1 Úvod

Cieľom projektu je pomocou knižnice Open MPI implementovať v jazyku C/C++ algoritmus Priradenie poradie preorder vrcholom, ako bol definovaný na prednáškach predmetu PRL.

1.1 Vstup

Vstupom je reťazec reprezentujúci uzly binárneho stromu. Indexujeme od 1, ak je uzol na pozícií i, jeho ľavý potom je na pozícií 2i, pravý potomok na pozícií 2i+1.

1.2 Výstup

Výstupom sú rovnaké znaky ako na vstupe ale v preorder poradí.

2 Rozbor a analýza algoritmu

Preorder je spôsob prechodu stromom, v ktorom najprv navštívime otcovský uzol, a potom obidvoch jeho synov. Algoritmus sa skladá z viacerých krokov.

1. Spočítame Eulerovsku cestu daného stromu. Každú hranu stromu nahradíme dvojicou hrán, tým pádom vytvoríme orientovaný graf. Pre každú hranu vypočítame funkciu následníka Etour. Táto funkcia každej hrane priradí nasledujúcu hranu v grafe.
2. V poli Etour hranu vedúcu do koreňa upravíme tak, že vedie sama do seba, tým rozsekneme Eulerovsku kružnicu.
3. Spočítame pole Rank, ktoré určuje pozíciu danej hrany pri prechode preorder daným stromom. Na výpočet použijeme algoritmus ListRanking. Tento algoritmus postupne zvyšuje hodnotu danej hrany, až kým sa nedostane na koniec grafu.
4. V poli Rank máme pozíciu od konca, otočíme teda poradie takto: rank[i] = počet hrán - rank[i].
5. Každej hrane priradíme hodnotu weight, tá je 1 ake je to dopredná hrana, inak je hodnota 0.
6. Vypočítame sumu suffixov podľa poľa Etour a hodnôt weight. Vo výsledom poli SuffixSum budú všetky hrany s hodnotou weight 1 mať napísanú vzdialenosť od konca. Na výpočet použijeme algoritmus suffix sum.
7. Vypočítame hodnotu preorder pre každý výstupný uzol každej hrany tak, že ak sa jedná o doprednú hranu, hodnota preorder pre daný uzol je: počet uzlov - SuffixSum[hrana] + 1. Do poľa preorder na prvú pozíciu vložíme koreň stromu.

2.1 Analýza zložitosti

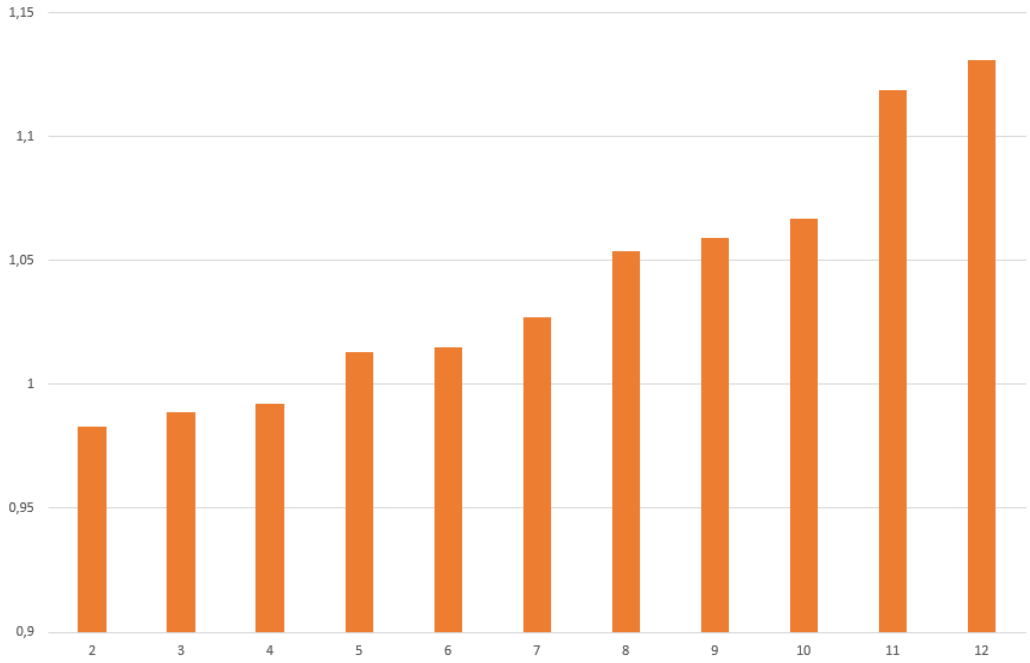
1. Spočítanie Etour: `O(c)`
2. Rozseknutie kružnice: `O(1)`
3. Spočítanie Rank: `O(log n)`
4. Otočenie Rank: `O(1)`
5. Vyrátanie weight: `O(1)`
6. Spočítanie SuffixSum: `O(log n)`
7. Výpočet preorder: `O(c)`

3 Implementácia

Algoritmus bol implementovaný v jazyku C++ za pomoci knižnice Open MPI. Na začiatku prebieha analýza vstupu, vytvorenie zoznamu hrán podľa vrcholov. Každej hrane je priradená hodnota weight podľa toho či je to dopredná hrana alebo nie. Je vytovrení zoznam následníkov adjacency list pre ďalšie použitie. Ďalej prebieha paralelný výpočet poľa Etour pre vyrátanie následníkov pre každú hranu. Výsledky zo všetkých procesorov psolu agreguje procesor 0. Ďalším krokom je paralelný výpočet hodnôt Rank, opäť výsledky agreguje procesor 0. Nakoniec sa vypočítava suma suffixov a z nej pole preorder. Pomocou poľa preorder na sa výstup vypíše poradie vrcholov preorder.

4 Experimenty a namerané hodnoty

Experimenty prebiehali na školskom serveri **merlin**. Testy prebiehali na vstupnom reťazci dlhom od 2 do 12 znakov. Experimenty prebiehali opakovane, a do nasledujúceho grafu boli zanesené priemerné hodnoty. Na vodorovnej osi je počet uzlov, na yvislej osi čas výpočtu. V grafe je čas výpočtu v milisekundách. Väčší počet uzlov nepovoľuje server.

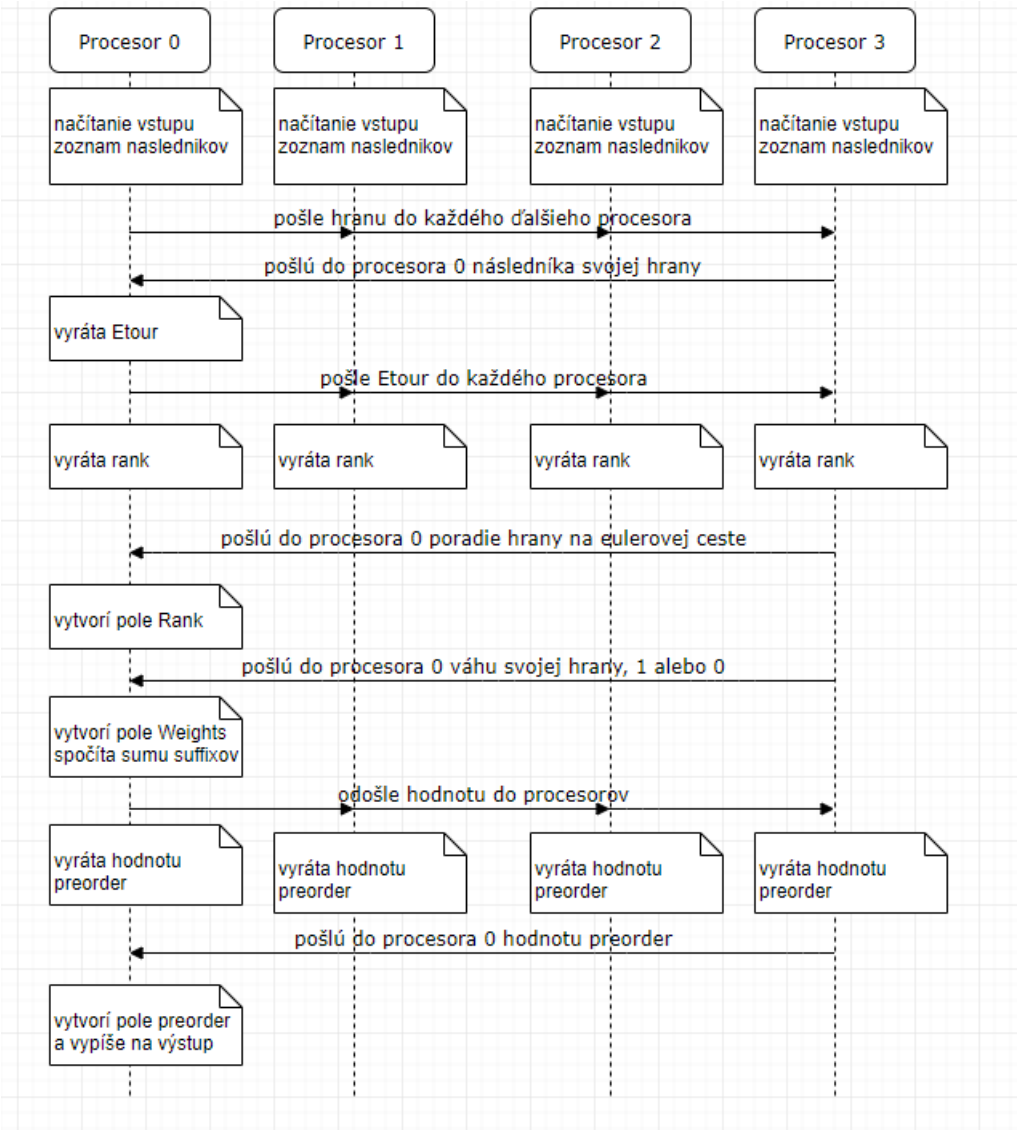


Obr. 1: Graf dĺžky výpočtu

Z grafu a nameraných hodnôt vyplýva, že pre väčší počet uzlov výpočet trvá dlhšie, čo je očakávané chovanie. Experimenty považujem za úspešné.

5 Komunikačný protokol

Nasledujúci komunikačný protokol zhruba popisuje, ako si procesory posielajú medzi sebou správy. Diagram popisuje komunikáciu pre príklad, kde na vstupe je reťazec o 3 znakoch, teda sú využité 4 procesory.



Obr. 2: Komunikačný protokol

6 Záver

V projekte bol úspešne implementovaný algoritmus Priradenie hodnot preorder vrcholom pomocou knižnice Open MPI. Následne bola overená časová zložitosť algoritmu. Z experimentov vyplynulo, že časová zložitosť implementovaného algoritmu približne odpovedá teoreticky spočítanej zložitosti.