PRL – Projekt 2 Přiřazení pořadí preorder vrcholům

Jedná se o algoritmus pracující na stromové struktuře reprezentované polem. Jádrem algoritmu je suma sufixů a výstupem je pořadí jednotlivých uzlů stromu při preorder průchodu.

Algoritmus

Vstupem algoritmu je Eulerova cesta pro daný strom a algoritmus používá jeden procesor pro každou hranu ve stromě. Algoritmus se skládá ze tří hlavních kroků. Jako první se paralelně pro každou hranu vypočte váha (*weight*), která je 1, pokud se jedná o dopřednou hranu a jinak 0. Následně se spočítá suma sufixů nad polem hodnot vah s podle pole následníků Eulerovy cesty. V posledním kroku paralelně všechny procesory s dopřednou hranou vypočtou výsledné preorder pořadí pro index daný cílovým uzlem dané hrany.

Analýza složitosti

První a třetí krok algoritmu jsou prováděny pro jednotlivé hrany paralelně, jejich složitost je proto konstantní. Celková složitost algoritmu proto odpovídá sumě prefixů, která má časovou složitost $t(n) = O(\log n)$ za použití n procesorů. Celková cena algoritmu je proto $c(n) = O(n \log n)$.

Implementace

Cílem projektu je implementovat algoritmus pro libovolný binární strom na vstupu reprezentovaný řetězcem, kde každý znak reprezentuje hodnotu jednoho uzlu, a následně vypsat hodnoty v preorder pořadí. Pro implementaci byl použit jazyk `C++` s knihovnou `Open MPI`.

Počet potřebných procesorů vyplývá z počtu uzlů vstupního stromu (délka řetězce) n, jedná se o počet orientovaných hran binárního stromu, tedy $2n - 2$. Program očekává přesně tento počet procesů, pokud je nemá k dispozici, ukončí se. Reprezentace stromu má pro každý uzel na pozici i jeho levého potomka na pozici $2i$ a pravého na pozici $2i + 1$ indexováno od 1. V rámci programu se ale jinak vše indexuje standardně od 0.

Hrany se přiřazují v pořadí doleva dopředná, doleva reverzní, doprava dopředná, doprava reverzní po jednotlivých úrovních ve směru zprava doleva. Na základě toho je možné pro každý procesor dopočítat, do kterého uzlu směřuje jeho hrana. Pro tento uzel se následně sestaví seznam sousednosti. Podle indexu hrany (*rank* procesoru) se zjistí index reverzní hrany a vypočítá se Eulerova cesta. Podle indexu hrany se také zjistí, jestli je hrana dopředná, a přiřadí se jí odpovídající váha.

Před výpočtem sumy sufixů se provede několik dalších přípravných kroků. Každý proces s hranou, která není koncem Eulerovy cesty odešle svému následníkovi svůj index. Každý proces kromě prvního přijme index svého předchůdce. Procesy bez předchůdce nebo následníka mají uloženou speciální hodnotu, která signalizuje, že tímto směrem nemají komunikovat (`MAX_INT`). Poté se zahájí vnitřní cyklus sumy sufixů od 0 po $\log_2 n$. Každý proces který má následníka mu odešle index svého předchůdce a přijme od něj jeho váhu a index jeho následníka. Každý proces, který má předchůdce, přijme index jeho předchůdce a odešle svou váhu a index svého následníka. Popsaná výměna funguje díky tomu, že spolu v jednu chvíli mohou komunikovat vždy jen dva procesory. Speciální hodnoty zamezí odesílání zbytečných zpráv, především zabrání cyklení na hodnotě 0 u koncové hrany, kde by bylo nutné dopočítávat počet tázajících se procesorů podle indexu cyklu. Korekci hodnot na konci není nutné provádět, protože poslední hrana je vždy zpětná a má tedy váhu 0.

V posledním kroku každý proces s dopřednou hranou vypočítá index v preorder průchodu jako $n - weight$ a odešle ho procesu s odpovídající pozicí danou indexem uzlu, do kterého směřuje hrana. Proces *root* nakonec sesbírá pomocí *gather* operace všech n indexů preorder průchodu (provede se všemi procesy, ale hodnoty na vyšších indexech nemají význam). Poté se hodnoty uzlů podle hodnoty preorder indexů umístí na správnou pozici a vypíšou.

Závěr

V některých bodech byl program implementován mírně odlišně, než jak jsou definovány použité algoritmy. V rámci programu se indexuje od 0, podle čehož je upraven výpočet hodnot v jednotlivých výpočtech indexů. Z algoritmu sumy prefixů byla vypuštěna teoretická část, kdy procesy po dosažení poslední hodnoty do konce cyklu přičítaly hodnotu 0. Také nebyla implementována korekce hodnoty po skončení cyklu, jelikož pro řešený problém nebyla potřebná. Zadaný algoritmus se podařilo úspěšně implementovat.