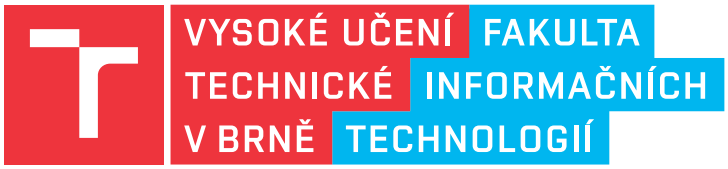


Dokumentácia projektu č. 2
Predmet: **PRL** 2018/19
Projekt: **Výpočet úrovne vrcholu**
Autor: **Michael Halinár** xhalin01

1 Rozbor

Tento paralelný algoritmus potrebuje pri danom binárnom strome 2*n-2 procesorov, pričom n je počet uzlov binárneho stromu. Postup pre vytváranie prechodu je nasledovný. Ako prvé sa vytvoria zoznamy hrán podľa daného vstupu. Následne zoznam susednosti. V tomto zozname má koreň 2 susedov, uzol má 3 susedov a listy majú iba jedného suseda. Po vytvorení zoznamu susednosti nasleduje vytvorenie eulerovej cesty, kde každý procesor zistí svojho následníka. Potom prichádza na rad vytvorenie váh hrán, ak hrana ide z uzla s vyšším indexom do uzla s nižším indexom ide o hranu doprednú a dostáva váhu 1, inak váhu -1. Teraz uzly majú všetko potrebné pre vytvorenie sumy suffixov. Začíname od prvej hrany a robíme súčet váh hrán. Po vypočítaní sumy suffixov už vypočítame úrovne jednotlivých uzlov podľa vzorca: $level(v) = suffix + 1$, kde hrana (u, v) je dopredná. Jednotlivé hodnoty sú posielané procesoru s číslom 0, ten vypíše výsledné úrovne.

2 Analýza

Pole etour $O(c)$, vypočítanie váh $O(c)$, suffix $O(log \cdot n)$, korekcia $O(c)$. Časová náročnosť: $O(logn)$. Priestorová náročnosť: $p(n) = 2 \cdot n - 1 = O(n)$.

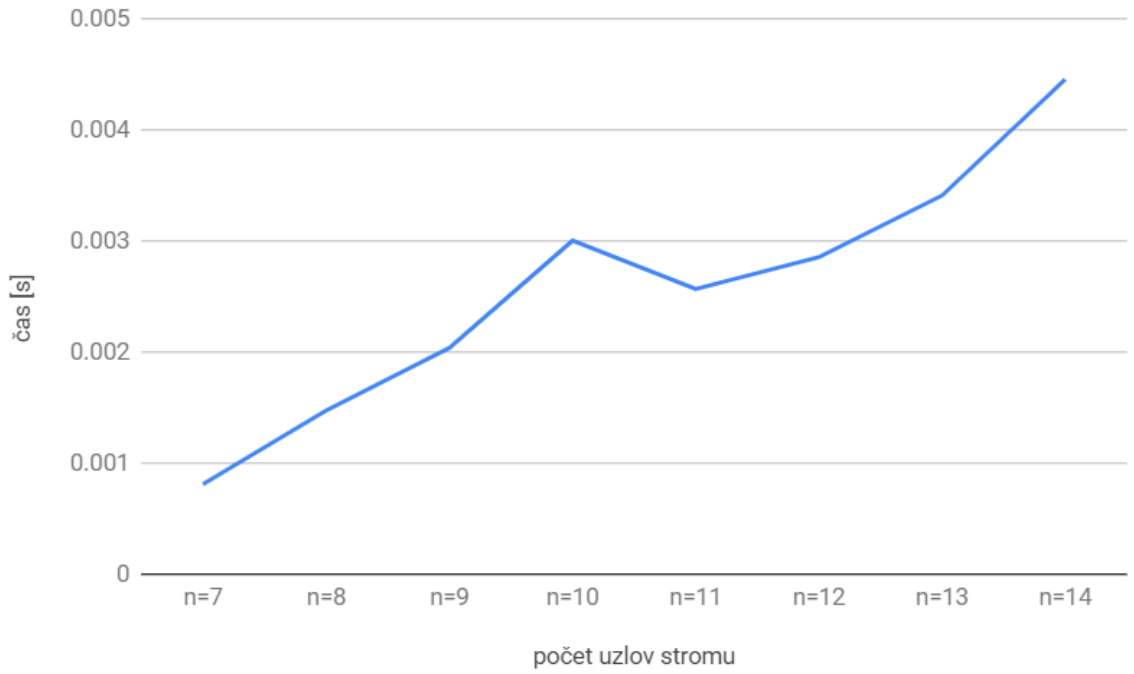
Celková cena: $c(n) = O(log\, n) \cdot (n)$.

3 Implementácia

Po rozparsovaní binárneho stromu sú hrany uložené do štruktúry ktorá ju reprezentuje uzlom z ktorého vychádza a uzlom do ktorého vchádza. Zároveň je vytvorená aj reverzná hrana. Tieto hrany sú uložené do ďalšej štruktúry ktorá tieto dve hrany zoskupuje, taktiež je v tejto štruktúre obsiahnutý ukazateľ ktorý v prípade takom že hrana má následníka, priradí tohoto následníka do ukazateľa `next`. Ďalšou položkou v tejto štruktúre je unikátne ID hrany. Tieto štruktúry vložíme do vektora hrán a tým sme vytvorili zoznam susednosti. Nasleduje výpočet hrany etour čiže hrany následníka. V cykle prechádzame zoznam susednosti ak reverzia hrany má nejakého následníka, uložíme ho do premennej myEtour, Inak sa jedná o listový uzol a ako jeho následníka nastavime jeho reverznú hranu. Ďalej si každá hrana zistí svoju váhu podľa indexov uzľov z ktorého vychádza a do ktorého vchádza. V tejto fáze algoritmu je potrebné rozposlanie poľa váh a poľa následníkov, čiže každá hrana pošle svoju hodnotu procesoru s číslom 0, ten vytvorí polia rozpošle ich znovu všetkým hranám. Takže každý procesor bude vedieť kto má akého následníka a kto má akú váhu. Nasleduje výpočet sumy suffixov, ten prebieha paralelne pre každý procesor. Výpočet je získaný z poľa váh kde postupuje od začiatku a každá nasledujúca hodnota je hodnotou predchádzajúcou + hodnotou ktorá sa nachádza na danom indexe. Podľa sumy suffixov teraz vypočítame úroveň uzlov do ktorých vedú dopredné hrany podľa hore uvedeného vzorca. Vypočítané hodnoty sú poslané procesoru ktorý má ID 0, ten vypíše svoju úroveň 0, a následne všetky ostatné úrovne.

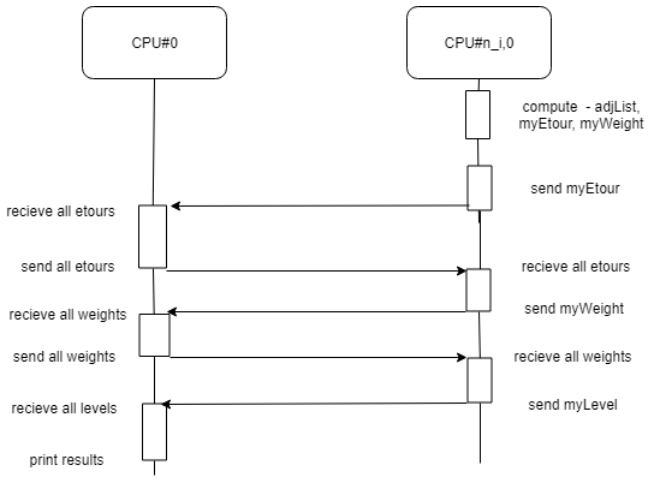
4 Experimenty

Pre meranie času bola použitá funkcia MPI_wtime(). Boli merané časy pre 7 až 14 uzlov. Keďže referenčný server na ktorom sa spúšťa program by viac procesorov nepovolil. Pri každej veľkosti stromu bolo spustených 5 behov a z nich urobený priemer.



5 Komunikačný protokol

Tento komunikačný protokol popisuje komunikáciu medzi jednotlivými procesmi. Procesor s číslom 0 vykonáva rovnaké činnosti ako ostatné procesory, naviac ale zhromažďuje údaje od ostatných a na konci vypíše postupnosť.



6 Záver

Tento algoritmus bol testovaný na školskom serveri Merlin ktorý povoľuje používanie maximálne 26 procesorov, čo pre nás znamená binárny strom o maximálnej dĺžke 14 uzlov a teda nemôžeme využiť plný potenciál tohoto algoritmu.