

HEIMADÆMI 5

TÖL203G Tölvunarfræði 2

Kári Hlynsson¹

Háskóli Íslands

21. febrúar 2023

Verkefni 1

Skoðið Java kóðann fyrir Mergesort (ALGORITHM 2.4 í bókinni, glæra 8 í fyrirlestri 9). Segjum að við köllum aðeins á merge-fallið (neðsta línan í sort-fallinu) ef $a[mid+1]$ er minna en $a[mid]$.

- (a) Hvers vegna er í lagi að gera þetta?
- (b) Á hvernig inntaki myndum við græða mest á að bæta þessu inn?

Lausn

Hluti (a)

Hlutverk merge fallsins er að raða hlutfylkjum sem hefur þegar verið raðað endurkvæmt. Segjum sem svo að $a[0] < \dots < a[mid]$ og $a[mid+1] < \dots < a[n]$. Ef $a[mid] < a[mid+1]$.

Hluti (b)

Við græðum mest á þessari útfærslu ef sérhvert stak í fylkinu er í réttu hlutfylki. Til að sjá þetta betur skulum við taka dæmi. Látum $a[] = \{0, 4, 3, 2, 1, 8, 9, 6, 5, 7\}$. Þegar við köllum á sort byrjum við á því að sortera vinstri og hægri hlutann og fáum hlutfylkin $a[0..4] = \{0, 1, 2, 3, 4\}$ og $a[5..9] = \{5, 6, 7, 8, 9\}$. Nú er $a[mid] < a[mid+1]$ og við köllum ekki á merge en við sjáum jafnframt að fylkið er raðað svo það er óþarft.

¹Slóð á Github kóða: <https://github.com/lvthnn/TOL203G/tree/master/HD5>

Verkefni 2

Leysið dæmi 2.3.18 á bls. 305 í kennslubókinni. Kallið ykkur útgáfu QuickX og berið hana saman við upphaflegu útgáfuna í bókinni (Quick.java) með forritinu SortCompare.java. Þið getið hent út úr SortCompare notkun á öðrum röðunar-aðferðum. Skilið breyttu útgáfunni af partition-fallinu og niðurstöðu úr samanburði á Quick og QuickX í SortCompare. Til að fá raunhæfan samanburð notið $n = 1000000$ og $trials = 10$. Þá ættuð þið líka að breyta útprentunarskipuninni í SortCompare, þannig að þið fáið fleiri aukastafi í hlutfallinu milli tímana. Það ætti ekki að vera mjög mikill munur á þessum tveimur útgáfum. Hvers vegna?

Lausn

Útfærslan er til sýnis fyrir neðan í FORRITI 1. median er fall sem var útfært til að skila miðgildi þriggja staka. Úr því að $a[]$ er stokkað í public útfærslunni af sort notum við fyrsta stakið í hlutfylkinu, miðjustak og síðasta stak.

```
18 private static int partition(Comparable[] a, int lo, int hi) {
19     int i = lo, j = hi+1;
20
21     int m = median(a, lo, (lo+hi)/2, hi);
22     Comparable v = a[m];
23     exch(a, lo, m);
24
25     while (true) {
26         while (less(a[++i], v)) if (i == hi) break;
27         while (less(v, a[--j])) if (j == lo) break;
28         if (i >= j) break;
29         exch(a, i, j);
30     }
```

FORRIT 1: Útfærslan á partition í QuickX klasanum

Munur í tímaflækju á Quick og QuickX er nánast enginn. 10 mælingum var safnað úr SortCompare á hlutfallinu t_Q/t_X þar sem t_Q er keyrslutími Quick og t_X er keyrslutími QuickX. Meðaltal var 1.0036 (staðalfrávik 0.066) svo keyrslutími er nokkurn veginn sá sami.

Munurinn er lítill því $a[]$ hefur þegar verið slembistokkað, svo það er alveg tilviljanakennt hvort við drögum úrtak úr fylkinu sem inniheldur miðjaða stærð, en hraði reikniritisins eykst auðvitað ef skiptistökur eru miðjuð.

Verkefni 3

Fylkinu $[2, 3, 1, 4, 5, 7, 6, 8]$ hefur verið skipt upp (*partitioned*) um vendistak (og það sett á réttan stað), en það er ekki gefið upp hvert vendistakið var. Hver af stökunum gætu hafa verið vendistakið? Teljið upp öll möguleg stök og rökstyðjið að þau séu möguleg og hin séu það ekki.