Kodutöö 3

Seda koduülesannet võib täita kas üksi või paarilisega. Kui valite töötamise paaris, veenduge, et mõlema osaleja nimed oleksid dokumentatsioonis mainitud. Kogu kood, joonised ja dokumentatsioon tuleks üles laadida enda GitHubi hoidlasse. Hoidla nimi võib olla näiteks algoritmid-2020.
Kui ülesanne on lõpetatud, esitage Moodle'is hoidla link.

Ülesandeid võib lahendada nii paberil (või digitaalselt, visuaalselt), kui implementeerida kood vabalt valitud keeles.

Ülesanne 1: Lineaarotsing (*Linear Search*)

1. Rakendage *Linear Search* algoritm vabalt valitud programmeerimiskeeles.
2. Analüüsige oma algoritmi aja- ja ruumikomplekssust.
3. Arutlege lühidal, kuidas *Linear Search* algoritmi saab kasutada reaalmaailma rakendustes ja millised on selle piirangud.

Ülesanne 2: Kahendotsingu (*Binary Search*) rakendamine ja analüüs

1. Kirjutage programm, mis teostab *Binary Search*'i sorteeritud täisarvude massiivil.
2. Võrrelge **teoreetilises** analüüsis valminud *Binary Search*'i ja *Linear Search*'i aegkomplekssust.
3. Tooge näide stsenaariumist, kus Binary Search on kasulikum kui Linear Search, ja selgitage miks.

Ülesanne 3: *Jump Search*

1. Kirjutage lühike ülevaade *Jump Search* algoritmist, sealhulgas selle põhiprintsiibid ja pseudo-koodi näide.
2. Võrrelge *Jump Search*'i ajalist komplekssust *Linear Search*i ja *Binary Searchiga*.
3. Arutlege lühidalt stsenaariumide üle, kus Jump Search võib olla efektiivsem võrreldes Linear Searchi ja Binary Searchiga.

Ülesanne 4: Kolmikotsing ja Kahendotsing (*Ternary Search vs Binary Search*)

1. Kirjutage lühike ülevaade **Ternary Search** algoritmist, sealhulgas selle põhiprintsiibid ja pseudo-koodi näide.
2. Võrrelge *Ternary Search*'i ja *Binary Search*'i aegkomplekssust. (Kas mõõdetud tulemus, teoreetiline võrdlus, vms)
3. Arutlege lühidalt, kas Binary Search on üldiselt tõhusam kui Ternary Search ning millistes olukordades.

Ülesanne 5: Otsingualgoritmide praktiline rakendamine

Valige reaalmaailma probleem, kus otsingualgoritmi saab rakendada, ja kirjeldage konkreetse algoritmi sobivust stsenaariumi jaoks. Näited sellistest reaalmaailma probleemidest hõlmavad:

1. Konkreetse raamatu otsimine raamatukogu kataloogist.
2. Toote otsimine veebipoest.
3. Saadaolevate lendude leidmine lennufirma broneerimissüsteemis.
4. Patsiendi andmete tõhus taastamine haigla andmebaasist.
5. Kiire kontakti leidmine telefoni- või e-posti rakenduses.
6. Konkreetse filmi või saate otsimine voogedastusplatvormil.
7. Konkreetse faili või kausta leidmine arvutis.
8. Sõna otsimine sõnaraamatust.
9. Kasutaja või konkreetse sisu otsimine sotsiaalmeedia platvormidel.
10. Konkreetse eseme otsimine ettevõtte süsteemis.

Pärast ühe nende stsenaariumide või mõne muu sarnase reaalmaailma probleemi valimist kirjeldage, milline otsingualgoritm oleks kõige tõhusam ja miks. Arutage ka potentsiaalseid modifikatsioone, mis võiksid algoritmi teie valitud rakenduse jaoks optimeerida. Pidage meeles, et tuleb arvestada andmete omadusi, nagu suurus, struktuur ja uuendamise sagedus.

Boonusülesanne (2 punkti):

Kirjeldage lühidalt Fibonacci Search algoritmi ja selgitage, kuidas seda saab kasutada suuremahuliste andmete sorteeritud massiivides. Töötage välja konkreetne stsenaarium, kus Fibonacci Search oleks efektiivsem kui teised otsingualgoritmid, näiteks Binary Search või Ternary Search. Selgitage, miks valitud stsenaariumis on Fibonacci Search parem valik, arvestades andmestruktuuri omadusi ja otsinguvajadusi.