# PRL 1. projekt – Odd–even merge sort

Tomáš Kantor (xkanto14)

7. dubna 2022

## 1. Rozbor a analýzu algoritmu

Algoritmus Odd–even merge sort je paralelní řadící algoritmus, který je založen na procesorech compare and swap (CS), které mají dva vstupy $x_1$ $x_2$ a dva výstupy $y_1 = min(x_1, x_2)$ a $y_2 = max(x_1, x_2)$. Z těchto procesorů jsou složeny řadící sítě $N \times N$, které mají na vstupu dvě seřazené posloupnosti délky $N$ a na výstupu jednu seřazenou posloupnost délky $2N$. Tyto sítě lze vytvořit ze dvou menších sítí $N/2 \times N/2$ a jedné vrstvy prvků CS.

### 1.1. Odvození časové složitost

$t(2n)$ značí dobu běhu sítě velikosti $n \times n$, která spojí dvě seřazené posloupnosti délky $n$. Platí[1]:

$$t(2) = 1 \text{ pro } n = 1$$

$$t(2n) = t(n) + 1 \text{ pro } n > 1$$

Celkově tedy $t(2n) = log(n) + 1$ je doba potřebná k vytvoření seřazené posloupnosti délky $2n$ ze dvou seřazených posloupností délky $n$. Pro zcela neseřazené pole čísel je potřeba nejprve vytvořit posloupnosti délky 1. Pak posloupnosti délky 2, 4 ... n/2, n. To je potřeba provést zřejmě $log(n)$ krát. Celková časová složitost je tedy $T(n) = log(n) \times (log(n) + 1) = O(log^2(n))$.

### 1.2. Odvození počtu procesorů

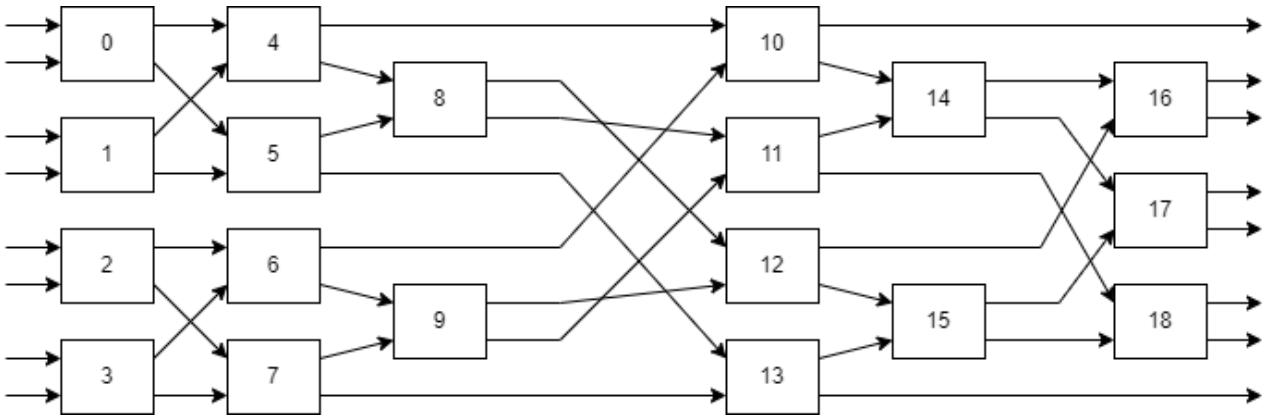
Pokud $P(2n)$ označíme počet procesorů potřebných k vytvoření seřazené posloupnosti délky $2n$ ze dvou posloupností délky $n$, pak platí[1]:

$$p(2) = 1 \text{ pro } n = 1$$

$$p(2n) = 2p(n) + (n - 1) \text{ pro } n > 1$$

Z toho vyplývá, že $p(2n) = 1 + nlog(n)$. Pro seřazení zcela náhodné posloupnosti je potřeba vytvořit posloupnosti délky $1, 2...n/2, n$. To je $log(n)$ takový posloupností. Celkový počet procesorů je $P(n) = log(n) \times (1 + nlog(n)) = nlog^2(n)$.

Celková cena je $C(n) = P(n)T(n) = (nlog^2(n)) \times (log^2(n)) = nlog^4(n)$. Cena algoritmu není optimální.




Obrázek 1: Schéma zapojení jednotlivých procesorů v řadící síti pro 8 vstupů

## 2. Implementace

Schéma 1 bylo implementováno pomocí knihovny Open MPI. Procesor 0 načte data ze souboru, vstupní posloupnost vypíše v zadaném formátu a následně hodnoty rozešle procesorům 0 až 3 pomocí funkce `MPI_Send`. Pomocí parametru `tag` je zvolen správný vstup.

Funkce `void cmp_swap(int in1, int in2, int out1, int out2, int tag1, int tag2)` popisuje jednu komponentu CS. Parametry `in1` a `in2` znamenají číslo komponenty, která je připojena na vstup. Parametry `out1` a `out2` znamenají číslo komponenty, která je připojena na výstup. Parametr `tag1` znamená číslo vstupu komponenty `out1`, na který je výstup připojen. Stejně tak pro paramter `tag1` a číslo komponenty `out1`. Funkce `cmp_swap` volá vždy dvakrát funkci `MPI_Recv` a dvakrá funkci `MPI_Send`. Každý procesor spustí funkci `cmp_swap` jednou, správné parametry jsou voleny uvnitř konstrukce `switch(rank)`.

Procesory spolu komunikují podle schématu na obrázku 1. Procesor 0 přijímá zprávy procesorů 10, 13, 16, 17, 18 a vypíše je na výstup v zadaném formátu.

## 3. Závěr

V rámci projektu byl implementován algoritmus odd–even merge sort. Implementován byl v jazyce C pomocí knihovny Open MPI. Byly provedeno testy ověřující jeho funkčnost a to jak na lokálním stroji, tak i na referenčním serveru Merlin. Z analýzy ceny tohoto algoritmu vyplývá, že není optimální.

## Reference

[1] Akl, S. G. *The design and analysis of parallel algorithms.* [b.m.]: Prentice-Hall International, 1989.