suurempi, hakua voidaan jatkaa listan alkuosan loppuosasta (siis alkuperäisen listan toisesta neljänneksestä). Näin jatketaan, kunnes tarkasteltavan hakualueen keskimmäinen alkio on sama kuin haettava arvo (jolloin haluttu arvo on löytynyt) tai hakualue on tyhjä (joilloin voidaan todeta, että haluttua arvoa ei ole listassa lainkaan).

Tällaista hakutapaa kutsutaan *binäärihauksi*. Seuraavaksi vielä binäärihaun periaate vähän täsmällisemmin esitettynä:

1. Aloita niin, että hakualueena on koko alkuperäinen lista.
2. Jos hakualue on tyhjä eli hakualueen alaindeksi on suurempi kuin hakualueen yläindeksi, lopeta haun suoritus. Haettavaa arvoa ei ole listassa lainkaan. Muussa tapauksessa laske hakualueen keskimmäisen alkion indeksi (Keskimmäinen tarkoittaa tässä sitä alkiota, joka on hakualueen keskimmäisessä paikassa.)
3. Vertaa haettavaa arvoa hakualueen keskimmäiseen alkioon:
   (a) Jos haettava arvo ja keskimmäinen alkio ovat yhtäsuuret, haettava arvo on löytynyt. Lopeta haun suoritus ja palauta keskimmäisen alkion indeksi.
   (b) Jos haettava arvo on pienempi kuin hakualueen keskimmäinen alkio, jatka kohdasta 2, mutta niin, että hakualueena on nykyisen hakualueen alkupuolisko.
   (c) Jos haettava arvo on suurempi kuin hakualueen keskimmäinen alkio, jatka kohdasta 2, mutta niin, että hakualueena on nykyisen hakualueen loppupuolisko

Seuraavaksi funktio, joka hakee binäärihaulla ensimmäisenä parametrina annetusta listasta toisena parametrina annetun arvon. Funktion sisällä pidetään muuttujien `ala` ja `yla` avulla tietoa siitä, mikä on hakualue milläkin hetkellä. Aluksi `ala`-muuttujan arvoksi asetetaan nolla ja `yla`-muuttujan arvoksi listan viimeisen alkion indeksi. Joka kierroksella lasketaan aina hakualueen keskimmäinen indeksi, jonka arvo tallennetaan muuttujaan `keski`. Tämän jälkeen verrataan haettavaa arvoa hakualueen keskimmäisellä indeksillä olevaan alkioon. Vertailun tuloksen mukaan joko poistutaan funktiosta tai muutetaan joko hakualueen ylä- tai alaindeksin arvoa.

```
def binaarihaku(lista, arvo):
    ala = 0
    yla = len(lista) - 1
    while ala <= yla:
        keski = (ala + yla) / 2
        if lista[keski] == arvo:  # arvo loytyi, lopetetaan
            return keski
        elif lista[keski] < arvo:
            ala = keski + 1  # jatketaan hakua loppuosasta
        else:
            yla = keski - 1  # jatketaan hakua alkuosasta
    return -1   # hakualue on tyhja, mutta arvoa ei loytynyt
```

Annettu funktio toimii siis oikein vain siinä tapauksessa, että parametrina annetussa listassa luvut ovat suuruusjärjestyksessä. Funktio ei mitenkään tarkista sitä, että

tämä pitää paikkansa. Jos listan alkiot ovat satunnaisessa järjestyksessä, funktion paluuarvo voi olla täysin väärä eikä funktio anna mitään ilmoitusta siitä, että se ei toiminut oikein.

Binäärihaku on hyvin tehokas verrattuna peräkkäishakuu. Tarkastellaan esimerkiksi hakua listasta, jossa on miljoona alkiota. Peräkkäishakua käytettäessä joudutaan pahimmassa tapauksessa suoritamaan funktiossa `hae-pisteet` olevaa `for`-käskyä miljoona kierrosta (kaikki listan alkiot pitää käydä läpi, jos haettava alkio ei ole listassa). Binäärihakua käytettäessä riittää, että `while`-käskyä suoritetaan pahimmassakin tapauksessa noin 20 kierrosta. Vielä paremmin ero tulee näkyviin, jos mietitään tilannetta, jossa listan koko kasvaa kahteen miljoonaan. Peräkkäishakua käytettäessä pahimmassa tapauksessa tarvittavien `for`-käskyn kierrosten määrä kasvaa miljoonalla, kun taas binäärihaussa `while`-käskyn kierrosten määrä kasvaa vain yhdellä.

Alla on vielä esimerkkinä koko ohjelma, jossa on ensin pyydetty käyttäjää antamaan pelipisteet suuruusjärjestyksessä ja sen jälkeen on käytetty `binaarihaku`-funktiota halutun arvon hakemiseen.

```
#pistehaku-binaarihaulla.py

def lue_pisteet():
    print "Anna pelaajien pisteet suuruusjarjestyksessa pienimmasta alkaen."
    rivi = raw_input("Monenko pelaajan pisteet haluat antaa? ")
    lkm = int(rivi)
    pistelista = [0] * lkm
    i = 0
    while i < len(pistelista):
        rivi = raw_input("Anna seuraavan pelaajan pisteet: ")
        pistelista[i] = int(rivi)
        i += 1
    return pistelista


def binaarihaku(lista, arvo):
    ala = 0
    yla = len(lista) - 1
    while ala <= yla:
        keski = (ala + yla) / 2
        if lista[keski] == arvo:  # arvo loytyi, lopetetaan
            return keski
        elif lista[keski] < arvo:
            ala = keski + 1  # jatketaan hakua loppuosasta
        else:
            yla = keski - 1  # jatketaan hakua alkuosasta
    return -1   # hakualue on tyhja, mutta arvoa ei loytynyt


def main():
    pisteet = lue_pisteet()
    rivi = raw_input("Mita arvoa etsitaan? ")
    haettava_arvo = int(rivi)
```