HPP #2 Merge Sort

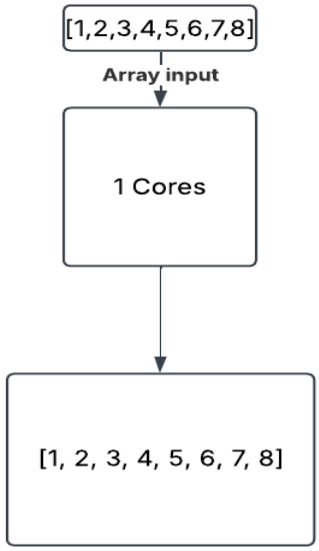
STUDENTNUMMER: 1734098

Joey Welvaadt
HOGESCHOOL UTRECHT | 18-02-25
HTTPS://GITHUB.COM/JOEYWELVAADT1999/HPPV2

1. Cores ontwerpen

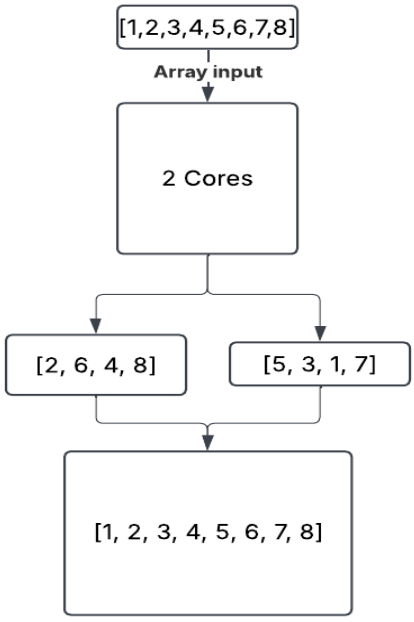
1 Core

Met een enkele thread is het sorteren van de lijst nog steeds sequentieel. Er veranderd nog niets.



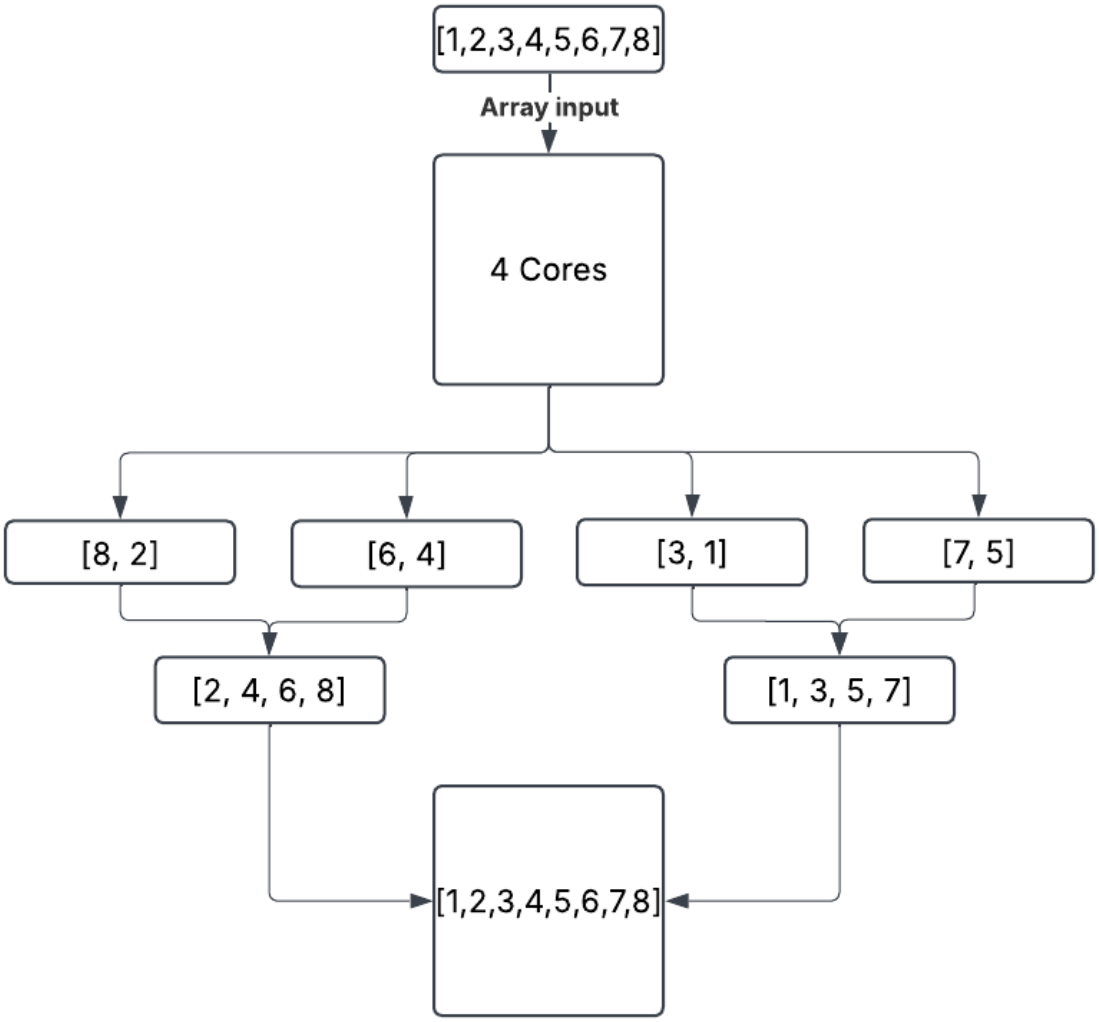
2 Cores

Hier wordt de lijst als eerst opgesplitst in 2 gelijken lijsten van lengte n/2. Vervolgens worden deze lijsten elke op hun eigen thread gesorteerd en vervolgens weer samen gevoegd tot 1 enkele lijst.



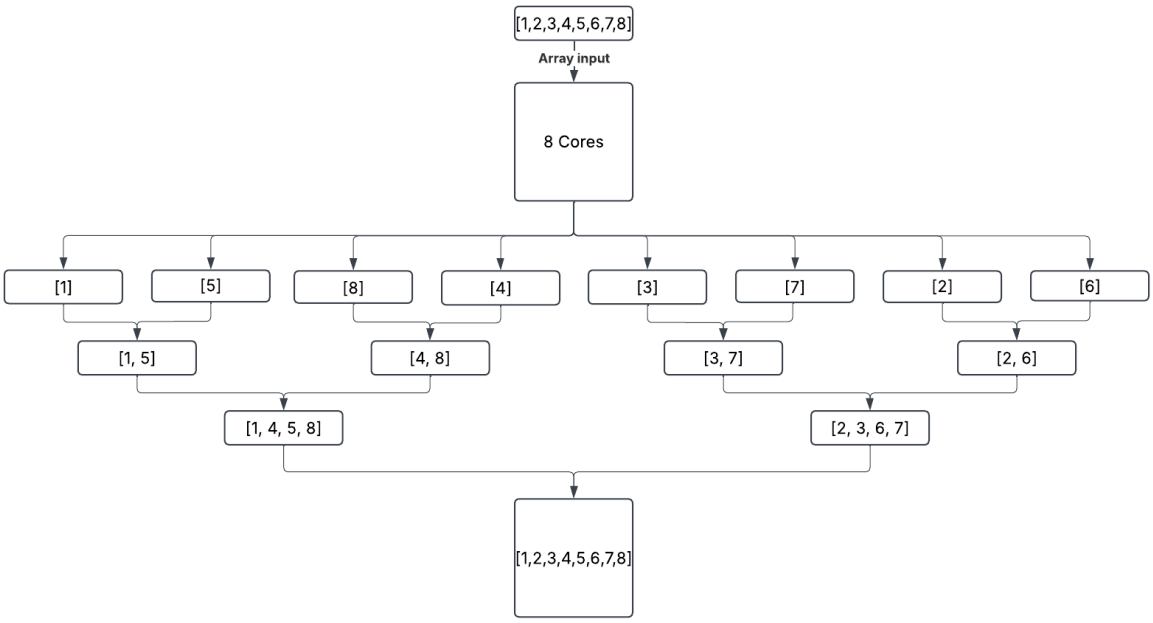
4 Cores

Aangezien we nu 4 verschillende lijsten nodig hebben, delen we de originele lijst op in 4 lijsten van lengte n/4. Vervolgens wordt elk van deze lijsten op hun eigen thread gesorteerd. Na het sorteren worden de 4 lijsten samengevoegd in 2 lijsten. De overige 2 threads stoppen hier en we gaan verder met 2 cores. Vanaf hier is het hetzelfde (zonder de initiele splitsing) als de paragraaf 2 Cores.



8 Cores

Als laatst hebben we de implementatie met 8 cores, hetzelfde als de vorige 2 implementaties beginnen we met het opsplitsen van de lijst in 8 gelijke lijsten met lengte n/8. Vervolgens worden de lijsten op hun eigen thread gesorteerd, samengevoegd in 4 lijsten van n/4 en stoppen de overige threads en gaan door met 4. Vanaf dit punt volgt het de zelfde implementatie als 4 Cores.



2. Analyse

Sequentiële Merge Sort (1 Core)

De sequentiele merge sort is opgedeeld in 2 stukken, het stukken. O(log N) dit komt door de recursieve verdeling van de array in kleinere stukken. Waar O(N) komt door het samenvoegen van de stukken.
Dit samen geeft de notatie O(N log N) als resultaat.

Parallel Merge Sort (2 Cores)

Aangezien er nu 2 verschillende cores aan het werk zijn word de originele O(N log N) opgesplitst. Deze implementatie heeft daarom een notatie van O((N log N) / 2) + O(N). Waar O(N) wordt toegevoegd door het uiteindelijk samenvoegen van de 2 lijsten over de verschillende cores.

Parallel Merge Sort (C Cores)

De Big O notatie voor 2 cores is nu duidelijk. In deze paragraaf wordt er een generale notatie gemaakt voor C aantallen cores. Dit doen we omdat voor 4 en voor 8 cores de notatie O((N log N) / 4) en O((N log N) / 8) respectief wordt. Ook moeten we rekening houden met het aantal keer dat we de lijsten samenvoegen. Dit is gelijk aan C (het aantal cores) delen door 2 (Zie afbeeldingen hoofdstuk 1). Wat een Big O notatie geeft van O(0.5CN).

De Big O notatie voor de algemene ontworpen implementatie wordt: O((N log N) / C) + O(0.5CN)

3. Overhead & Communicatie

1 Core

Voor een implementatie met 1 core is er geen communicatie tijd, er is namelijk geen communicatie met andere cores.

2 Cores

Voor de implementatie met 2 cores is er communicatie tijd aanwezig. Dit bestaat uit een splitsing (1 eenheid) en een merge (1 eenheid). Dit brengt ons tot een totaal van 2 communicatie tijd eenheden.

4 Cores

Voor de implementatie met 4 cores wordt de lijst eerst 3 keer gesplitst. Eerste keer door de helft en vervolgens beide helften weer door 2. Ook moeten de lijsten 3 keer worden samengevoegd, eerst van 4 naar 2 dan van 2 naar 1. Dit geeft ons een resultaat van 6 communicatie tijd eenheden.

8 Cores

Als laatst hebben we de implementatie met 8 cores. Hier wordt de lijst als eerst opgesplitst in 8 gelijke stukken. Hiervoor hebben we 7 splitsingen nodig. 1 lijst door de helft, dan de 2 halve lijsten door de helft, vervolgens de 4 kwart lijsten opnieuw door de helft (1 + 2 + 4 = 7). De lijsten worden daarbij ook weer 7 keer samengevoegd. Dit geeft ons een resultaat van 14 communicatie tijd eenheden.

4. Run-time complexitiy