Ola Kristoffer Hoff

Algoritmer og datastrukturer – IDATT2101
Øving 2 – Sortering

Jeg har valgt å gjøre alternativ 1, sammenligne ulike typer quicksort. Jeg har samarbeidet med Oline Amundsen. Jeg implementerte quicksort metoden (med et delingstall) fra boka, og de andre nødvendige metodene som trengtes for at den skulle fungere. Så implementerte jeg quicksort metoden (med to delleingstall) fra anbefalt nettside (www.geeksforgeeks.org/dual-pivot-quicksort/) og endret på det som var anbefalt å gjøre (i starten av «partition()»-metoden), ellers så er den slik jeg hentet den fra nettsiden.

Deretter laget jeg en int[]-genererene metode (genererListe()), som kan gi meg en liste med tilfeldige verdier, med mange duplikater, og en sortert liste. Så laget jeg to test metoder for å se at sorteringen ble riktig, «testRekkefolge()» og «hentSum()».

Deretter la jeg inn tidtakingen for sortering av de to metodene.

Test resultater:

For notat: Jeg kjørte alle testene 10 ganger for å sikre at svaret var mer deterministisk og ikke et enkelt tilfelle. Verdiene i tabellene under er gjennomsnittet av de 10 gjennomgangene (tidtakingen), eller totalt resultat. Alle tabellene har også 100.000.000 elementer. Tiden ble tatt i nanosekunder, og deretter heltalsdividert til millisekunder. Metodene er også kjørt med hver sin dyp-kopi av samme liste, som er forskjellig fra hver gang (generes på nytt). Alle tidtakingene ligger som et vedlegg i bunn av rapporten.

Tabell over helt tilfeldige data

Type	Quicksort1	Quicksort2
Sum test	10/10	10/10
Rekkefølge test	10/10	10/10
Tid (gjennomsnitt)	12780ms	11677ms
Raskest	1/10	9/10

Vi kan konkludere at quicksort med to delingstall gjennomsnittlig er raskere på usorterte tilfeldige lister, da den gjennomsnittlig var 1103ms raskere som er i overkant av et helt sekund.

Tabell med mange duplikater (annenhver)

Type	Quicksort1	Quicksort2
Sum test	10/10	10/10
Rekkefølge test	10/10	10/10
Tid (gjennomsnitt)	2867ms	3331ms
Raskest	10/10	0/10

Her er det quicksort med et delingstall som gjennomsnittlig er raskest, gjennomsnittlig med nesten et halvt sekund. Her kjørte jeg også noen ekstra tester ved å øke til 3 og 4 forskjellige tall, altså at hvert 3. og 4. element var likt og allerede på at hvert 4. element var likt så ble quicksort med to delingstall raskere igjen, dette på 5. også, men på 6. ble den første raskest, så på 7. var den andre raskest igjen og det fortsatte å bytte oppover (testet til og med 10).

Kan hende resultatene hadde blitt annerledes hvis de ikke var «delevis» sortert, det vil si at metoden min lager de i et mønster f. eks. {0, 1, 0, 1, 0, 1,...} eller {0, 1, 2, 3, 0, 1, 2, 3,...}. Men å undersøke dette mye nærmere i detalj følte jeg lå utenfor øvingen og gjorde derfor ikke dette, oppgaven så jo «f.eks. hvor annenhvert element er likt.».

Tabell med mange duplikater (gjennomsnittlig 10 eksemplarer av hvert element)

Type	Quicksort1	Quicksort2
Sum test	10/10	10/10
Rekkefølge test	10/10	10/10
Tid (gjennomsnitt)	12406ms	10387ms
Raskest	0/10	10/10

Her er et interessant resultat, for ved at det hvert element i listen hadde gjennomsnittlig 10 duplikater så var quicksort med to delingstall gjennomsnittlig hele to sekunder raskere. Har ikke noe god teori på hvorfor dette er tilfelle, men tydelig at quicksort med to delingstall dro bedre fordel av duplikatene enn det den andre gjorde her.

Tabell som er sortert fra før

Type	Quicksort1	Quicksort2
Sum test	10/10	10/10
Rekkefølge test	10/10	10/10
Tid (gjennomsnitt)	2327ms	3175ms
Raskest	10/10	0/10

Her ser vi at quicksort med et delingstall er rundt 0.8 sekunder raskere gjennomsnittlig, det koster her da mer å ha flere punkter å dele på en det lønner seg, noe som egentlig ikke er en overraskelse med tanke på at alt er sortert fra før av så man trenger bare «sjekke» at alt er på stell, noe som går fortere med quicksort med et delingstall.

Alle tidtakingene respektivt fra venstre til høyere (Tilfeldig, duplikat annenhver, duplikat 10, sortert) sort1: quicksort med et delingatall, sort2: quicksort med to dleingstall.

	sort1	sort2		sort1	sort2		sort1	sort2		sort1	sort2
1	12887	11553		2827	3231		12111	10133		2319	3242
2	11895	11546		2806	3256		12428	10330		2304	3156
3	13216	11447		2860	3349		12415	10377		2344	3194
4	12614	11774		2770	3306		12476	10396		2351	3132
5	13161	12346		2880	3455		12422	10422		2310	3166
6	13352	11303		2907	3307		12379	10515		2324	3177
7	12997	11263		2913	3179		12416	10338		2321	3179
8	12678	11124		2904	3411		12585	10490		2325	3175
9	13253	12056		2904	3447		12372	10445		2353	3170
10	11751	12353		2898	3364		12453	10423		2314	3163
Gjennomsnitt	12780,4	11676,5		2866,9	3330,5		12405,7	10386,9		2326,5	3175,4