Øving 2 – Quicksort

Ingebrigt Hovind

Algoritmer og datastrukturer

Disse målingene ble tatt ved å generere en tilfeldig tabell med den gitte lengden som inneholdt tall mellom 0 og 100 000. Det ble generert to forskjellige slike tabeller, én for sorteringen med én pivot og én for sorteringen med to pivots. Målingene her representerer tiden brukt på én sortering, da dette tok lang nok tid til at maskinklokken ble nøyaktig i forhold.

Det ble forsøkt å optimalisere ved hjelp av å bruke insertion sort med binary search i tilstrekkelig korte tabeller. I realiteten så tok sorteringen like lang tid selv med insertion sort, noe som jeg går ut ifra at er fra en svakhet i koden. Jeg valgte derfor å gå tilbake slik programmet var originalt, med bruk av median3sort istedenfor insertion sort.

Dette er dataene med ren quicksort, kun optimalisert ved å bruke taggen -Ofast i kompileringen:

10 millioner tall

	Tilfeldige tall	Annenhvert tall er likt	Sortert tabell
Med én pivot	824 ms	558 ms	227 ms
Med to pivots	818 ms	561 ms	216 ms

100 millioner tall

	Tilfeldige tall	Annethvert tall er likt	Sortert tabell
Med én pivot	8361 ms	5833 ms	2143 ms
Med to pivotrs	8227 ms	5761 ms	2040 ms

Man kan se her at det er litt raskere å bruke dual-pivot på alle tre kategoriene, men resultatene er så nære at tabellen med 10 millioner tall der annethvert tall er likt så brukte dual pivot lenger tid.

Når tidsforskjellene er så nære hverandre, så blir det vanskelig å konkludere med hvilken sortering som er raskest, da små forskjellig i ledig prosessorkraft kan ha vært forskjellen, spesielt i de mindre arrayene. Likevel så vil jeg si at differansen i arrayene med 100 millioner tall er store nok til å konkludere med at dual pivot er raskere.

Alle sorteringene besto både checksum-testen og verifisert til å være i stigende rekkefølge.