Työn tavoitteena on toteuttaa tietokoneohjelma, joka järjestää annetun aineiston tasapainoitettuun binääriseen hakupuuhun. Tämän jälkeen puusta haetaan aineistoa. Puuhun voidaan edelleen lisätä uutta aineistoa tai hakea aineistoa. Lisäykset ja haut tapahtuvat aineiston avaimen perusteella.

Kuvan 1. puussa avaimina ovat olleet **2, 4, 6, 8, 10, 12, 14, 30, 28**. Jokaisen avaimen on määrittänyt aineisto, mutta nyt vain avainten arvot on esitetty puussa. Tämä riittää myös harjoitustyössä, eli käsitellään vain avaimia.

Kuva 1. Tasapainotettu binäärinen hakupuu.

Ohjeita

Harjoitustyössä tehtävänä on tehdä ohjelma, joka

- lukee tiedostosta avaimet
- lisää luetut avaimet puuhun
- tulostaa syntyneen puurakenteen
- hakee avaimen mukaisen tiedon puusta (hakee avaimen)
- lisää yksittäisiä avaimia puuhun

Tietorakenteena on tasapainoitettu binäärinen hakupuu, jonka toteutus voi tapahtua taulukolla tai linkitetyllä rakenteella. Ohjelman tulee antaa välitulosteita siten, että käyttäjä voi nähdä puun rakentumisen vaiheet. Samoin yksittäiset lisäykset tulee havainnollistaa. Haku ilmoittaa avaimen olemassaolon puussa: joko avain löytyy tai sitten ei. Löytyykö puusta avaimet **6**, **1**, **10** ja **16**?

Käytä testiaineistona edellisen esimerkin avainjonoa annetussa järjestyksessä. Jatka sitten lisäämällä yksittäin avaimet **26**, **24**, **22**, **20**, **18**, **16** tässä annetussa järjestyksessä. Löytyykö avaimet **10**? ja **26**? Entäpä avain **32**?

Puun tulostuksen tulee olla sellainen, että puun rakenne näkyy (avaimet tulostettuna puumaiseen muodostelmaan - grafiikkaa tms. ei vaadita). Vinkki: puun tulostus sivuttain (juurisolmu vasemmalla, oikea lapsi ylempänä, vasen alempana) on huomattavasti helpompi toteuttaa kuin pystyyn.

Mikä on haun kompleksisuus? Toteutuuko teoreettinen kompleksisuus toteuttamassasi ohjelmassa? Testaa ohjelmaasi, kun avaimia on 10, 100, 1000, 10000, 100000 (tai vielä enemmän).

Pohdi myös seuraavia asioita: