IN3030, Oblig1

Jeg har en Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz 2.59 GHz..

Jeg ser at A3 er tregere for små N, mest sannsynlig fordi synkronisering og oppretting av trådene er mer krevende enn å bare løse oppgaven sekvensielt. Men jeg ser at for store veldig store N, så er speedupen rundt 4x, som samsvarer med at jeg har 4 kjerner på min maskin. Ved veldig store N som jeg testet (ikke i oppgaven) så jeg nærmere 8x speedup, noe som kanskje har noe å gjøre med at jeg har 8 fysiske tråder på min CPU. A3 sliter litt mer når K blir stor også, da trående må vente på hverandre i kø for å ikke gå i beina på hverandre i k-1 til 0. Når K er liten og N veldig stor, så er A3 ekstremt mye raskere enn A2, da hver tråd kan lete gjennom deler av array på likt. Men ved små N så er A2 ekstremt mye mer nyttig, da det er så dyrt å starte tråder og synkronisere at A2 allerede er ferdig før A3 starter.

Jeg har ikke helt skjønt hvordan jeg skal lage grafene. Men her legger jeg ved speedupene jeg opplevde ved ulike verdier av N og K. Her ser vi at jo større N blir, jo bedre gjør A3 det.

K = 20.

N	Speedup
1000	0.02
10000	0.07
100000	0.08
1000000	0.18
10000000	1.7
100000000	3.7
1000000000	4.2

K = 100.

N	Speedup
1000	0.03
10000	0.01
100000	0.22
1000000	0.32
10000000	0.92
100000000	3.7
1000000000	5.3

	Array.sort	A2	A3
N=1000/K=20	3ms	1ms	37ms
N=10000/K=20	9ms	2ms	27ms
N=100000/K=20	201ms	3ms	34ms
N=1000000/K=20	555ms	6ms	32ms
N=10000000/K=20	7067ms	58ms	34ms
N=100000000/K=20	75017ms	614ms	163ms
N=1000000000/K=20	867056ms	6188ms	1452ms

	Array.sort	A2	A3
N=1000/K=100	2ms	1ms	31ms
N=10000/K=100	7ms	1ms	61ms
N=100000/K=100	240ms	6ms	27ms
N=1000000/K=100	588ms	9ms	28ms
N=10000000/K=100	6680ms	82ms	89ms
N=100000000/K=100	76505ms	766ms	208ms
N=1000000000/K=100	852560ms	7779ms	1443ms