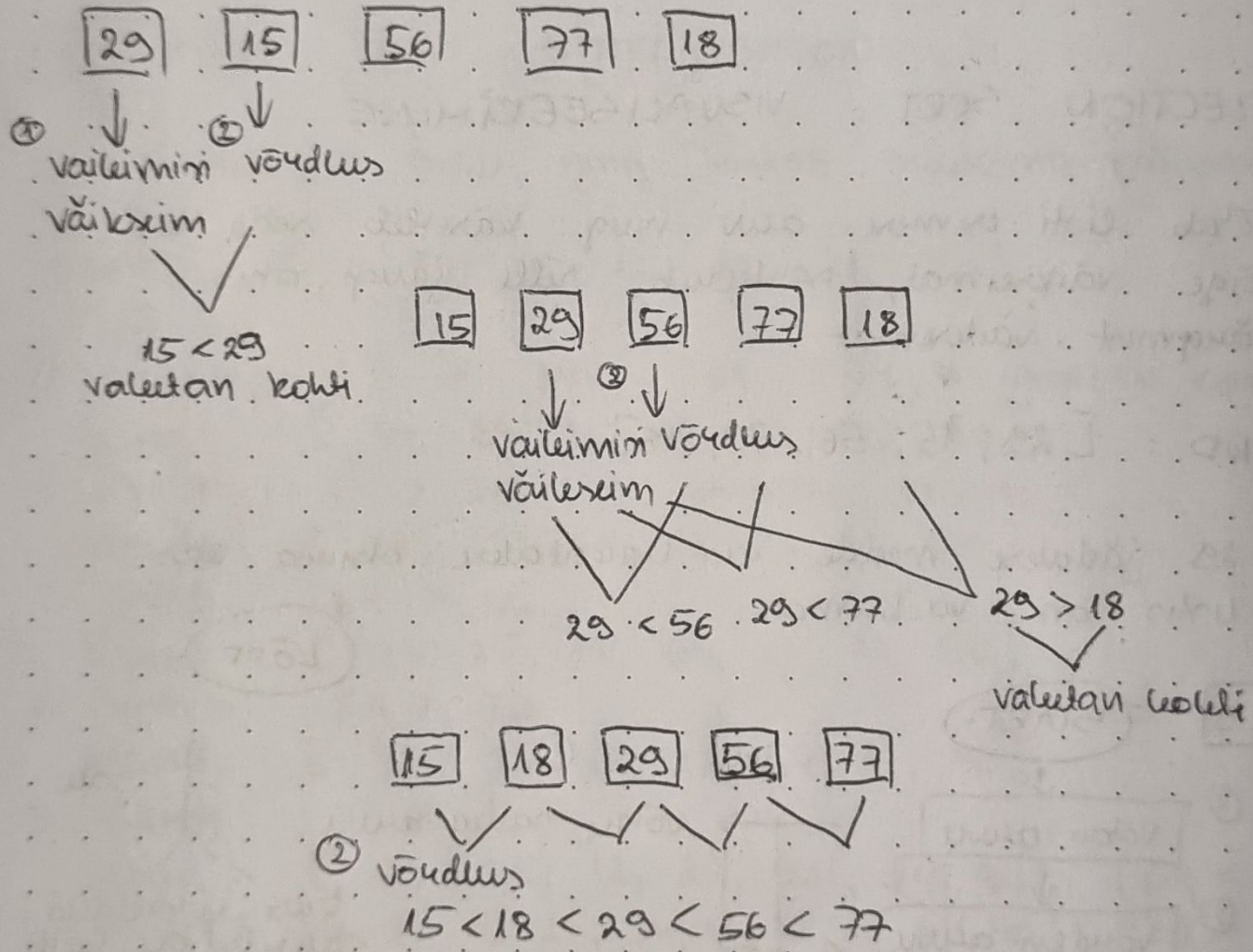
① BUBBLE SORT SIMULATSIOON

– Võrdleb 2 arvu ning tõstab suurema paremale, kui

LOEND : [64; 34; 25; 12; 22; 11; 90]

1. SAMM 64 & 34 64 > 34 ⇒ VAHETAN KOHAD
   loend: [34; 64; 25; 12; 22; 11; 90]
2. SAMM 64 > 25 → VAHETAN KOHAD
   loend: [34; 25; 64; 12; 22; 11; 90]
3. SAMM 64 > 12
   loend: [34; 25; 12; 64; 22; 11; 90]
4. samm 64 > 22
   loend [34; 25; 12; 22; 64; 11; 90]
5. SAMM 64 > 11
   loend [34; 25; 12; 22; 11; 64; 90] (64 < 90)
6. SAMM & loend [25, 34, 12, 22, 11, 64, 90] (34 > 25)
7. SAMM & loend [25, 12, 34, 22, 11, 64, 90] (34 > 12)
8. SAMM & loend [25, 12, 22, 34, 11, 64, 90] (34 > 22)
9. SAMM & loend [25, 12, 22, 11, 34, 64, 90] (34 < 64)
10. SAMM & loend [12, 25, 22, 11, 34, 64, 90] (25 > 12)
11. SAMM & loend [12, 22, 25, 11, 34, 64, 90] (25 > 22)
12. SAMM & loend [12, 22, 11, 25, 34, 64, 90] (25 > 11)

1. Võtan arvu 29
2. Võrdlen arvu 29 arvuga 15
3. Kas 29 < 15 - ei 29 > 15
~~2. Võrdlen arvu 29 a~~
3. Tõstan arvu 15 esimesse kohale
2. Võrdlen liitis järgmist arvu, nüüd on vaikses uuesti 29 järgmise arvuga 56 29 < 56
2. Võrdlen arvu 29 77-ga 29 < 77
2. Võrdlen arvu 29 18-ga 18 < 29
3. Tõstan arvu 18 kõige kohale



13. SAMM & loend [12, 22, 11, 25, 34, 64, 90] (12 < 22)
14. SAMM & loend [12, 11, 22, 25, 34, 64, 90] (22 > 11)
15. SAMM & loend [~~15~~ 11, 12, 22, 25, 34, 64, 90] sorteeritud

Lahti on kirjutatud iga MUUTUS, mida Bubble sort tegema peab.

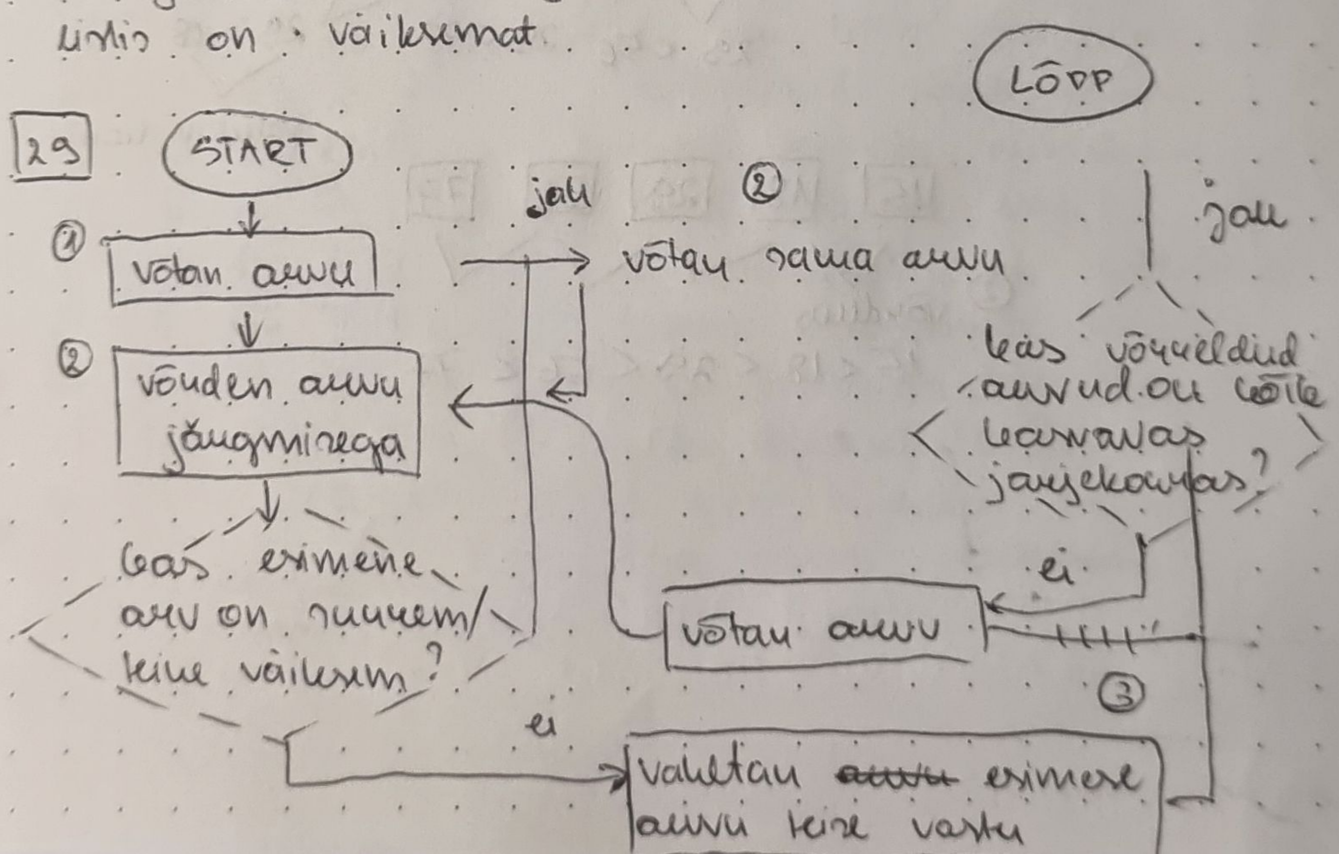
1. võrdlen kahte arvu
2. vahetan kohad
3. teen sama, mida sammus 1 ja kordan nii kaua kuni iga paremal asuv arv on suurem.

② SELECTION SORT VISUALISEERIMINE

– võtab listi esimese arvu ning käsitleb seda kui kõige väiksemat, sorteerib selle järgi, otsib järgmist väiksemat

LOEND: [29; 15; 56; 77; 18]

1. 29 jäetakse meelde ja hakatakse otsima kas listis on väiksemat



④ AJAKOMPLEKSUSTE ANALÜÜS

n^2

a) Halvimal juhul ajakomplekssus $O(n^2)$ on eelmainitutest kõigil - Bubble, Insertion kui ka Selection Sordil.

b) Kui vaadata üldiselt milline oleks parim siis Counting Sort - võimaldab juba etteantud vahemikus (1 - 100) sorteerida ning siis peab lugema ainult loendis elementide esinemist. Lisaks sobib see hästi just väikeste vahemikuga täisarvude jadas.

Kui aga põhineda eelmainitud sortimisalgoritmide siis Selection Sort? ~~Teg~~ Tegelikult neil ei tundu suurt vahet olevat ajalise keerukuse ja mälu poolest.

⑤ STABIILSUS JA ADAPTIIVSUS SORTIMISEL

a) Stabiilne on ~~ola~~ sorteerimisalgoritm, mis ~~jätab~~ jätab samasuguse väärtusega elemendid peale sorteerimist samasse järjekorda.

NT. [5; 7; 11; 5; 8; 5] - LOEND

1 2 3 4 5 6 - indeksid, mis näitavad kus element algselt listis paikneb

[5; 5; 5; 7; 8; 11] - sorteeritud loend

1 4 6 ← indeksid

Seda saaks edukalt teha Insertion sort sorteerimisalgoritmi kasutades. (see on stabiilne)

BOONUS

LOEND [8, 3, 5, 4, 7, 6, 2]

a) Bubble Sort [~~3, 5, 2~~ 3, 5, 4, 7, 6, 2, 8]

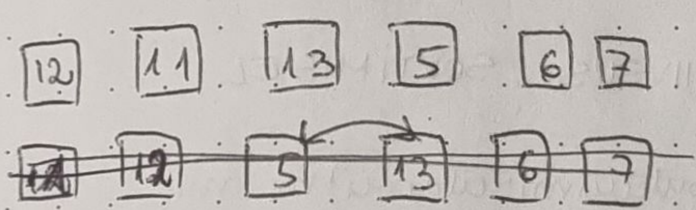
b) Selection Sort [2, 8, 3, 5, 4, 7, 6]

c) Insertion Sort [8, 5, 3, 4, 7, 6, 2]

③ INSERTION SORT PRAKTIKAS

LOEND [12, 11, 13, 5, 6, 7]

- võrdlen arvu sellest vasakul ja liigutan parempoolse arvu vasemale kui see on väiksem



11 12 13 5 6 7
11 12 5 13 6 7
11 12 5 6 13 7
11 12 5 6 7 13
11 5 12 6 7 13
11 5 6 12 7 13
11 5 6 7 12 13
5 11 6 7 12 13
5 6 7 11 12 13

siin jäi üks samm vahele

1. kui loend on osaliselt sorteeritud läheb loendi sorteerimiseks suhteliselt vähe samme vaja.
2. loogiliselt saab suhteliselt kiiresti elementide kohad ära vahetada, kui need on juba osaliselt sorteeritud
3. pole vajadust mingit osa uuesti sorteerida

b) Adaptiivseks sorteerimisalgoritmiks saab nimetada algoritmi mille kiirus sõltub sellest, kas loend on juba sorteeritud osaliselt või mitte. Näiteks kui loend on juba pooleldi sorditud vaja vähem ülesandes nt. 3 Insertion Sorti puhul, siis tuleb teha vähem tööd. Loengus 1 toodud tabeli puhul võib eelmainitud kolmest algoritmist adaptiivseks nimetada vaid Insertion ja Bubble Sorti, sest neil on parimal juhul ajaline keerukus O(n) ja keskmisel/halvimal O(n^2). Samas kui Selection Sortil on kõik sama – O(n^2).