**Ülesanne 1 on leitav eraldi python failis binaarpuu.**

**Ülesanne 2**

**Min-kuhja:**

Omadused:

Väikseima elemendi asukoht on juur.

Iga sõlme võti on väiksem kui selle laste võtmed.

Täidetud täispuu struktuurilt.

Aja- ja Ruumikomplekssus:

Sisestamine (insert): O(log n) aeg, kus n on kuhja suurus.

Miinimumi leidmine (find-min): O(1) aeg.

Eemaldamine (extract-min): O(log n) aeg.

Kuhja ehitamine (build-heap): O(n) aeg.

Ruumikomplekssus: O(n).

Sorteerimine ja Prioriteetjärjekord:

Sobib hästi sorteerimiseks (Heap Sort): Ehitades min-kuhja, saab iga kord võtta miinimumi ja lisada selle sorteeritud järjekorda.

Prioriteetjärjekorra haldamine: Miinimumi leidmine on kiire, sobib kasutamiseks prioriteetjärjekorras.

**Max-kuhja:**

Omadused:

Suurima elemendi asukoht on juur.

Iga sõlme võti on suurem kui selle laste võtmed.

Täidetud täispuu struktuurilt.

Aja- ja Ruumikomplekssus:

Sisestamine (insert): O(log n) aeg, kus n on kuhja suurus.

Maksimumi leidmine (find-max): O(1) aeg.

Eemaldamine (extract-max): O(log n) aeg.

Kuhja ehitamine (build-heap): O(n) aeg.

Ruumikomplekssus: O(n).

Sorteerimine ja Prioriteetjärjekord:

Sobib hästi sorteerimiseks (Heap Sort): Ehitades max-kuhja, saab iga kord võtta maksimumi ja lisada selle sorteeritud järjekorda.

Prioriteetjärjekorra haldamine: Maksimumi leidmine on kiire, sobib kasutamiseks prioriteetjärjekorras.

**Kokkuvõte:**

Mõlemad kuhja struktuurid on efektiivsed operatsioonideks, nagu sisestamine, otsimine ja eemaldamine.

Sobivad hästi sorteerimiseks ja prioriteet järjekorra haldamiseks.

Min-kuhja sobib hästi väikseima elemendi kiireks leidmiseks, samal ajal kui max-kuhja sobib suurima elemendi kiireks leidmiseks.

Kuhja sorteerimise aeg on O(n log n), mis on efektiivne praktilistes rakendustes.

Kuhja struktuurid on kasulikud, kui on vaja kiiret juurdepääsu kas maksimumile või miinimumile.

**Ülesanne 3**

BST on binaarne puu, kus igal sõlmel on kaks alamsõlme (vasak ja parem). Igal sõlmel on võti (võrdlusväärtus), mida kasutatakse võrdluseks. Vasak alampuu iga sõlme võtmed on väiksemad kui sõlme võti. Parema alampuu iga sõlme võtmed on suuremad kui sõlme võti. Tänu sellele struktuurile saavutame kiire otsingu, lisamise ja eemaldamise.

**Põhielemendid**

Juursõlm: Puu sõlm, millele pole ühtegi vanemat sõlme.

Alamsõlmed: Sõlmed, mis on seotud ülemiste sõlmedega (vasak ja parem).

Lehtsõlm: Sõlm, mis ei oma alamsõlmi (puu ots).

Võti: Unikaalne väärtus, mida kasutatakse võrdlemiseks.

Vanem ja Laps: Vanem on ülemine sõlm, laps on alumine sõlm.

Kui BST muutub tasakaalustamatuks (nt kui lisatakse järjest suurenevad või kahanevad võtmed), võib puu muutuda lineaarseks ahelaks.

Tasakaalustamata puudel võib otsinguaeg suureneda O(n)-ni, kus n on puu kõrgus.

Lisaks võib lisamine või eemaldamine olla ebaefektiivne, kui puu struktuur on ebakorrapärane.

**Optimeerimine**

AVL Puu ja Punase-Must Puu: Need on tasakaalustatud BST tüübid, kus jälgitakse puu tasakaalustatust ning tehakse automaatsed tasakaalustavad operatsioonid.

AVL Puu: Iga sõlme tasakaalu faktor (kõrguse vahe parema ja vasaku alampuu vahel) on lubatud olla ainult -1, 0 või 1.

Punane-Must Puu: Puudel on lisaks tasakaalustatusele värvilised atribuudid, mis tagavad tasakaalu pärast igat operatsiooni.

**Ülesanne 4**

**Punase-Musta Puu Andmestruktuur:**

Punase-musta puu on tasakaalustatud binaarne otsingupuu, kus igal sõlmel on lisaks võtmele ka värv (punane või must).

**Põhilised omadused:**

Iga sõlme värv on kas must või punane.

Juursõlm on alati must.

Kõik lehtsõlmed (null-sõlmed) on mustad.

Kui sõlm on punane, siis mõlemad selle lapse sõlmed on mustad.

Igal lihtteel (mis on tee juurest leheni) on sama arv musti sõlmi, mis on tuntud kui mustade kõrgus.

**Punase-Musta Puu ja Binaarse Otsingupuu Võrdlus:**

Binaarne otsingupuu (BST) ja punase-musta puu (RBT) on mõlemad binaarsed otsingupuud, kuid RBT lisab värvireeglid ja tasakaalustamise, et tagada parem tõhusus.

Teoreetiline Võrdlus:

**Otsingu Aeg:**

BST: Parimal juhul O(log n), kuid võib lõppeda O(n)-ga (kui puu muutub lineaarseks).

RBT: O(log n) kõikidel juhtudel, kuna värvireeglid tagavad tasakaalustatuse.

Lisamine ja Eemaldamine:

BST: Parimal juhul O(log n), kuid võib olla O(n) halvimas juhul.

RBT: O(log n) kõikidel juhtudel tänu tasakaalustatuse tagamisele värvireeglite abil.

**Tasakaalustamine:**

BST: Sõltub sisestatud võtmetest; võib muutuda tasakaalustamataks, eriti kui võtmed sisestatakse järjestikku.

RBT: Säilitab tasakaalu tänu värvireeglitele; tasakaalustab end pärast iga operatsiooni.

**Värvireeglid:**

Punase-musta puu värvireeglid tagavad, et puu püsib tasakaalus ka pärast lisamist ja eemaldamist.

Tasakaalustamiseks võib punase-musta puu muuta sõlmi värvidega ja viia läbi pöördeid, et säilitada tasakaalu.

Need omadused aitavad vältida puu lineaarseks muutumist, tagades kiire otsingu, lisamise ja eemaldamise.

**Ülesanne 5**

**AVL Puu Andmestruktuur:**

AVL puu on tasakaalustatud binaarne otsingupuu, kus igal sõlmel on tasakaalufaktor, mis mõõdab vasaku ja parema alampuu kõrguste erinevust.

Peamised omadused:

Iga sõlme tasakaalufaktor on -1, 0 või 1.

Kõik puu alam-puud on ise tasakaalustatud AVL puud.

Juursõlm on tasakaalustatud, mistõttu kogu puu on tasakaalustatud.

**Otsingu Aeg:**

Mõlemad: O(log n) kõikidel juhtudel, kuna mõlemad puud tagavad tasakaalustatuse.

Lisamine ja Eemaldamine:

AVL Puu: Lisamine ja eemaldamine võivad olla veidi kallimad kui punase-musta puu puhul, kuna AVL puu nõuab rangemat tasakaalustamist.

Punase-Musta Puu: Võib olla veidi kiirem lisamise ja eemaldamise osas, kuna selle tasakaalustamine ei ole nii range kui AVL puu puhul.

Tasakaalustamine:

AVL Puu: Kasutab rangemaid tasakaalustamise reegleid, mis nõuavad rohkem keerukaid operatsioone.

Punase-Musta Puu: Kasutab vähem ranged värvireegleid, mis võivad lubada rohkem paindlikkust tasakaalustamisel.

Millistes Rakendustes Eelistada:

AVL Puu: Kui otsingud on sagedasemad kui lisamised ja eemaldamised.

Rakendustes, kus otsingu aeg on kriitilise tähtsusega ja mõned lisakulud lisamisel ja eemaldamisel on vastuvõetavad.

Kui puu struktuur peab olema täpselt tasakaalus.

Punase-Musta Puu:

Kui lisamised ja eemaldamised on sagedasemad kui otsingud.

Rakendustes, kus on oluline säilitada kiiremad lisamise ja eemaldamise operatsioonid isegi siis, kui see tähendab veidi pikemat otsinguaega.

Kui lubatakse rohkem tasakaalustamise paindlikkust ja täpsust.