Přiřazení pořadí preorder vrcholům

Popis algoritmu

Algoritmus nejprve zpracuje vstupní posloupnost zadanou parametrem, z této posloupnosti vytvoří strom, kde každý uzel má levého potomka na pozici 2*i a pravého potomka na pozici (2*i)+1. Z nově vytvořeného stromu vytvoří matici sousednosti a (2*n)-2 hran, kde každou hranu zpracovává paralelně jeden procesor. Procesory vytvoří Eulerovu cestu , na jejíž základě pomocí Suffix Sums zjistí preorder pořadí vrcholů.

Obecný kód algoritmu

1. Řídící proces zpracuje vstupní posloupnost a vytvoří strom
2. Řídící prvek vytvoří seznam sousednosti všech hran a tyto hrany zašle na ostatní procesory, včetně jeho inverzní hrany a předchůdce (předchůdce, kvůli budoucímu zpracování Sumou Sufixů)
3. Každý procesor paralelně zjistí díky inverzní hraně svého následníka. A začne probíhat Suffix Sums
4. Suffix Sums:
   4.1. Každý prvek obdrží váhu jeho následníka a přičte ji, ke své váze. Zároveň zjistí následníka svého následníka a toho nastaví za svého
   4.2. Každý prvek obdrží předchůdce svého předchůdce, jehož si nastaví jako svého vlastního předchůdce
   4.3. Cyklus se opakuje do doby, dokud 1. procesor nemá za následníka poslední procesor
5. Každý proces vypočítá své preorder pořadí a zašle ho na 1. procesor
6. Řídící procesor vypíše vstupní hodnoty v preorder pořadí

Analýza algoritmu

Na začátku algoritmu 1. proces projede celkově 3x každý uzel, tím pádem složitost 1. kroku a 2. kroku je *O(3n)*. Následně již procesory pracují paralelně a Eurelova cesta (krok 2) se zjistí v konstantním čase *O(1)*. Algoritmus Sufix Sums proběhne v logaritmickém čase *O(log n). 5. krok proběhne v konstantním čase a 6. krok proběhne v lineárním čase.*

- Časová složitost - $t(n)$= *O(log n)+ O(3n) + O(*n*) = O(log n) + O(4n)*
- Prostorová složitost – *p(n) = O(2*n)-2*
- Cena algoritmu - $c(n)$= *O(4n)+ O(log n) + O(2*n)-2*

Implementace

Pro implementaci byl využit jazyk C++ a knihovna OpenMPI.

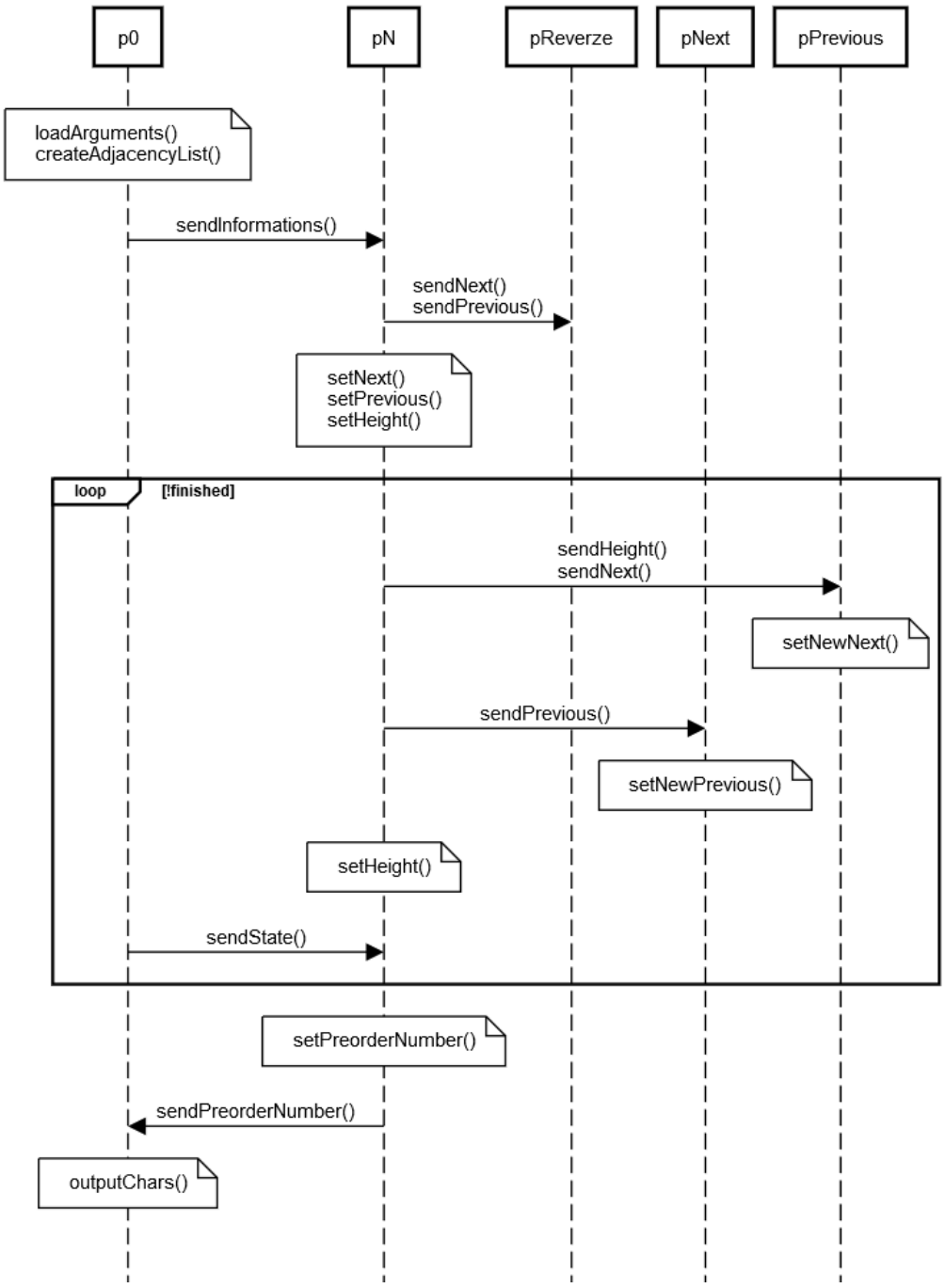
Po spuštění programu začne řídící procesor (procesor 0) číst vstupní data. Z těchto dat vytvoří ve 2 krocích několik vektorů obsahující údaje o každém uzlu (hrana k rodiči/potomkům, včetně jejich inverzních hran). 2 Kroky jsou nutné, jelikož při vytváření hrany k potomku není známa inverzní hrana. Z těchto polí je reprezentována matice sousednosti a všechny potřebné hodnoty jsou zaslány na jednotlivé procesory (inverzní hrana, následník, předchůdce, uzel (kam hrana směřuje), atd.).

Následně každý procesor zašle/přijme údaje do/z inverzní hrany a nastaví svého následníka na základě určitých podmínek. Tím je znát následník v Eulerově cestě, díky které může začít algoritmus Suffix Sums(Následník již zná svého předchůdce, proto ví odkud přijmout potřebnou hodnotu). Zároveň se každému procesoru nastaví váha na 1 pokud obsahuje uzel(jinak váha zůstane na hodnotě 0).

Algoritmus Suffix Sums funguje způsobem, že každý neprvní procesor posílá svého předchůdce svému následovníku a ten posílá zpátky svého následovníka, včetně svojí váhy. Původní prvek si sečte váhu a čeká na zprávu od řídícího procesoru, zda je cyklus kompletní. Cyklus je kompletní v momentě, kdy 1. procesor má za následníka poslední procesor.

Poté co je Sufix Sums kompletní vypočte se hodnota preorderu u dopřených hran (hrana obsahuje uzel) pomocí vzorce (početUzlů-váha+1) a tím se určí pořadí uzlu na výstupu. Tento výstup se zašle na řídící procesor, který tyto data zpracuje a vytvoří z nich výstup.

Komunikační protokol



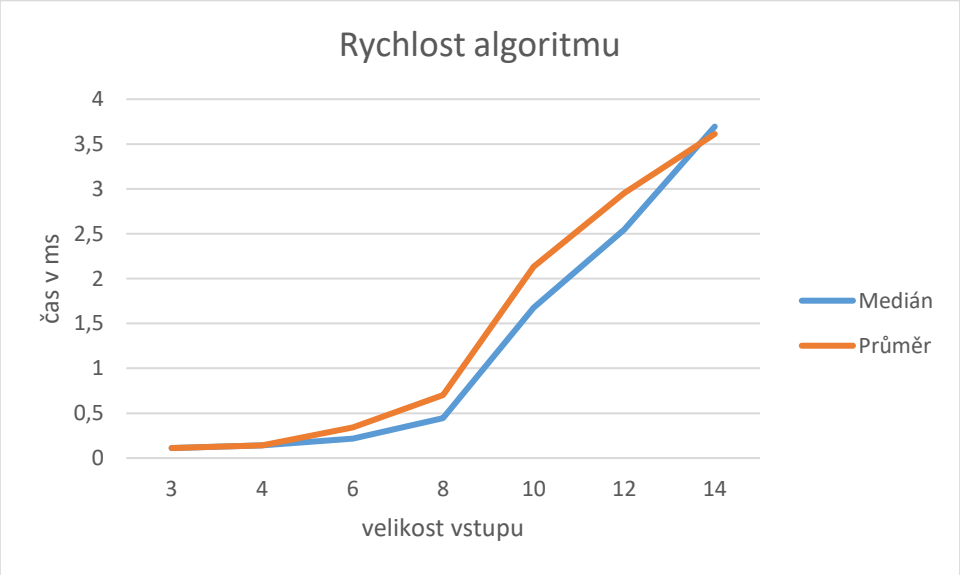
Obrázek 1 - Sekvenční diagram algoritmu

Naměřené výsledky

Pro měření výsledků jsem použil rozdíl proměnných typu *MPI_Wtime()*. V programu je měřena pouze paralelní část algoritmu, v měření tedy není započítáno zpracování vstupních dat, vytvoření matice sousednosti a závěrečný výpis dat. Měřena je tedy pouze Eulerova cesta a Suffix Sums.

Během měření jsem měnil počet vstupů a celkem jsem měřil od vstupu velikosti 3, až po vstup velikosti 14 (jelikož maximální možný počet procesorů, s kterým je školní portál Merlin schopen pracovat je 26 procesorů). Celé měření probíhalo na školním portálu Merlin. Každá hodnota byla měřena celkově 10x a do grafu byly uvedeny průměr a medián těchto hodnot. Čas je veden v mS.

Algoritmus pracuje podle očekávání, tedy čím větší vstup, tím je algoritmus pomalejší. Rozdíl mezi vstupem o velikosti 3, byl zpracován přibližně 17 krát rychleji, než nejdelší možný vstup (tedy vstup o velikosti 14).



Závěr

Časová složitost měřené části algoritmu, by teoreticky měla být v logaritmickém čase, jelikož Eulerovu cestu je možné zjistit v konstantním čase a Suffix Sums v logaritmickém čase. Použitý počet kroků v Suffix Sums této předpovědi odpovídají. Nicméně naměřené hodnoty v grafu ukazují spíše na lineární časovou složitost. Tím pádem časová složitost algoritmu neodpovídá teoretické předpovědi. To může být způsobeno pomalým předáváním hodnot mezi jednotlivými procesory.