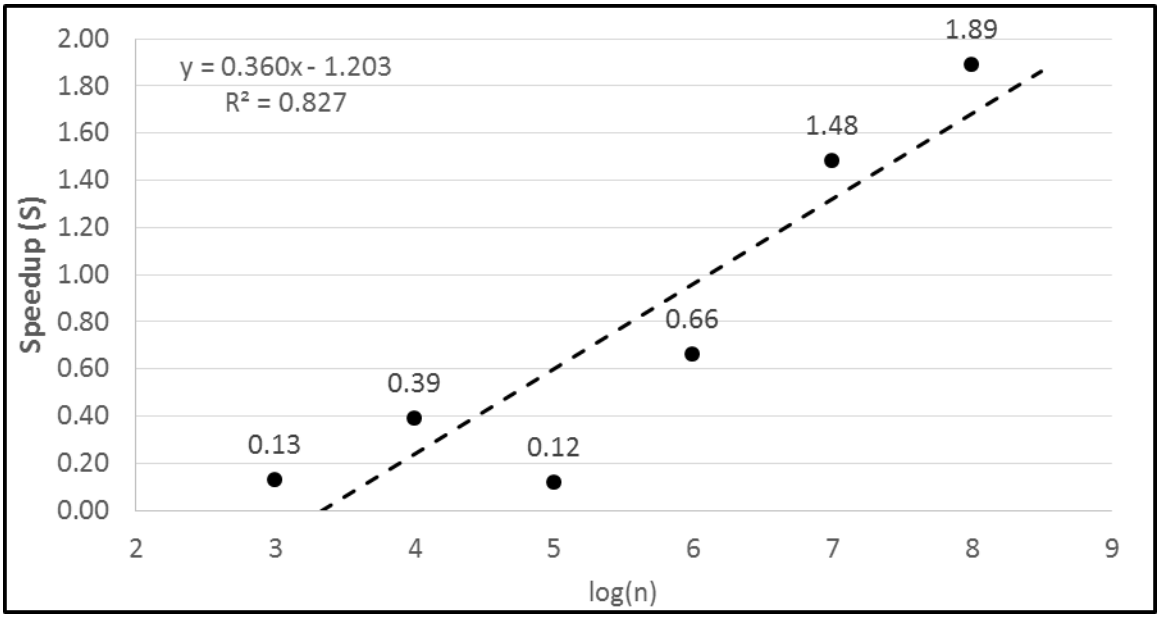
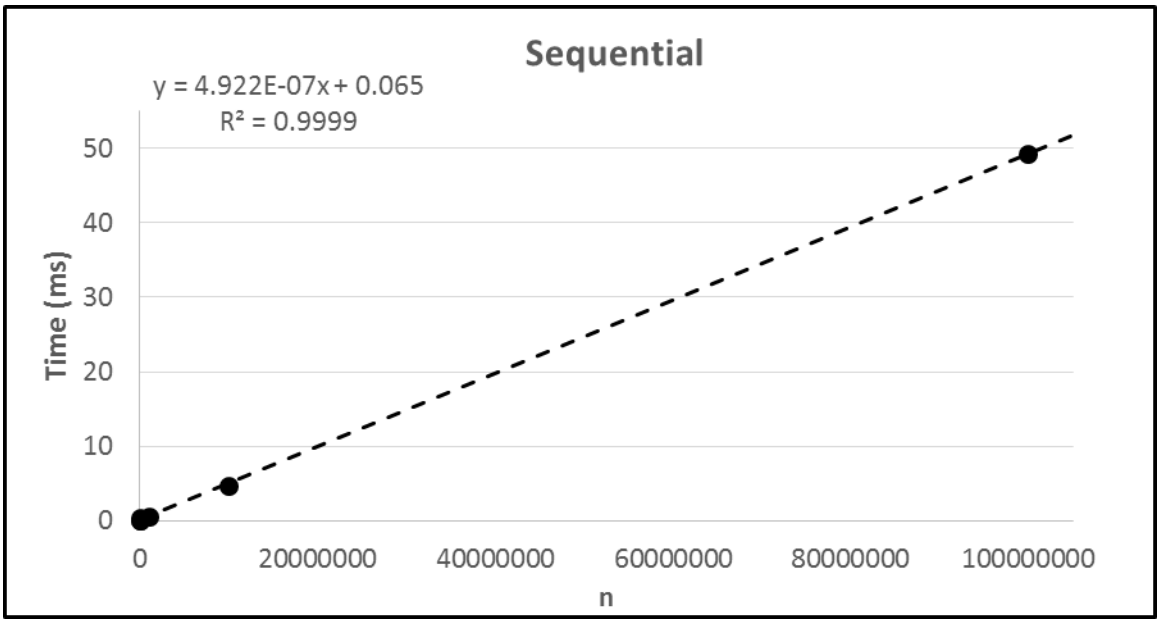
Resultat

CPU: Intel Core i5 5257U (2.70GHz) – 4 kjerner – 4 tråder

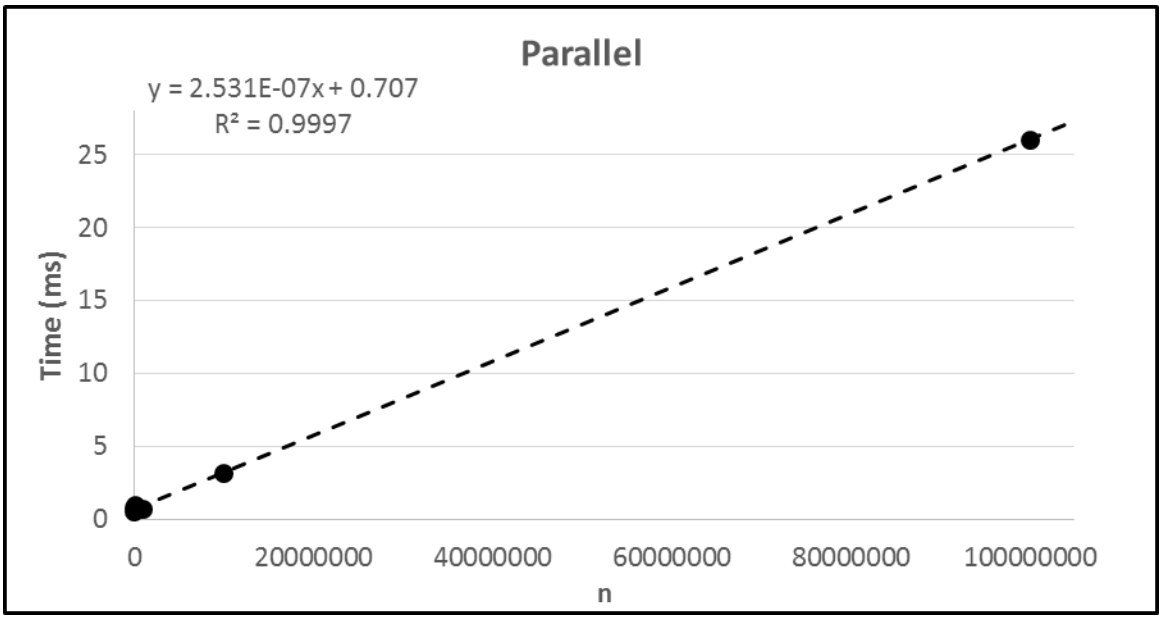
n	Algo	Time (ms) for each trial									Median	Speed -up
		1	2	3	4	5	6	7	8	9		
10^3	Sort()	0.59	0.30	0.44	0.13	0.12	0.11	0.10	0.11	0.11	0.12	0.13
	Seq	0.11	0.04	0.07	0.07	0.07	0.08	0.03	0.03	0.03	0.07	
	Para	3.04	1.04	1.22	0.52	0.58	0.49	0.34	0.41	0.41	0.52	
10^4	Sort()	4.23	0.92	1.49	1.34	1.37	0.96	0.98	0.93	1.28	1.28	0.39
	Seq	0.59	0.38	0.33	0.34	0.23	0.21	0.34	0.11	0.05	0.33	
	Para	2.54	1.18	0.70	0.90	0.80	0.77	0.78	0.86	0.98	0.86	
10^5	Sort()	16.59	13.70	11.67	9.89	8.72	8.06	6.38	6.35	6.38	8.72	0.12
	Seq	2.28	1.04	0.17	0.16	0.12	0.05	0.05	0.05	0.05	0.12	
	Para	5.28	2.01	1.91	1.10	1.00	0.41	0.37	0.37	0.41	1.00	
10^6	Sort()	151.20	75.52	75.70	75.57	75.94	77.73	75.43	76.04	76.00	75.94	0.66
	Seq	5.40	0.81	0.49	0.51	0.50	0.47	0.48	0.47	0.55	0.50	
	Para	9.99	20.01	0.90	0.84	0.75	0.76	0.72	0.74	0.73	0.76	
10^7	Sort()	982.40	884.39	885.58	885.15	885.28	886.21	883.22	883.62	883.14	885.15	1.48
	Seq	11.25	13.34	4.75	4.66	5.07	4.63	4.88	4.68	4.74	4.75	
	Para	17.13	30.45	3.27	3.01	3.10	3.10	3.21	3.20	3.05	3.20	
10^8	Sort()	10249.55	10151.21	10205.91	10155.87	10157.40	10148.17	10213.65	10155.33	10152.00	10155.87	1.89
	Seq	66.42	85.38	49.31	49.53	48.92	48.84	49.55	49.06	49.25	49.31	
	Para	98.11	109.09	96.46	25.93	25.76	26.02	26.15	25.84	25.93	26.02	



***Graf* 1**: Speedup vs. log(n), med trendline: $Speedup(S) = 0.360\log(n) - 1.203$



***Graf* 2**: Median tid vs. n for sekvensielle løsningen, med trendline: $Tid = (4.922 \times 10^{-7})n + 0.065$



***Graf* 3**: Median tid vs. n for parallell løsningen, med trendline: $Tid = (2.531 \times 10^{-7})n + 0.707$

Analyse

Med en ganske høy verdi for R^2 $(= 0.827)$ i Graf 1 som viser speedup som en funksjon av n, kan vi si at sammenhengen mellom de to variabler er logaritmisk. Fra trendline likningen i Graf 1, er det også lett å se når speedup er større enn 1:

$$1 = 0.0360 \log(n) - 1.203 \Leftrightarrow n \approx 1.3 \times 10^6$$

➔ Speedup > 1 når 'n' er i størrelsesorden av 10^6

Hvis vi tar en nærmare titt på alle datapunktene i Graf 1, så kan vi se at speedup ikke alltid øker når n går større. Når n øker fra 10^4 til 10^5, reduserer speedup fra 0.39 til 0.12, og dette ikke er noe jeg forventet. Først trodde jeg at dette er bare på grunn av noen tilfeldige feil, men når jeg kjørte programmet igjen og igjen, så fikk jeg samme resultat. Da tenkte jeg at konstanter i $\mathcal{O}$-notasjon kanskje har noe å gjøre med det.

Når vi bruker sekvensiell algoritme for å løse oppgaven, må vi gå gjennom alle elementer i data-arrayen. Det betyr at sekvensielle løsningen har kompleksiteten $\mathcal{O}(n)$ plus med en ekstra konstant. Denne konstanten kommer fra den delen av algoritmen som alltid tar samme tid for å kjøre, uansett hvor stor n er. For eksempel, for hva som helst input-størrelse, behøver vi alltid instikk-sortere de første 40 tallene; og dette tar en konstant tid hver gang vi kjører programmet. På samme måte, er komplekstiteten til parallell løsningen også $\mathcal{O}(n)$ plus med en ekstra konstant. Dette er fordi parrallell løsningen kjører samme algoritme som sekvensiell løsningen, men bare med mange tråder. Derfor er det ikke så overraskende å se at det finnes en lineær sammenheng mellom tid og n, som vises i Graf 2 og 3.

- $Sekvensiell\ tid = (4.922 \times 10^{-7})n + 0.065$
- $Parallel\ tid = (2.531 \times 10^{-7})n + 0.707$

$$\Leftrightarrow Speedup = S(n) = \frac{sekvensiell\ tid}{parallel\ tid} = \frac{(4.922 \times 10^{-7})n + 0.065}{(2.531 \times 10^{-7})n + 0.707}$$

For verdier av n som er mindre enn 10^4, så behøver vi ikke bry oss om n-termer i både numeratoren og denominatoren, fordi de er mange ganger mindre enn konstantene 0.707 og 0.065. Men når $n = 10^5$, så er n-termer på samme størrelseorden som konstantene i likningen. Dette forklarer hvorfor speedup går ned fra 0.39 til 0.12 når n øker fra 10^4 til 10^5. Det er liksom en "overgangstilstand" (transition state) når n er mellom 10^4 og 10^5.

For verdier av n som ligger mellom 10^5 og 10^8, så er forholdet mellom speedup og n logaritmisk slik Graf 1 viser. Vi kan også analysere hva speedup er når n går mot uendelig:

$$\lim_{n\to\infty} S(n) = \lim_{n\to\infty} \frac{(4.922 \times 10^{-7})n + 0.065}{(2.531 \times 10^{-7})n + 0.707} = \frac{4.922}{2.531} = 1.94$$

Dette betyr at det finnes en grenseverdien for speedup, uansett hvor stor n er. Jeg vet ikke om det er sant; det er bare min hypotese basert på resultatene jeg fikk. Men for å kunne teste denne hypotesen, trenger jeg en mye kraftigere datamaskin enn den som jeg har nå, slik at jeg kan kjøre programmet når n er mange ganger større enn 100 mill.

En annen ting som jeg merker er: når $n = 10^3$, så er `Array.sort()` algoritmen 1.71 ($= 0.12/0.07$) ganger tregere enn sekvensiell løsningen, og 4.33 ($= 0.52/0.12$) ganger raskere enn parallell løsningen. Men når $n = 10^8$ så er algoritmen 206 og 390 ganger tregere enn sekvensiell og parallell løsningen, henholdsvis. Det betyr at `Array.sort()` blir veldig ineffektivt når n blir stor. For å kunne forstå hvorfor, kan vi se på Javadoc for Arrays klassen[1], hvor det står at algoritmen som de bruker har kompleksiteten $\mathcal{O}(n \log n)$; mens for både sekvensiell og parallell løsningen, er kompleksitet bare $\mathcal{O}(n)$.

[1] https://docs.oracle.com/javase/7/docs/api/java/util/Arrays.html#sort%28int[]%29