

Heimadæmi 5 - Tölvunarfræði 2

Magnús Daníel Einarsson

February 2023

Verkefni 1

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

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

Lausn:

- Þetta er í lagi því mergesort tekur sér aðeins stað ef stökin eru ekki röðuð.
- Þegar fylkið er núþegar raðað því þá förum við aldrei í merge fallið því $mid+1$ er stærri tala en mid .

Verkefni 2

[Quicksort] Leysið dæmi 2.3.18 á bls. 305 í kennslubókinni. Kallið ykkar 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 er það?

Lausn:

```
public static void findmid(Comparable[] a, int lo, int hi, int mid){
    if (!less(lo, mid)){
        exch(a, mid, lo);
    }
    if (!less(mid, hi)){
        exch(a, hi, mid);
        if (!less(lo, mid)){
            exch(a, mid, lo);
        }
    }
}

// partition the subarray a[lo..hi] so that a[lo..j-1] <= a[j] <= a[j+1..hi]
// and return the index j.
```

```

private static int partition(Comparable[] a, int lo, int hi) {
    int i = lo;
    int j = hi + 1;
    int k = lo + (hi-lo)/2;
    findmid(a, lo, k, hi);
    exch(a, lo, k);
    Comparable v = a[lo];

    while (true) {

        // find item on lo to swap
        while (less(a[++i], v)) {
            if (i == hi) break;
        }

        // find item on hi to swap
        while (less(v, a[--j])) {
            if (j == lo) break;          // redundant since a[lo] acts as sentinel
        }


        // check if pointers cross
        if (i >= j) break;

        exch(a, i, j);
    }

    // put partitioning item v at a[j]
    exch(a, lo, j);

    // now, a[lo .. j-1] <= a[j] <= a[j+1 .. hi]
    return j;
}

```

 magnusde95@tg-dw012 / University-Assignments
 For 1000000 random Doubles
 Quick is 0.81871 times faster than QuickX

Ástæðan fyrir því að Quick er hraðar en QuickX er að sú að það kostar of mikinn tíma að nota þessa auka aðferð til að finna alltaf miðju stakið í minni tölvu.

Verkefni 3

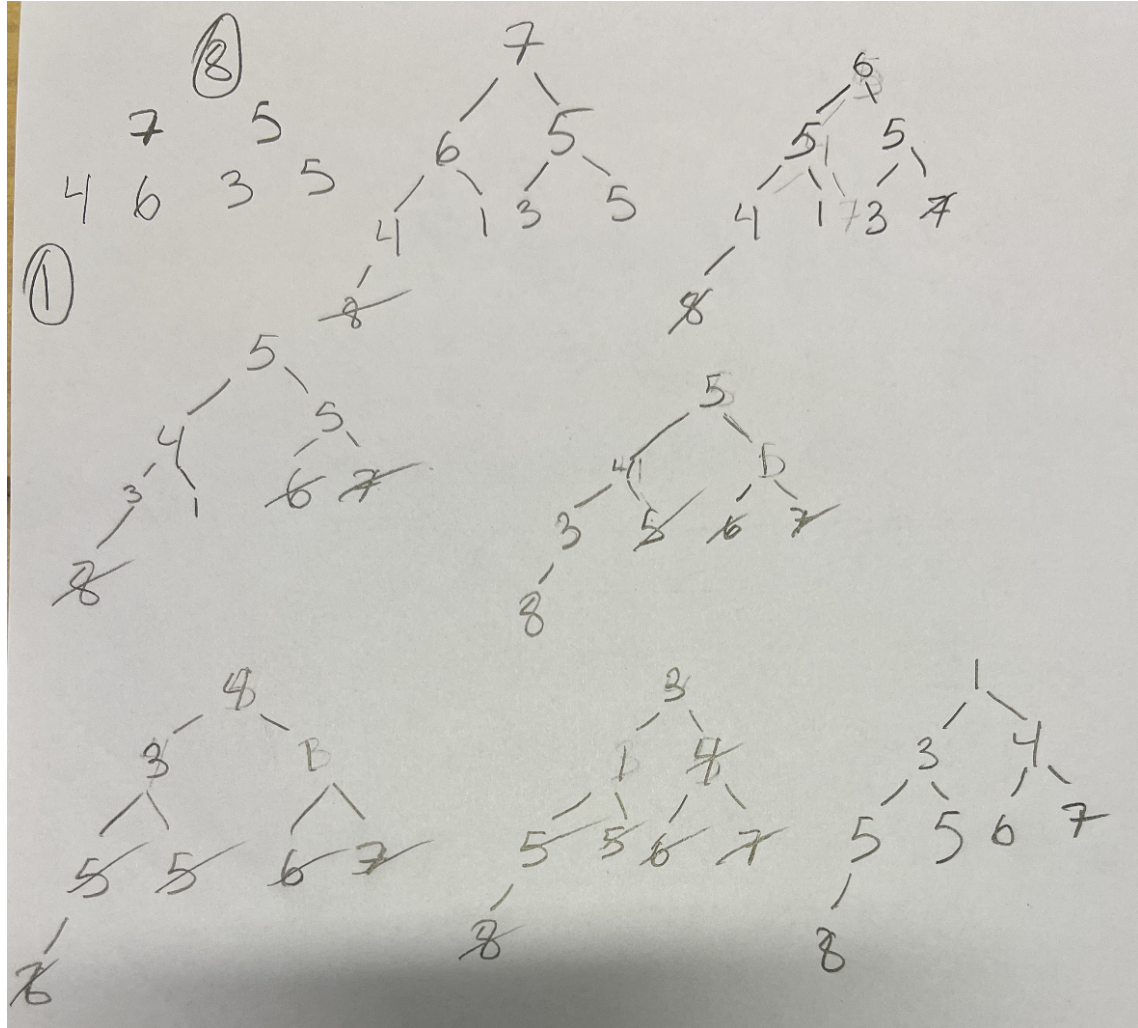
[Quicksort] Fylkinu [2, 3, 1, 4, 5, 7, 6, 8] hefur verið skipt (partitioned) um vendistak (og það sett á réttan stað), en það er ekki gefið upp hvert vendistakið var. Hver af stökunum gætu hafa verið vendistakið? Teljið upp öll möguleg stök og rökstyðjið að þau séu möguleg og hin séu það ekki.

Lausn: Vendistak er bara rétt ef það endar á réttum stað í fylkinu eftir að það hefur verið notað. Einu stökin sem eru á réttum stað eru 4,5 og 8. Þess vegna eru þetta einu stökin sæm gætu verið vendistök.

Verkefni 4

[Hrúgur] Gefið er fylkið [5, 4, 8, 1, 6, 3, 5, 7]. Sýnið hvernig það raðast með hrúguröðun (heap sort) með því að búa til mynd sem er sambærileg við myndina á bls. 324 í kennslubókinni (eða á glæru 35 í fyrirlestri 11). Sýnið líka hrúguna sem tvíundartré eftir að fylkinu hefur verið hrúguraðað (þrep 1).

Lausn:



		5	4	8	1	6	3	5	7
8	4	5	4	8	7	6	3	5	1
8	2	5	7	8	4	6	3	5	1
8	1	8	7	5	4	6	3	5	1
7	1	7	6	5	4	1	3	5	8
6	1	6	5	5	4	1	3	7	8
5	1	5	4	5	3	1	6	7	8
4	1	5	4	1	3	5	6	7	8
3	1	4	3	1	5	5	6	7	8
2	1	3	1	4	5	5	6	7	8
1	1	1	3	4	5	5	6	7	8