811312A Tietorakenteet ja algoritmit, 2018 - 2019, Harjoitus 3

Harjoituksessa käsitellään algoritmien aikakompleksisuutta.

Tehtävä 3.1 Kuvitteelliset algoritmit A ja B lajittelevat syötteenään saamat lukutaulukot. Algoritmi A käyttää $32 \cdot nlg(n)$ operaatiota ja algoritmi B käyttää $3 \cdot n^2$ operaatiota, kun taulukon koko on n. Arvioi, minkä kokoisten taulukoiden lajitteluun kannattaa käyttää algoritmia A ja minkä kokoisten taulukoiden lajitteluun kannattaa käyttää algoritmia B.

Tehtävä 3.2 Lataa koneellesi (linkki alempana) ohjelmatiedosto **`h3_t2.cpp`**. Käännä ohjelma ja suorita se (voi tehdä luokassa esimerkiksi CodeBlocksilla tai kääntämällä komentoriviltä). Komentoriviltä käännetään antamalla syöte

g++ h3_t2.cpp –o t2.exe (Windowsissa) tai **g++ h3_t2.cpp –o t2** (Linuxissa).

Ohjelmassa voidaan valita suoritukseen jokin kolmesta algoritmista (A, B ja C) halutulla syötteen koolla. Tee ohjelman avulla seuraavaa:

a) Taulukoi algoritmin A suoritukseen kuluneita aikoja, kun syötteen koko vaihtelee välillä 500 – 20 000. Pyri arvaamaan mikä on suoritusaika, kun syötteen koko on 50 000. Tarkista arvauksesi suorittamalla ohjelma.
b) Taulukoi algoritmin B suoritukseen kuluneita aikoja, kun syötteen koko vaihtelee välillä 25 – 200. Pyri arvaamaan mikä on suoritusaika, kun syötteen koko on 400. Tarkista arvauksesi suorittamalla ohjelma.
c) Taulukoi algoritmin C suoritukseen kuluneita aikoja, kun syötteen koko vaihtelee välillä 100 000 – 5 000 000. Pyri arvaamaan mikä on suoritusaika, kun syötteen koko on 80 000 000. Tarkista arvauksesi suorittamalla ohjelma.

Arvioi vielä, mikä olisi algoritmien A ja B suoritusaika, kun syötteen koko on 1000 000.

Tehtävä 3.3 Seuraava lisäyslajittelu on hyvä pienten taulukoiden lajittelualgoritmi. Algoritmin oikeellisuus on perusteltu luennoilla. Määritä (ja perustele) algoritmin kompleksisuusluokka. Riittää tarkastella ainoastaan huonointa tapausta.

```
Syöte: Taulukko A[0,..,n-1], n >= 1
Tuloste: Taulukon luvut järjestyksessä A[0] <= … <= A[n-1]

LISAYS(A)
 1.   for j = 1 to n-1
 2.      k = A[j]
 3.      i = j-1
 4.      while i>=0 && A[i]>k
 5.         A[i+1] = A[i]
 6.         i = i-1
 7.      A[i+1] = k
 8.   return
```

Ohjelmointitehtävä toisella puolella ->

Ohjelmointitehtävä

Tehtävä 3.4. Ohjelmissa tarvitaan usein erilaisten objektien joukkojen satunnaista järjestämistä. Tällaista algoritmia sanotaan **satunnaisen permutaation** tuottamiseksi. Tarkastele seuraavaa algoritmia, jolla voidaan tuottaa lukujen 1, …, n satunnaisen järjestyksen sisältävä kokonaislukutaulukko A[0,..,n-1].

```
Syöte: Luku n >= 1. Taulukko A[0,..,n-1].
Tulostus: Taulukossa A luvut 1..n satunnaisessa järjestyksessä.
RANDPERM(n,A)
 1. for i = 0 to n-1 // Taulukon alustus järjestykseen 1,2,..,n
 2.    A[i] = i+1
 3. for i = 0 to n-1
 4.    x = satunnaisluku väliltä 0..i
 5.    vaihda A[i] ja A[x]
 6. return
```

Ohjelmoi algoritmi (joko C- tai Python-kielellä) ja mittaa suoritukseen kuluvaa aikaa kasvattaen taulukon kokoa. Minkä kokoisilla taulukoilla on käytännöllistä suorittaa algoritmia? Mikä on algoritmin aikakompleksisuusluokka, kun syötteen koon mitta on taulukon koko?

Sovelletaan em. mainittua algoritmia. Korttipakassa on 52 korttia, jotka jakaantuvat neljään maahan (hertta, ruutu, risti, pata), joissa kaikissa on 13 korttia numeroituina 1, ...,13. Pokerissa jaetaan kaikille pelaajille aluksi viisi korttia. Halutaan tietää, mikä on pelaajan todennäköisyys saada jaossa väri, eli kaikki kortit samaa maata. Tätä voidaan arvioida simuloimalla: Mallinna pakan kortteja luvuilla 1, ...,52, tuota suuri määrä lukujen 1, …,52 permutaatioita ja laske, kuinka monessa näistä ensimmäiset viisi lukua vastaavat saman maan kortteja. Tämän lukumäärän suhde tuotettujen permutaatioiden lukumäärään arvioi haluttua todennäköisyyttä. (Laskettu todennäköisyys on n. 0.00198)

Neuvoja:

C-ohjelma. Satunnaisluvun generoiminen ja jonkin operaation keston laskeminen sekunneissa tapahtuu seuraavasti:

```
#include <stdlib.h>
#include <time.h>
clock_t start,end;
double totaltime;
// Satunnaisluku
int rand_x;
rand_x = rand();
// Isoilla taulukoilla kannattaa varmistaa että satunnaisluku
// on varmasti 32-bittinen ja kutsua funktiota
unsigned int bigRandom(){
      unsigned int random =
            (((unsigned int) rand() <<  0) & 0x0000FFFF) |
            (((unsigned int) rand() << 16) & 0xFFFF0000);

      return random;
}

// Operaation kesto
start = clock();
// OPERAATIO
end = clock();
totaltime = (double)(end-start)/CLOCKS_PER_SEC;
```

Pythonissa vastaavat operaatiot voidaan tehdä seuraavasti:

```
import time
import random
#Satunnaisluku x niin, että väliltä a <= x <= b
x = random.randint(a,b)
#Operaation kesto
start = time.perf_counter()
#OPERAATIO
end = time.perf_counter() - start
```