**Kodutöö 3**

Otsingualgoritmide analüüs

**Ülesanne 1: Lineaarotsing**

Lineaarotsingu ajakompleksus on O(n), kuna töötleme ainult ühes tsüklis terve listi läbi. Parim oleks O(1), kui otsitav arv on esimesel positsioonil.

Ruumikompleksus on O(1), kuna otsingus kasutame võrdlemiseks ühte muutujat „i“.

Pilt, millel on kujutatud tekst, kuvatõmmis, Font, number

Kirjeldus on genereeritud automaatselt

Reaalmaailmas saaks kasutada lineaarotsingut, et teada saada mitmendal kohal mingi objekt järjekorras on. Näiteks tuludeklaratsiooni esitamise ajal on serverid üle koormatud ja sisselogimine on aeglustunud ning inimesed näevad mis kohal nad on järjekorras sisselogimisel.

**Ülesanne 2: Binary search**

Otsime elementi, mida ei ole loendis ehk peame läbi käima terve loendi.

Otsingu muster on N/2x, kuna iga iteratsiooniga me poolitame otsimisruumi.

Pilt, millel on kujutatud käekiri, Font, järjekord, kalligraafia

Kirjeldus on genereeritud automaatselt

Viimane iteratsioon on valimist 1 kuna oleme kõik valikud kõrvaldanud.

Pilt, millel on kujutatud Font, järjekord, käekiri, number

Kirjeldus on genereeritud automaatselt

T(1) võtab aega c1 kuna võrdlus ei sõltu mingist muust tegurist ning meil on võimalik leida binaarotsingu ajakompleksus milleks on O(logN).

Pilt, millel on kujutatud käekiri, tekst, Font, järjekord

Kirjeldus on genereeritud automaatselt

Võrreldes lineaarotsingu ja binaarotsingu ajakompleksust, siis binaarotsingul on eelis suurematel otsingutel.

Pilt, millel on kujutatud järjekord, Diagramm, tekst, Font

Kirjeldus on genereeritud automaatselt

Üks eelis lineaarotsingul binaarotsingu üle on, et loend võib olla juhuslikus järjekorras, aga binaarotsing nõuab, et sisend oleks sorteeritud. Sel juhul kui on antud suvaline andmetabel, mis ei ole sorteeritud, tasuks kasutada lineaarotsingut, et leida kas otsitav element on tabelis või mitte.

**Ülesanne 3: Jump search**

Pilt, millel on kujutatud tekst, käekiri, Font, number

Kirjeldus on genereeritud automaatselt

Optimaalne sammude arv on ruutjuur loendi pikkusest ning igal iteratsioonil kasvab samm m korda. Algoritmi ajakompleksus on O(√n).

Pilt, millel on kujutatud Font, järjekord, number, käekiri

Kirjeldus on genereeritud automaatselt

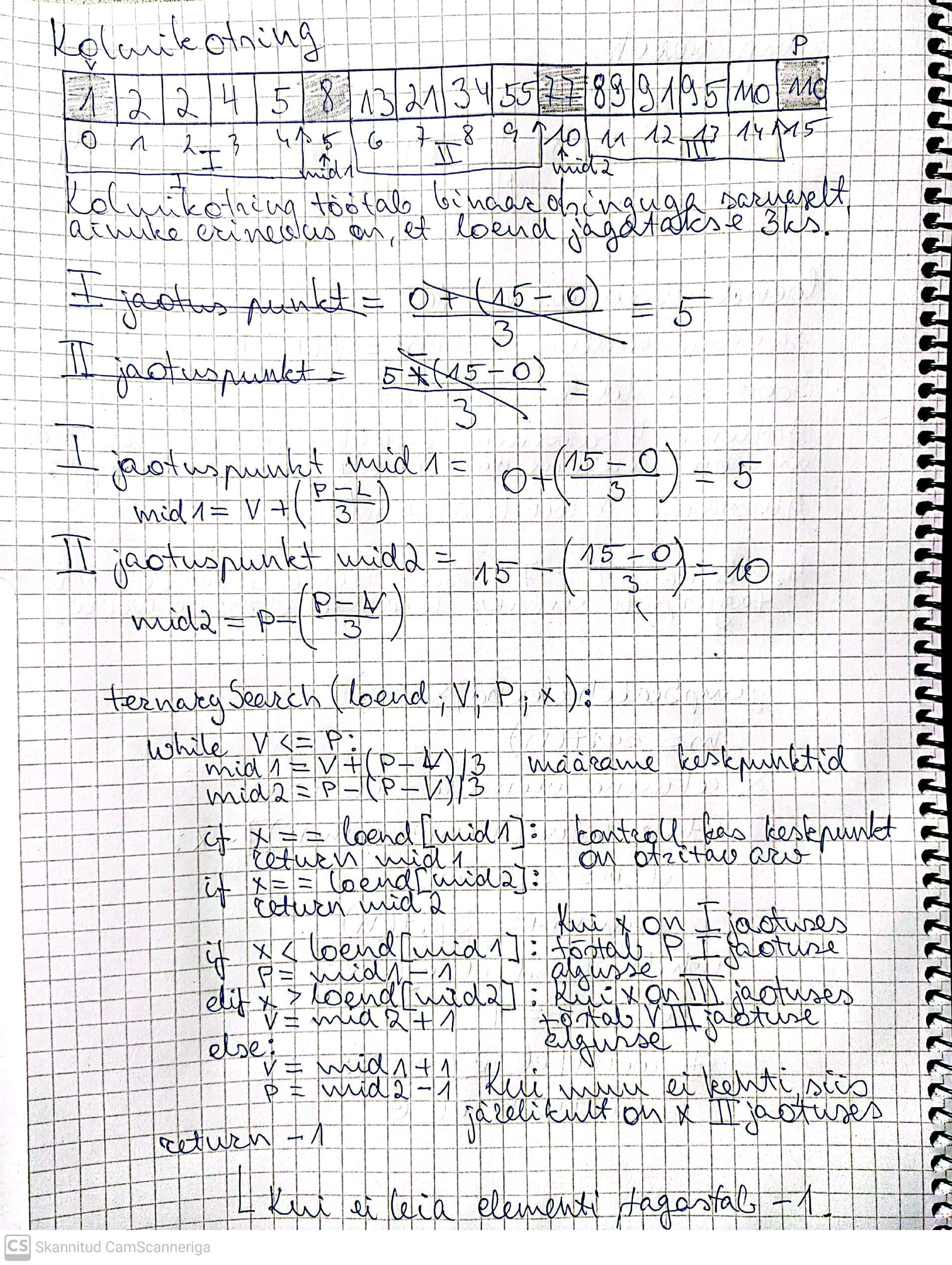
Pilt, millel on kujutatud järjekord, Diagramm, diagramm, Paralleelne

Kirjeldus on genereeritud automaatselt

StackOverflow foorumist leidisn selgituse, kus Jump Searchil võib olla eelis lineaar- ja binaarotsingu üle. Kui info on talletatud näiteks kasseti peal, kus tagasiliikumine on kulukas, sest kassett tuleb peatada, tagasi kerida ning siis uuesti läbi käia. Jump Searchiga avastaks otsitava elemendi kiiremini kuna teame, mis hetkel vaja tagasi kerida.

<https://stackoverflow.com/questions/58472462/in-binary-search-why-does-traversing-back-cost-more-than-traversing-forward>

**Ülesanne 4: Kolmikotsing ja Kahendotsing**



On teada, et binaarotsingu ajakompleksus on 0(log2N). Kolmikotsingu ajakompleksus on 0(log3N).



Joonis <https://www.codingninjas.com/studio/library/time-space-complexity-of-searching-algorithms>

Graafikul ütleks, et kolmikotsingul on eelis binaarotsingu üle.

Pilt, millel on kujutatud järjekord, Diagramm, diagramm, Paralleelne

Kirjeldus on genereeritud automaatselt

Kuigi mõlemad algoritmid töötavad sarnaselt teeb halvimal juhul kolmikotsing ikkagi rohkem võrdlusi kui binaarotsing.

|  |  |
| --- | --- |
|  |  |

**Ülesanne 5: Otsingualgoritmide praktiline rakendamine**

Kiire kontakti leidmine telefonis

Eeldame, et kontaktid on telefoni salvestatud korrektselt. Kontaktil on ees- ja perenimi ning järjestatud tähestikulises järjekorras. Ainult numbrina salvestatud kontakte meil ei ole.

Arvan, et kõige tõhusam algoritm oleks binaarotsing, sest olenemata kui suur on kontaktinimekiri, teeb binaarotsing uuritud otsingualgoritmidest kõige vähem võrdlusi võrreldes teistega.