HEIMADÆMI 5

TÖL203G Tölvunarfræði 2

Kári Hlynsson[1]

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 $\texttt{a[0]} < \cdots < \texttt{a[mid]}$ og $\texttt{a[mid+1]} < \cdots < \texttt{a[n]}$. Ef $\texttt{a[mid]} < \texttt{a[mid+1]}$.

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.

[1] Slóð 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 röðunaraðferðum. 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 tímanna. Það ætti ekki að vera mjög mikill munur á þessum tveimur útgáfum. Hvers vegna?