Pikalajittelualgoritmi

Kari Peisa

Hajoita ja hallitse

Ositus

- Valitse johtava alkio listasta
- Uudelleenjärjestä alkiot listassa siten, että kaikki johtavaa alkiota pienemmät tai sen kanssa yhtäsuuret sijoittuvat sen vasemmalle ja suuremmat oikealle puolelle.

Hallitse (Rekursio)

- Lajittele johtavan alkion vasemmanpuoleinen ja oikeanpuoleinen osalista kutsumalla niitä rekursiivisesti samalla pikalajittelualgoritmilla, mikäli osalistan pituus >1.

Yhdistä

- Koska kaikki osalistat on lajiteltu samassa alkuperäisessä listassa, ei ole tarvetta niiden yhdistämiseen.

Kompleksisuus

Pahimmassa tapauksessa pikalajitelun ositus tuottaa johtavan alkion suhteen jaon 0 - (N-1) jokaisessa rekursion kutsussa. Tämä merkitsee kompleksisuutta $O(N^2)$.

Syy: Osituksen tuottama jako 0 - (N-1) kaikille osalistoille vaatii eri vaiheissa vähintään N + (N-1) + (N-2) + … +2+1 = $\frac{N(N+1)}{2}$ alkion siirron, toisin sanoen, neliöllisen kompleksisuuden $O(N^2)$. Näin käy esimerkiksi, jos lista on valmiiksi väärässä suuruusjärjestyksessä ja valitaan aina listan ensimmäinen alkio johtavaksi alkioksi.

Kun ositus tuottaa jaon *k - (N-1-k)*, missä k>0, pikalajittelun kompleksisuus on O(N Log N)

Syy :

- 1) Ositusvaihe kullekin osalistalle on mahdollisine alkioiden siirtoineen lineaarinen läpikäyntialgoritmi eli O (N).

- 2) Osalistojen rekursiiviset kutsut muodostavat rekursiopuun, jonka syvyys on O(LOG N). (Rekursiokutsut kohdistuvat kullakin kerralla aina vaan pienempiin osiin alkuperäistä listaa)

Seuraava funktio havainnollistaa rekursiokutsujen muodostaman puun rakennetta pikalajittelussa. Oletetaan osituksen tuottavan vakiosuhteen 1 - 9 jokaisella rekursiokutsulla.

Funktio **fractions[n, fraction]** palauttaa kaikki peräkkäiset kokonaislukuosuudet, jotka voidaan saada vaiheittain, kun n (n > 1) alkiota jaetaan e.m. suhteessa niin monta kertaa kuin se on mahdollista.

```
Clear[fractions, fractionsLength]
```

In[1]:=
```
fractions[n_Integer, fraction_] :=
 NestWhileList[Floor[fraction #] &, n, # > 1 &] /; n > 1 && 0 < fraction < 1
```

In[2]:=
```
fractions[100, 0.9]
Length[%]
```

Out[2]=
```
{100, 90, 81, 72, 64, 57, 51, 45, 40, 36, 32,
 28, 25, 22, 19, 17, 15, 13, 11, 9, 8, 7, 6, 5, 4, 3, 2, 1}
```

Out[3]=
```
28
```

Funktio **fractionsLength[n, fraction]** palauttaa tarvittavien rekursiokutsujen lukumäärän. Seuraavassa grafiikassa tulostetaan kutsujen lukumäärät eri arvoille n aina arvoon 10 000 saakka.

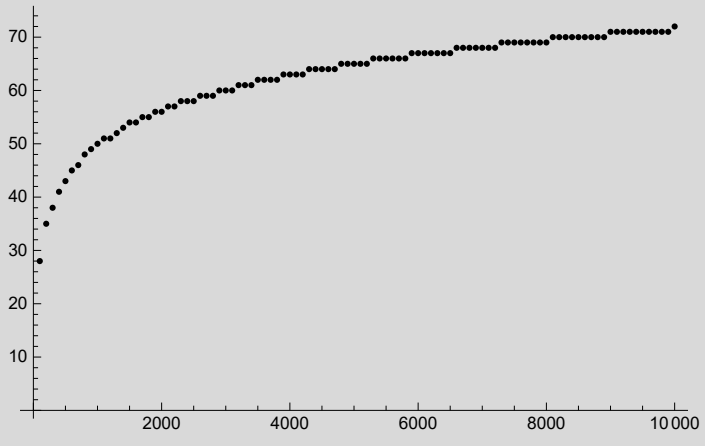
In[4]:=
```
fractionsLength[n_Integer, fraction_] :=
 Length[NestWhileList[Floor[fraction #] &, n, # > 1 &]] /; n > 1 && 0 < fraction < 1
```

In[5]:=
```
fractionsLength[100, 0.9]
```

Out[5]=
```
28
```

In[12]:=
```
fig1 = ListPlot[
   Table[{n, fractionsLength[n, 0.9]}, {n, 100, 10000, 100}], PlotStyle → Black]
```

Out[12]=

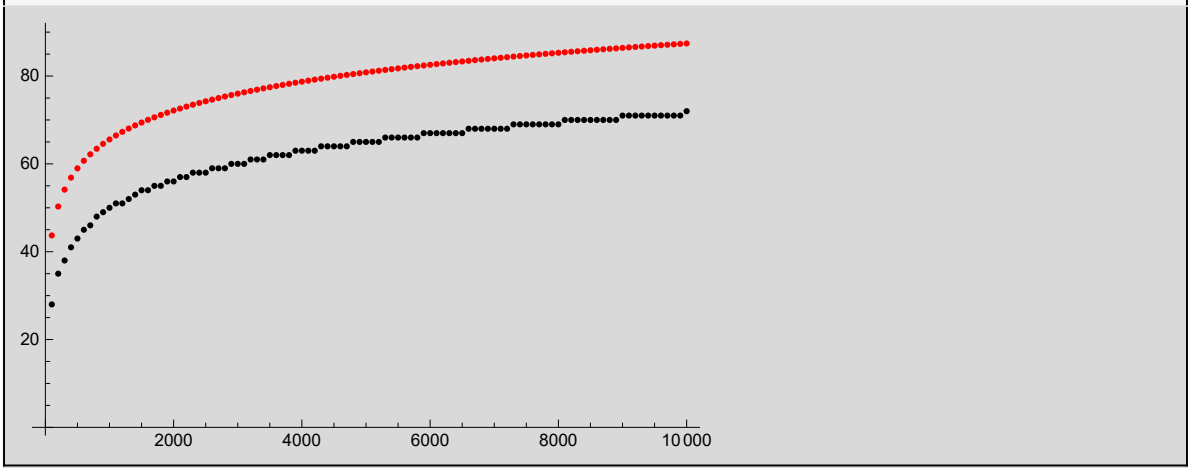


Verrataan rekursiokutsujen lukumäärien kasvua logaritmifunktioon $\text{Log}_{\frac{10}{9}} N$

In[10]:=
```
fig2 = ListPlot[Table[{n, Log[10/9, n]}, {n, 100, 10000, 100}], PlotStyle → Red];
```

In[13]:=
```
Show[fig2, fig1]
```

Out[13]=



Keskimääräisessä tapauksessa pikalajittelun kompleksisuus on O(N Log N)

Syy :

- 1) Kun ositusvaihe tuottaa jaon k - (N-1-k) millä tahansa arvolla k>0, rekursiokutsut muodostavat rekursiopuun, jonka syvyys on jokin logaritmifunktio N:stä.
- 2) Kun alkiot ovat vertailun suhteen satunnaisesti jakaantuneet listaan, vain muutamissa harvoissa tapauksissa saadan jako 0 - (N-1), ja kompleksiuus O(N Log N) saavutetaan.