PRL – Paralelní a distribuované algoritmy
Přiřazení pořadí preorder vrcholům

Jméno a příjmení: Klára Nečasová
Login: xnecas24
Akademický rok: 2017/2018

Úvod

Cílem projektu bylo pomocí knihovny Open MPI a v jazyce C++ implementovat úlohu Přiřazení pořadí preorder vrcholům.

Rozbor a analýza algoritmu

Vstupem paralelního algoritmu pro přiřazení pořadí preorder vrcholům je n symbolů, které reprezentují vrcholy binárního stromu. Algoritmus používá $2n - 2$ procesorů, přičemž každý z nich se stará o jednu orientovanou hranu orientovaného grafu, který byl vytvořen z binárního stromu. Algoritmus se skládá z několika kroků:

1. výpočet následující hrany v Eulerově cestě,
2. přiřazení váhy hraně,
3. výpočet součtu suffixů (suffix sum),
4. výpočet pořadí preorder vrcholu,
5. zaslání výsledku hlavnímu procesoru.

Analýza složitosti algoritmu

Počet procesorů je dán vztahem: $p(n) = 2n - 2$.

1. V prvním kroku každý procesor vypočítá následující hranu Eulerovy cesty, a to s časovou složitostí $\mathcal{O}(c)$.
2. Přiřazení váhy každé hraně ve druhém kroku má také časovou složitost $\mathcal{O}(c)$.
3. Ve třetím kroku každý procesor provede výpočet součtu suffixů, a to pomocí $\lceil \log_2(2n - 2) \rceil$ iterací, takže časová složitost je $\mathcal{O}(\log(n))$.
4. Výpočet pořadí preorder vrcholu a jeho odeslání, které se provádí ve čtvrtém kroku, provede polovina procesorů, odpovídající počtu vrcholů, s časovou složitostí $\mathcal{O}(c)$.
5. V pátém kroku přijme hlavní procesor s $myId = n-1$ pozici v preorder průchodu od poloviny procesorů, což má časovou složitost $\mathcal{O}(n)$.

Jelikož pátý krok již není součástí algoritmu pro přiřazení pořadí preorder vrcholům, lze celkovou časovou složitost algoritmu vyjádřit vztahem:

$$t(n) = 3 \cdot \mathcal{O}(c) + \mathcal{O}(\log(n)) = \mathcal{O}(\log(n)).$$

Celková cena algoritmu:

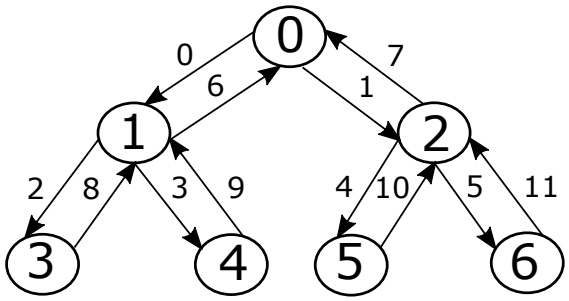
$$c(n) = t(n) \cdot p(n) = \mathcal{O}(\log(n)) \cdot (2n - 2) = \mathcal{O}(n \log(n)).$$

Každý procesor si uloží vstupní řetězec (hlavní procesor si navíc vytvoří pole pro výpis pořadí preorder), což zabírá $\mathcal{O}(n)$ paměti. Dále si uloží váhu a součet suffixů, které zabírají $\mathcal{O}(c)$ paměti. Každý procesor tedy využije přibližně $\mathcal{O}(c \cdot n)$ paměti.

Implementace

Algoritmus byl implementován s využitím knihovny Open MPI v jazyce C++. Implementace vychází z algoritmu uvedeného na přednáškách předmětu PRL. Liší se v tom, že nebylo potřeba implementovat algoritmus List ranking, nalezení pozice hrany, orientace hrany a vytvářet seznam sousednosti (místo výpočtu celé Eulerovy cesty byla vypočítána jen následující hrana dané hrany). Zmíněné kroky bylo možné vypustit díky zvolenému číslování hran a vrcholů v orientovaném grafu.

Nejdříve je zjištěna délka vstupu, která odpovídá počtu vrcholů n v binárním stromu, jež je však reprezentován orientovaným grafem. Pro vstupní řetězec platí, že levý potomek vrcholu i je na pozici $2i$ a pravý potomek na pozici $2i + 1$, přičemž znaky ve vstupním řetězci jsou číslovány od 1. V grafu jsou znaky řetězce reprezentovány uzly tak, že jsou číslovány od 0 po jednotlivých úrovních stromu. Mezi odpovídajícími vrcholy jsou dvě hrany (dopředná a zpětná), celkem je tedy v grafu $2n - 2$ hran (procesorů). Hrany se číslují vzestupně od nuly zleva doprava po úrovních stromu počínaje kořenem. Nejdříve jsou očíslovány hrany dopředné a následně zpětné. Názorná ukázka číslování vrcholů a hran je uvedena na obrázku 1.



Obrázek 1: Znázornění způsobu číslování vrcholů a hran

Poté je vypočítána následující hrana v Eulerově cestě. K jejímu výpočtu se používá číslo aktuální hrany ($myId$), počet vrcholů n ($numOfNodes$), počet procesorů (hran) $2n - 2$ ($numOfProcs$) a vztahů, které platí v binárním stromě. Jelikož poslední hrana nemá následníka, je její následník označen číslem -1.

Výpočet váhy aktuální hrany lze díky zvolenému číslování provést tak, že dopředné hrany jsou hrany s číslem od 0 do $n - 2$ včetně a zpětné hrany jsou hrany s číslem od $n - 1$ do $2n - 3$ včetně.

Hlavní částí algoritmu je výpočet součtu suffixů. Cyklus, který musí každá hrana (každý procesor) provést $\lceil \log_2(2n - 2) \rceil$ krát, lze rozdělit na dvě části. První část slouží pro stanovení, které dvojice hran spolu budou komunikovat. Konkrétně, hrana e posílá následující hraně příznak *stopReceiving*. Pokud má hrana nastaven příznak na 1, tak v dalších iteracích tento příznak už nepřijímá. Jinak hrana e přijímá příznak od svého předchůdce. V první iteraci má tento příznak nastavena pouze hrana číslo 0 (kořen). Ve druhé části probíhá samotná výměna zpráv, kdy hrana e posílá předchozí hraně svoji váhu $weight(e)$ a číslo následující hrany. Po dokončení obou částí hrana e aktualizuje hodnotu příznaku *stopReceiving*, číslo následující hrany a mezivýsledek součtu suffixů. Mezi první a druhou částí iterace a po aktualizaci hodnot po druhé části iterace byla ze synchronizačních důvodů použita bariéra. Při implementaci bylo také nutné vyřešit problém spočívající ve vícenásobném dotazování poslední hrany (posledního procesoru). Řešením je neodesílání příznaku *stopReceiving* v případě, že je následníkem hrany poslední hrana. Poslední hrana (procesor) tedy vůbec nekomunikuje.

Následně procesory 0 až $n-2$ včetně, které reprezentují dopředné hrany, vypočítají svou pozici v průchodu preorder. K tomu využijí vztah: $preorderPosition(v) = n - weight(e) + 1$, kde n je počet vrcholů binárního stromu a $weight(e)$ je výsledná hodnota součtu suffixů hrany e, která směřuje do uzlu v.

Nakonec procesory 0 až $n-2$ včetně odesílají vypočítané pozice hlavnímu procesoru $n-1$, který představuje první zpětnou hranu. Ten po přijetí všech hodnot vypíše výsledné pořadí preorder vrcholů.

Komunikační protokol

Komunikační protokol zachycuje sekvenční diagram 3 (viz příloha). Uvažujme délku vstupního řetězce $n \geq 3$. V první iteraci procesory P_0 až P_{2n-5} posílají příznak *stopReceiving* (v sekvenčním

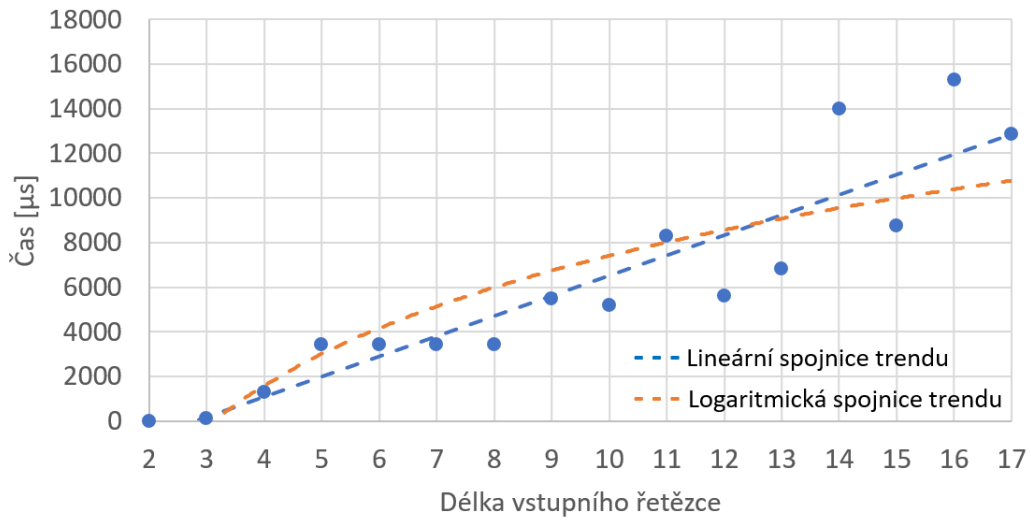
diagramu *Request*) svému následníkovi. Poté procesory P_{2n-4} až P_1 posílají svoji váhu a následníka (v sekvenčním diagramu *Data(next, weight)*). Po aktualizaci hodnot, tedy nastavení příznaku *stopReceiving*, mezivýsledku součtu suffixů a následníka, probíhá další iterace, v níž komunikují procesory ob jedno. Ze sekvenčního diagramu lze vyčíst, že se počet komunikujících procesorů postupně snižuje. Po dokončení algoritmu pro výpočet součtu suffixů odesílají procesory P_0 až P_{n-2} svou pozici v preorder průchodu procesoru P_{n-1}.

Pokud je délka vstupního řetězce $n < 3$, tak je výsledkem výpočtu součtu suffixů váha procesoru (hrany).

Experimenty

Pro experimentální ověření časové složitosti algoritmu byla vytvořena sada se šestnácti testovacími vstupními řetězci. Délky vstupních řetězců (odpovídají počtu uzlů v binárním stromu) byly zvoleny v intervalu od 2 do 17. Pro každou délku vstupního řetězce byl program spuštěn celkem desetkrát a poté byl vypočítán průměrný čas. Časová náročnost výpočtu byla měřena pomocí knihovny `chrono`. Zdrojový kód byl pro tyto účely modifikován, na dvě místa byla vložena časová razítka. Analyzovaný úsek zdrojového kódu zahrnuje kroky algoritmu 1–4 (viz kapitola Rozbor a analýza algoritmu). Celková doba výpočtu byla získána odečtením hodnot časových razítek od sebe.

Graf na obrázku 2 znázorňuje závislost doby běhu algoritmu na délce vstupního řetězce. V sekci Rozbor a analýza algoritmu jsme odvodili časovou složitost $t(n) = \mathcal{O}(\log(n))$. Pro zvolený interval délek vstupních řetězců se časová složitost implementace algoritmu pohybuje mezi logaritmickou a lineární časovou složitostí, což zcela nedpovídá teoretické časové složitosti algoritmu. Důvodem by mohlo být využití synchronizačního prostředku (bariéry).

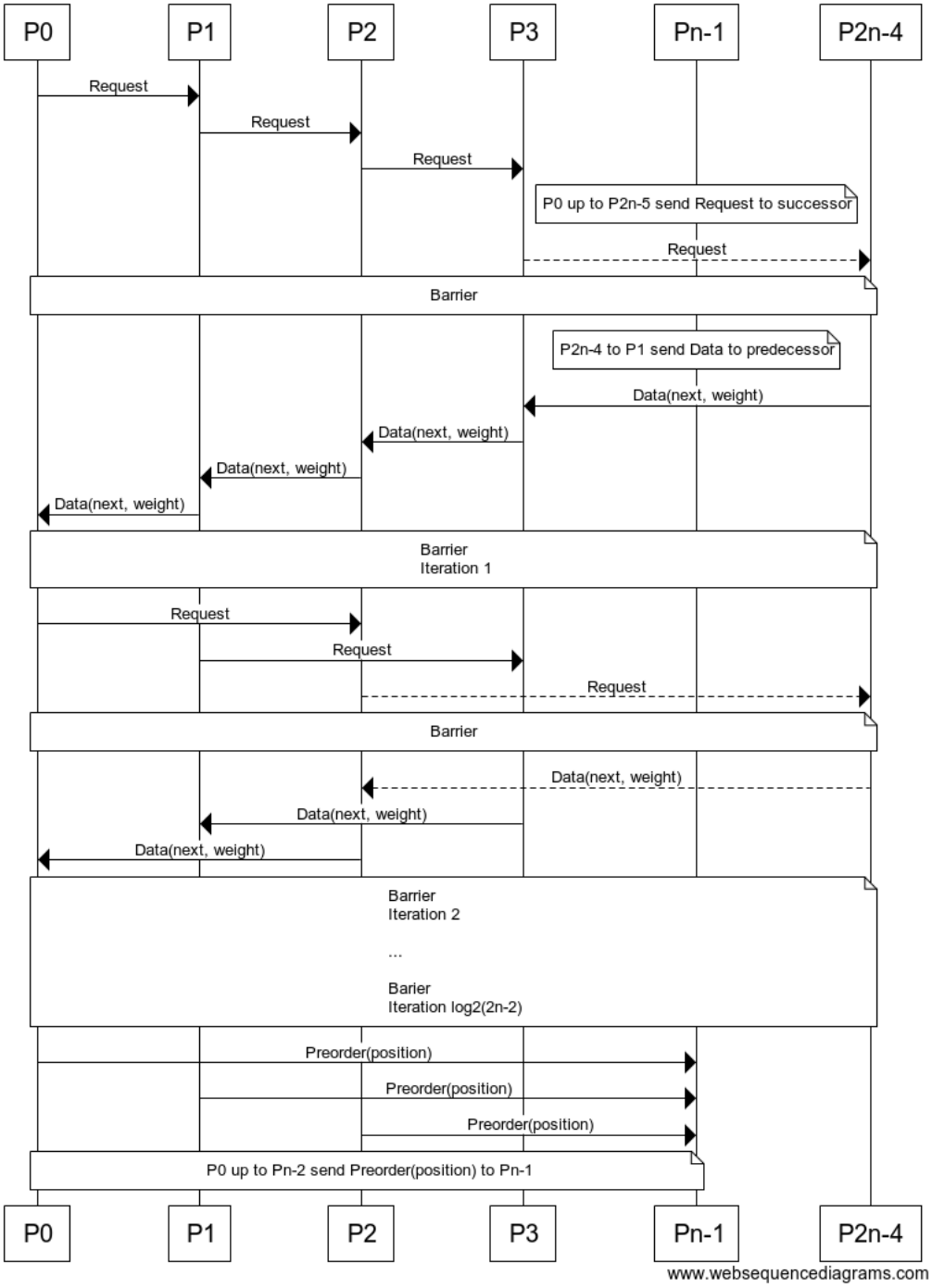


Obrázek 2: Závislost doby běhu algoritmu na délce vstupního řetězce (počtu vrcholů binárního stromu)

Závěr

Cílem projektu bylo implementovat úlohu Přiřazení pořadí preorder vrcholům. Nejdříve byl proveden rozbor a analýza celého algoritmu, včetně odvození jeho teoretické časové a prostorové složitosti. Další část dokumentu se věnovala samotné implementaci a komunikačnímu protokolu, který byl znázorněn pomocí sekvenčního diagramu. V poslední části byly provedeny experimenty, jejichž výsledky se blížily předpokládané časové složitosti implementovaného algoritmu.

Přílohy



Obrázek 3: Grafické znázornění komunikačního protokolu