.

Ďalej definujme *last*: $\mathbb{Z}^n \to \mathbb{Z}^n$ nasledovne:
$last(x_1,\ldots,x_n) = \langle y_1,\ldots,y_n\rangle$, kde $y_i = j$, ak $x_j = x_i$ a $x_k \neq x_i$ pre $j<k\le n$, a $y_i = 0$, ak takéto j neexistuje.

Tieto funkcie budeme potrebovať na simulácie Turingových stojov na sieťach so spoločnou pamäťou.

PRAM sieť môže počítať $prev(x_1,\ldots,x_n)$ a $last(x_1,\ldots,x_n)$ v konštantnom čase so šírkou slova $O(\log n)$.

Ukážeme len algoritmus pre *prev*, pre *last* je postup obdobný. Procesory rozdelíme na n^2 tímov, jeden tím pre dvojicu $\langle i,j\rangle$, $1 \le i,j \le n$. Každému tímu vyhradíme pamäťovú bunku, ktorú inicializujeme nulou. k-ty procesor tímu, $j<k<i$ (ostatné procesory tímu nebudú pracovať), overí, či $x_k \neq x_i$. Ak zistí, že $x_i = x_k$, tak do vyhradenej bunky zapíše číslo 1. Všetky tímy to urobia súčasne. Člen tímu s najnižším číslom prečíta túto hodnotu, overí, či je to stále nula a skontroluje, že $x_i = x_j$. Ak to platí, tak zapíše číslo j do i-tej bunky spoločnej pamäte.

Užitočná technická lemma

Nech $T(n,P(n))$ je čas potrebný na výpočet súčtu n čísel (konečnej alebo nekonečnej) pologrupy pomocou $P(n)$ procesorov na sieti so spoločnou pamäťou. Namiesto $T(n,n)$ budeme písať $T(n)$. Hovoríme, že f: $\mathbb{Z} \to \mathbb{Z}$ je konštruovateľná v konštantnom čase, ak sieť so spoločnou pamäťou s n procesormi môže spočítať $f(n)$ zo vstupu n v konštantnom čase. Podľa predchádzajúcich paragrafov je ľubovoľný polynóm s n, $\log n$ a odmocninami z n konštruovateľný v konštantnom čase.

Lemma 2.1: Nech f: $\mathbb{N} \to \mathbb{N}$, $f(n) \le n$, pre všetky $n \ge 0$, je konštruovateľná v konštantom čase. Potom $T(n) \le T\left(\left\lceil \frac{n}{f(n)} \right\rceil\right) + T(f(n),n) + O(1)$.

Dôkaz: Úlohu chceme riešiť pre n čísel s n procesormi. Sieť spočíta $f(n)$. Procesory sa rozdelia do $f(n)$ tímov tak, že každý tím bude mať maximálne $\left\lceil \frac{n}{f(n)} \right\rceil$ procesorov. Každý tím spočíta súčet svojich vstupov v čase $T\left(\left\lceil \frac{n}{f(n)} \right\rceil\right)$ (predpokladáme, že $T(n)$ je monotónna neklesajúca funkcia). Takto dostaneme $f(n)$ čiastočných súčtov, ktoré sa dajú sčítať v čase $T(f(n),n)$. ∎

Pomocou techník použitých pri počítaní súčtu sa dá ukázať, že súčet n čísel z konečnej pólogrupy sa dá spočítať v konštantnom čase pomocou $2^{O(n)}$.

procesorov (Vishkin, Widgerson [22]).

Podľa lemmy 2.1 pre $f(n)=O(\log n)$ a predchádzajúceho tvrdenia sa dá ukázať, že súčet n čísel v konečnej pologrupe sa dá spočítať v čase $O(\log n/\log\log n)$ použitím n procesorov (Reif [19], Parberry [16]).

Lemma 2.1 sa dá aplikovať aj na počítanie maxima z n čísel. Maximum z n čísel sa dá pomocou n^2 procesorov určiť v konštantnom čase (viz začiatok kapitoly). Teda použitím postupu lemmy 2.1 s $f(n)=\lceil\sqrt{n}\rceil$ je čas potrebný na určenie maxima z n čísel pomocou n procesorov

$T(n)\le T(\lceil\sqrt{n}\rceil)+T(\lceil\sqrt{n}\rceil,n) = T(\lceil\sqrt{n}\rceil)+O(1) = O(\log\log n)$.

Ohraničením hĺbky rekurzie na $\lceil\log k\rceil$, pre nejakú konštantu $k\ge 1$, čas môže byť skrátený na $O(\log k)$ s $n^{1+\frac{1}{k}}$ procesormi.

3. Simulácia sekvenčných strojov na sieťach so spoločnou pamäťou

V kapitole 1 sme si položili otázku: ktoré problémy z P sa dajú exponenciálne rýchlejšie riešiť na "rozumnom" paralelnom počítači, tj. takom, ktorý spĺňa paralelnú tézu. Odpoveď znela: "pravdepodobne nie všetky". Tu sa budeme zaoberať jednoduchšou otázkou: aké zrýchlenie vzhľadom k sekvenčným strojom môžeme dosiahnuť na PRAM. Najprv ukážeme, že PRAM s veľkou šírkou slova sú nesmierne silné.

Budeme hovoriť, že funkcia $T\colon \mathbb{N}\to\mathbb{N}$ je ws-konštruovateľná (word size), ak sieť so spoločnou pamäťou so šírkou slova $O(T(n))$ môže počítať $T(n)$ v konštantnom čase zo vstupu n (podľa vyššie uvedených algoritmov je mnoho dôležitých funkcií ws-konštruovateľných).

Lemma 3.1: (Parberry, Schnitger) Nech $T(n)$ je ws-konštruovateľná. Potom PRAM s ohraničenou sadou aritmetických inštrukcií môže simulovať $T(n)$-časovo ohraničený k-páskový Turingov stroj v konštantnom čase a so šírkou slova $O(T(n))$.

Dôkaz: (Náčrt) Máme daný Turingov stroj. Očíslujeme jeho inštrukcie. Procesory rozdelíme do $2^{O(T(n))}$ tímov, jeden tím pre každú postupnosť inštrukcií $r_0, r_1, \ldots, r_{T(n)-1}$ Turingovho stroja. Každý tím bude mať $2^{O(T(n))}$ procesorov. Každý tím najprv určí polohy hláv na všetkých k páskach vo všetkých časových okamžikoch. Použije sa na to metóda čiastočných súčtov z kapitoly 2, ktorá sa aplikuje na postupnosť pohybov hláv odvodenú z postupnosti inštrukcií. Potom sa overí, či:

1. Postupnosť stavov určená postupnosťou inštrukcií je uskutočniteľná. Tj. inštrukcia r_0 vyžaduje počiatočný stav Turingovho stroja a pre $1\le i<T(n)$, ak inštrukcia r_{i-1} prevedie stroj do stavu q, tak inštrukcia r_i musí vyžadovať stav q.

2. Pre každú z k pások a každý časový okamžik t, $0\le t<T(n)$:

(i) Ak v čase t hlava navštívila toto políčko prvý krát, tak symbol čítaný hlavou podľa inštrukcie r_t musí byť symbol, ktorý tam bol v počiatočnej konfigurácii.

(ii) Ak hlava navštívila toto políčko pred okamžikom t naposledy v čase s, tak symbol zapísaný na pásku inštrukciou r_s sa musí zhodovať so symbolom čítaným inštrukciou r_t.

Informácie nutné na toto overovanie sa spočítajú pomocou *prev* z predchádzajúcej kapitoly. Práve jeden tím zistí, že jeho postupnosť inštrukcií je správna. Potom určí, či koncový stav je prijímajúci a výsledok zapíše do bunky 0 spoločnej pamäte. Táto simulácia vyžaduje konštantný čas a šírku slova $O(T(n))$.■

Tento výsledok môže byť zrejmým spôsobom zovšeobecnený na simuláciu deterministických Turingových strojov vyčíslujúcich funkcie (koncová konfigurácia sa určí pomocou funkcie *last*) a na simuláciu nedeterministických Turingových strojov.

Je jasné, že tento výsledok silne závisí na veľkej šírke slova. Ale čo sa stane, ak budeme požadovať, aby platila paralelná téza? Tj. požadujeme, aby sieť so spoločnou pamäťou pracovala v čase $T(n)$ a mala šírku slova $W(n)$, kde $W(n)=T(n)^{O(1)}$. Ukážeme, že ľubovoľné polynomiálne zrýchlenie je možné v prípade šírky slova polynomiálnej vzhľadom k času.

Veta 3.2: Nech B: $\mathbb{N} \longrightarrow \mathbb{N}$ je *ws*-konštruovateľná a PRAM so šírkou slova $W(n)$ môže simulovať $B(n)$ časovo ohraničený deterministický Turingov stroj v čase $C(n)$. Potom PRAM so šírkou slova $O(W(n)+B(n)+\log T(n))$ môže simulovať $T(n)$ časovo ohraničený deterministický Turingov stroj v čase $O(T(n)/B(n)+C(n))$.

Náčrt dôkazu: Nech M je $T(n)$ časovo ohraničený k-páskový deterministický Turingov stroj. Simulácia sa bude skladať z $\lceil T(n)/B(n) \rceil$ fáz, z ktorých každá bude zodpovedať $B(n)$ krokom stroja M.simulácia bude počítať konfigurácie na konci každej fázy.

Zóna je časť pásky, ktorá sa môže zmeniť počas danej fázy, tj. $k(2B(n)-1)$ políčok pásky, ktoré sú vo vzdialenosti najviac $B(n)$ políčok od hlavy na začiatku fázy. V priebehu fázy sa simulácia riadi iba podľa obsahu tejto zóny a na konci fázy sa zóna aktualizuje.

V predvýpočte sa spočíta tabuľka s $2^{O(B(n))}$ položkami, ktorej indexom je počiatočná zóna a položkou je zóna, ktorá vznikne z tejto zóny po $B(n)$ krokoch stroja M. Tento predvýpočet bude trvať $C(n)$ krokov a bude vyžadovať šírku slova $O(W(n))$ bitov.

Na začiatku fázy sa určí počiatočná zóna "vykrojením" úsekov pások do vzdialenosti $B(n)$-1 krokov od polôh hláv. Z tabuľky získanej v predvýpočte sa určí ako bude vyzerať táto zóna po $B(n)$ krokoch a nakoniec sa aktualizuje obsah pások a polohy hláv podľa tejto výslednej zóny. Po tomto je sieť v stave na začiatku novej fázy.

$T(n)$ krokov stroja M sa simuluje v $\lceil T(n)/B(n) \rceil$ fázach. predvýpočet vyžaduje $O(C(n))$ krokov a každá fáza vyžaduje konštantný čas, ak sa použije ohraničená sada aritmetických inštrukcií. Šírka slova potrebná pri simulácii je maximum z $O(W(n)+B(n))$ (pri predvýpočte), $O(B(n))$ (pre identifikačné čísla procesorov pre všetky fázy) a $O(\log T(n))$ (na čítanie uloženej konfigurácie).■

Aplikáciou tejto vety a pomocou lemmy 3.1 zovšeobecnenej na prípad, že Turingov stroj vyčísluje funkciu dostávame nasledujúcu vetu.

Veta 3.3: Nech B: $\mathbb{N} \longrightarrow \mathbb{N}$ je *ws*-konštruovateľná. $T(n)$ časovo ohraničený deterministický Turingov stroj sa dá simulovať v čase $O(T(n)/B(n))$ sieťou so spoločnou pamäťou so šírkou slova $O(B(n)+\log T(n))$.

Dôkaz: $B(n)$ časovo ohraničený Turingov stroj sa dá simulovať v konštantnom čase so šírkou slova $O(B(n))$. Dosadením do predchádzajúcej vety dostaneme žiadaný výsledok.■

Táto veta má niekoľko dôležitých dôsledkov.

Pre pre $B(n)=T(n)^{\varepsilon}$ a využitím toho, že ak je $T(n)$ *ws*-konštruovateľná, tak aj $B(n)$ je *ws*-konštruovateľná dostávame:

Dôsledok 3.4: Ak je $T(n)$ *ws*-konštruovateľná funkcia, potom $T(n)$ časovo ohraničený deterministický Turingov stroj môže byť simulovaný v čase $T(n)^{1-\varepsilon}$, pre ľubovoľné reálne číslo $\varepsilon>0$, sieťou so spoločnou pamäťou, ktorá spĺňa paralelnú tézu.

To znamená, že ľubovoľné polynomiálne zrýchlenie je možné na stroji, ktorý spĺňa paralelnú tézu. To je silný výsledok, pretože sa usudzuje, že nie všetky problémy riešiteľné deterministicky v polynoniálnom čase sa dajú riešiť v polylogaritmickom čase paralelne na modele spĺňajúcom paralelnú tézu.

4. Dolné odhady časovej zložitosti na PRAM

Netriviálne dolné odhady zložitosti bývajú väčšinou veľmi zložité. V tejto kapitole uvedieme dva takéto odhady. Jeden pre CREW PRAM a druhý pre CRCW PRAM. Prvý je dolným odhadom časovej zložitosti výpočtu booleanskej funkcie OR na CREW PRAM a druhý je dolným odhadom časovej zložitosti pre počítanie súčtu n kladných celých čísel na CRCW PRAM

Veta 4.1: Na CREW PRAM výpočet booleanskej funkcie OR z n bitov vyžaduje aspoň $\log_b n$ krokov, kde $b>4.79$.

Dôkaz tejto vety je príliš technický a príliš dlhý na to, aby tu bol uvedený. Je ho možné nájsť v [17]. Tu naznačíme iba jeho ideu:

Hovoríme, že bit i ovplyvňuje procesor p v čase t, ak sa obsah lokálnej pamäte procesora v čase t líši podľa toho, či vstup i je 0 alebo 1, začiaľ čo ostatné vstupné bity sú nulové. Podobne, môžeme hovoriť o tom, že vstupný bit ovplyvňuje bunku m spoločnej pamäte v čase t. Indukciou podľa t sa dá dokázať, že počet vstupných bitov ovplyvňujúcich ľubovoľný procesor alebo pamäťovú bunku v čase t je najviac c^t, kde c je vhodná konštanta. Pretože všetky vstupné bity musia ovplyvniť výstupnú bunku na konci výpočtu, tak výpočet musí mať aspoň $\log_c n$ krokov.

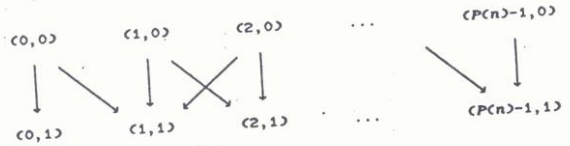
Úplný dôkaz uvedieme pre dolný odhad na CRCW PRAM. Kvôli prehľadnosti ukážeme najprv dolný odhad pre tzv. siete s ohraničeným prístupom a potom naznačíme ako tento dôkaz modifikovať, aby sme dostali dolný odhad pre sieť so spoločnou pamäťou.

Siete s ohraničeným prístupom nemajú spoločnú pamäť a jednotlivé procesory môžu komunikovať iba prostredníctvom špeciálnych komunikačných registrov. Každý procesor má práve jeden komunikačný register, ktorý je prístupný (pre zápis aj čítanie) pre všetky procesory v sieti. Ostatné registre jednotlivých procesorov sú lokálne a pre ostatné procesory neprístupné.

Veta 4.2: Nech $P(n)$ je ľubovoľná funkcia. Sieť s ohraničeným prístupom s $P(n)$ procesormi potrebuje na sčítanie n čísel čas aspoň $\lceil \log_3 n \rceil$.

Dôkaz: Nech $\mathbb{M}$ je sieť s $P(n)$ procesormi, ktorá dokáže sčítať n čísel v čase $T(n)$, $x=\langle x_0, x_1, \ldots, x_{n-1}\rangle$ je vstupná n-tica čísel menších alebo rovných $\mathbb{N}$. Predpokladáme, že procesory sú očíslované $0,1,\ldots,P(n)-1$ a výstup bude na konci výpočtu zapísaný do pamäte procesoru 0.

Nech G_x je orientovaný graf s vrcholmi $\langle p,t\rangle$, $0\le p<P(n)$, $0\le t<T(n)$ a z vrcholu $\langle p_1,t_1\rangle$ vedie hrana do vrcholu $\langle p_2,t_2\rangle$, ak $t_2=t_1+1$ a buď $p_1=p_2$ alebo v kroku t_1 výpočtu siete $\mathbb{M}$ na vstupe x buď procesor p_2 čítal hodnotu od p_1, alebo p_1 úspešne zapísal hodnotu do pamäte procesoru p_2. G_x sa nazýva **komunikačným grafom** siete na vstupe x (na obr. 2.2 je naznačená "prvá vrstva" takéhoto grafu). i-ty prvok n-tice x sa nazýva **dosiahnuteľný**, ak v G_x existuje cesta z vrcholu $\langle i,0\rangle$ do vrcholu $\langle 0,T(n)\rangle$.



obr. 2.2.

Nie je nutné, aby všetky prvky vstupnej n-tice boli dosiahnuteľné, pretože informácia sa môže šíriť od procesora aj tým, že procesor sa rozhodne nezapisovať (viz. počítanie maxima). Avšak ukážeme, že za predpokladu, že $\mathbb{N}$ je dostatočne veľké, existuje vstupná n-tica, ktorej všetky prvky sú dosiahnuteľné.

Pre spor predpokladajme, že každý vstup má nejaký nedosiahnuteľný prvok. Definujme **dosiahnuteľnú množinu** n-tice x ako $\mathbb{R}_x=\{i \mid x_i$ je dosiahnuteľné$\}$. Nech $\mathbb{Q}_x\subseteq\{0,1,\ldots,n-1\}$ je také, že $|\mathbb{Q}_x|=n-1$ a $\mathbb{R}_x\subseteq\mathbb{Q}_x$. Potom existuje jediné i také, že $0\le i<n$ a $i\notin\mathbb{Q}_x$. Pre určitosť predpokladajme, že $\mathbb{Q}_x$ je zvolené tak, aby i bolo minimálne. $\mathbb{Q}_x$ budeme nazývať **kritickou množinou** pre x, $\langle x_0,\ldots,x_{i-1},x_{i+1},\ldots,x_{n-1}\rangle$ budeme nazývať **kritickým reťazcom** pre x a x_i **nedosiahnuteľným symbolom** pre x.

Predpokladajme, že máme pevne zvolené x. Koľko existuje vstupov y takých, že $G_x=G_y$? Ak $G_x=G_y$ potom zrejme $\mathbb{R}_x=\mathbb{R}_y$ a kritická množina pre x je takisto kritická pre y. Predpokladajme, že existujú dva vstupy y_1 a y_2 s identickými

kritickými reťazcami také, že $G_x = G_{y_1} = G_{y_2}$. Potom by sa y_1 a y_2 mohli líšiť jedine nedosiahnuteľným prvkom vstupnej n-tice s najnižším indexom. Avšak súčet vstupných čísel nezávisí od nedosiahnuteľných prvkov, teda aj tieto prvky sa musia zhodovať. teda $y_1 = y_2$ a počet kandidátov na y je nanajvýš toľko, koľko je rôznych kritických reťazcov. Rôznych kritických reťazcov je N^{n-1}, a preto najviac N^{n-1} vstupov môže mať zhodný komunikačný graf (pripomeňme, že vstupné čísla sú z množiny $\{1,\ldots,N\}$).

Nech $G(n)$ je počet rôznych komunikačných grafov pre n vstupov. Podľa Dirichletovho princípu aspoň jeden graf musí byť komunikačným grafom pre aspoň $\frac{N^n}{G(n)}$ vstupných n-tíc. Ak je N zvolené tak, že $N > G(n)$, tak je táto hodnota väčšia ako N^{n-1}, čo je v spore s predchádzajúcim odstavcom. Teda musí existovať vstup, ktorého všetky prvky sú dosiahnuteľné. Pretože všetky uzly v G_x majú vstupný stupeň maximálne 3, tak $T(n) \geq \lceil \log_3 n \rceil$.

Aby sme mohli odhadnúť veľkosť N, potrebujeme poznať počet rôznych komunikačných grafov pre n vstupov - $G(n)$. Každý komunikačný graf má $T(n)$ vrstiev, kde každá vrstva zodpovedá jednému kroku výpočtu siete. Odhadneme počet možností pre jednu vrstvu. Pre operáciu čítania existuje $P(n)^{P(n)}$ možností. Podgraf zodpovedajúci operácii zápisu je bipartidným párovaním. Takýchto párovaní medzi dvomi množinami veľkosti $P(n)$ nie je viac než $P(n)^{P(n)+O(1)}$. Teda celkovo $G(n) \leq (P(n)^{2P(n)+O(1)})^{T(n)}$. Tým dostávame aj odhad pre N a dostávame silnejšie tvrdenie, stačí uvažovať súčet n čísel veľkosti $O(P(n)(\log P(n))(\log n))$ bitov.

Veta 4.2 sa dá zovšeobecniť pre ľubovoľnú funkciu n premenných taký, že keď zvolíme pevne $n-1$ vstupov a výstup, tak je zvyšný vstup určený jednoznačne.

Veta 4.3: Súčet n ľubovoľných celých čísel na CRCW PRAM so spoločnou pamäťou vyžaduje čas aspoň $\lceil \log n \rceil$ nezávisle od ohraničenia na počet procesorov (tj. ak ohraničíme počet procesorov na $P(n)$, kde $P(n)$ je ľubovoľná funkcia, a neohraničíme veľkosť vstupných čísel).

Dôkaz: Idea dôkazu je tá istá ako u predchádzajúcej vety. Stačí si uvedomiť, že sieť s $P(n)$ procesormi, ktorá pracuje $T(n)$ krokov, použije maximálne $P(n)T(n)$ rôznych buniek spoločnej pamäte. Komunikačný graf sa zostrojí nad množinou uzlov tvaru (b,t), kde b je číslo pamäťovej bunky a t je krok výpočtu.

5. Paralelní rozpoznávání bezkontextových jazyků

V poslední části naší přednášky se zabýváme paralelními algoritmy pro rozpoznávání bezkontetových jazyků, a to ze dvou důvodů.

Rozpoznávání bezkontextových jazyků je důležitá úloha. Na tom, jak kvalitní řešení pro ni máme k dispozici, záleží, jak kvalitní syntaktické analyzátory v překladačích jsme schopni vytvořit, jak efektivní algoritmy pro