Heimadæmi 4

Magnús Daníel Einarsson

February 2023

Verkefni 1

a) Bætið við klasann Card úr æfingadæminu aðferðinni toString(), sem skilar streng með gildi spilsins sem hægt er að prenta út. Þið getið notað ensku upphafsstafina fyrir spaða (S), hjarta (H), tígul (D) og lauf (C). Sömuleiðis fyrir mannspilin: ás (A), kóngur (K), drottning (Q) og gosi (J). Þannig á aðferðin að skila "H-K"fyrir hjartakóng, "C-5"fyrir laufafimmu, o.s.frv.

b) Skrifið forritið CardDeal, sem tekur á skipanalínunni töluna k sem er á bilinu 1 til 52. Forritið prentar þá út k spil sem valin eru af handhófi úr spilastokki. Til þess að við prentum ekki sömu spilin út aftur, þá er best að búa til 52-spila fylki. Það er fyllt af öllum mögulegum spilum í venjulegri röð og þetta fylki er síðan stokkað. Til þess getið þið notað aðferðina shuffle út StdRandom. Síðan prentar forritið út k fyrstu stökin í fylkinu. Skilið kóðanum fyrir forritið og skjáskoti af keyrslu.

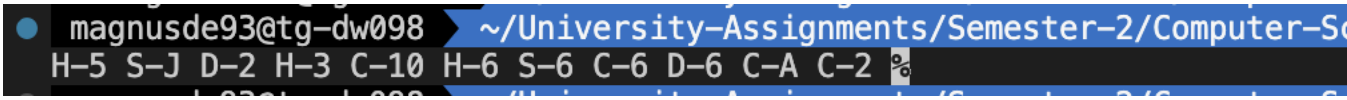
Lausn:

a)
```
public String toString(){
        String suits[] = {"S", "H", "D", "C"};
        String ranks[] = {"A","2", "3", "4","5","6","7","8","9","10","J","Q","K"};
        return suits[this.suit] + "-" + ranks[this.rank-1];
    }
```

b)
```
import java.util.Random;
import edu.princeton.cs.algs4.*;

public class CardDeal {
    public static void main(String[] args) {
        Card[] count = new Card[52];
        int k = Integer.parseInt(args[0]);

        for (int i = 0; i <= 3; i++){
            for (int l = 0; l<=12; l++){
                count[i*13+l]=new Card(i,l+1);
            }
        }
        StdRandom.shuffle(count);
        for (int m = 0; m <= k; m++){
           StdOut.printf(count[m] +  " ");;
        }
      }
}
```



Verkefni 2

[Valröðun] Í útfærslunni á Valröðun í bókinni (Selection.java) þá er ekki athugað hvort við þurfum að víxla á stökum a[i] og a[min]. Ef i er jafnt og min þá er þessi víxlun óþörf. Bætið við if-setningu á undan víxlunarskipuninni í röðunarfallinu sort sem athugar hvort það þurfi að víxla. Bætið tímamælingarkóða við báðar útgáfurnar og keyrið þær svo á skránni 32Kints.txt og athugið hvort það sé einhver hraðamunur. Þið ættuð að keyra hvort útgáfu a.m.k. þrisvar sinnum og taka meðaltalið, því það er alltaf einhver breytileiki í keyrslutímanum. Skilið breytta fallinu og niðurstöðum tímamælinganna.

Lausn:

```
public static void sort(Object[] a, Comparator comparator) {
        int n = a.length;
        for (int i = 0; i < n; i++) {
            int min = i;
            for (int j = i+1; j < n; j++) {
                if (less(comparator, a[j], a[min])) min = j;
               }
            if (a[i] != a[min]) exch(a, i, min);
               assert isSorted(a, comparator, 0, i);
        }
        assert isSorted(a, comparator);
    }
```

Meðaltal tíma áður en if setning er bætt við er 2.75sek. Eftir að if setningu var bætt við fór tíminn upp í 2.85sek.

Verkefni 3

[Röðun] Þetta er spurning um hegðun Valröðunar og Innsetningaröðunar á tilteknu N-staka inntaksfylki:

a) Öll stökin í fylkinu hafa sama gildi:

   i. Hversu marga samanburði notar Valröðun? Rökstyðjið!

   ii. Hversu marga samanburði notar Innsetningaröðun? Rökstyðjið!

b) Fylkið er óraðað en inniheldur aðeins tvö ólík gildi, A og B. Fjöldi staka af hvoru gildi er óþekktur:

   i. Hversu marga samanburði notar Valröðun? Rökstyðjið!

   ii. Hversu marga samanburði notar Innsetningaröðun? Rökstyðjið!

Lausn:

a) i. Það skiptir ekki máli hvernig stök eru í fylki þegar það kemur að valröðun. Stakið skoðar alltaf öll stök fyrir framan sig og skiptir við ef stakið er hærra en this.stak. Samanburðurinn er því lokað form á summunni n sem er $\frac{n(n-1)}{2}$

   ii. Þegar við beytum innsetningarröðun á fylki horfir this.stak á this.stak-1 og sér hvort það sé einhver munur á þeim og skiptir við það stak ef this.stak er minna virði en this.stak-1. Ef öll stök eru eins þá þurfa samt öll stök(nema fyrsta

stakið) á berjast við nágranna sinn um ekkert því öll stökin eru jöfn og fá jafn mikla vinna nema stak 0 því það á engann nágranna á undan sér. Þess vegna eru n-1 samanburðir í fylki þar sem öll stök eru jöfn.

b) i. Eins og kom fram í svari i í a lið þá skiptir ekki máli hversu mörg stök eru í fylkinu og hvernig stök eru í fylkinu. Öll stök hafa sinn persónuleika og eru ótrúlega forvitinn. Þau vilja því öll máta sig við öll önnur stök sem eru fyrir framan sig. Jafnan er því ennþá $\frac{n(n-1)}{2}$

ii. Þar sem við vitum ekki hvort stökin í fylkinu eru A eða B þá tekur það N^2 langan tíma að sortera þetta fylki.

Verkefni 4

[Röðun] Við getum skilgreint röðunarreikniritið Slembiröðun, sem virkar þannig að á meðan fylkið er ekki raðað þá veljum við tvo vísa i og j af handahófi (á milli 0 og N-1). Ef stök a[i] og a[j] eru í rangri röð í fylkinu þá víxlum við á þeim og höldum áfram. Forritið þetta reiknirit í Java (þið getið notað Selection.java sem fyrirmynd). Takið tímann á keyrslu á 1kints.txt. Keyrið forritið ykkar a.m.k. 5 sinnum og skoðið breytileikann í tímanum. Skilið Java kóðanum fyrir fallið og tímunum á keyrslunum.

Lausn:

```
import edu.princeton.cs.algs4.*;
import java.util.Random;

public class Verkefni {


    public static int uniform(int min, int max){
                Random r = new Random();
                return r.nextInt((max - min) + 1) + min;
    }

    public static void sort(int[] a){
        int n = a.length;
        while(!isSorted(a)){
            int z = uniform(0,n-1);
            int b = uniform(z,n-1);
            if (a[z]>a[b]){
                int temp = a[z];
                a[z] = a[b];
                a[b] = temp;
            }
        }
    }

    public static boolean isSorted(int[] b){
        for (int i = 1; i<b.length; i++){
            if (b[i]<b[i-1]) return false;
        }
        return true;
    }

    public static void main(String[] args) {
```

```
        In in = new In("1Kints.txt");
        int[] a = in.readAllInts();
        Stopwatch timer = new Stopwatch();
        sort(a);
        double eTime = timer.elapsedTime();
        StdOut.println("Elapsed time = " + eTime);

    }
}
```

```
magnusde93@tg-dw098
Elapsed time = 0.37
magnusde93@tg-dw098
Elapsed time = 0.423
magnusde93@tg-dw098
Elapsed time = 0.439
magnusde93@tg-dw098
Elapsed time = 0.723
magnusde93@tg-dw098
Elapsed time = 0.423
magnusde93@tg-dw098
```

Verkefni 5

[Shell röðun] Nota á Shell röðun með 3x+1 skrefstærðum á 10-staka fylki. Þá eru tvær skrefstærðir: 1 og 4 (reyndar er byrjað með h=4).

a) Hvert er besta inntak fyrir þessa tegund af Shellröðun? Rökstyðjið og sýnið heildarfjölda samanburða fyrir 10-staka fylki.

b) Sýnið hvernig þessi Shell röðun virkar á 10-staka fylki í öfugri röð (t.d. [10, 9, ..., 2, 1]). Sýnið fylkið eftir hvora umferð og fjölda samanburða. Berið fjölda samanburða hér saman við fjölda samanburða sem Innsetningaröðun myndi nota á þessu fylki.

Lausn:

a) það væri best að vera með fylkið [0,1,2,3,4,5,6,7,8,9]. Í fyrstu umferð eru 0, 4 og 8 borin saman. Öll eru á réttum stað. Næst er 1,5 og 9 borin saman, aftur öll á sínum stað. Næst er 2 og 6 borið saman. Síðast er 3 borið saman við 7. Þetta eru alls 6 samanburðir. Næst er farið í gegnum öll stök og borið saman við stakið á undan. Það eru 9 samanburðir. Þess vegna eru þetta 15 samanburðir.

b) $[10,9,8,7,6,5,4,3,2,1]$
$[2,9,8,7,6,5,4,3,10,1]$
$[2,1,8,7,6,5,4,3,10,9]$
$[2,1,4,7,6,5,8,3,10,9]$
$[2,1,4,3,6,5,8,7,10,9]$
$[1,2,3,4,5,6,7,8,9,10]$
Þetta eru alls 21 samanburðir.