HEIMADÆMI 3

TÖL203G Tölvunarfræði 2

Kári Hlynsson[1]

Háskóli Íslands

9. febrúar 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 handahófi 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 úr `StdRandom`. Síðan prentar forritið út k fyrstu stökin í fylkinu. Skilið kóðanum fyrir forritið og skjáskoti af keyrslu.

Lausn

Hluti (a)

Útfærsluna má sjá í Forriti 1 fyrir neðan.

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

FORRIT 1: Útfærsla á `toString()` í `Card` klasanum

[1]Slóð á Github kóða: `https://github.com/lvthnn/TOL203G/tree/master/HD4`

Hluti (b)

Útfærsluna má sjá fyrir neðan í FORRITI 2.

```java
import edu.princeton.cs.algs4.Insertion;
import edu.princeton.cs.algs4.StdRandom;

public class CardDeal {

  /**
   * Generates sorted array of playing cards.
   *
   * @return a deck of playing cards
   */
  public static Card[] generate_deck() {
    Card[] deck = new Card[52];
    for (int i = 0; i < 52; i++)
      deck[i] = new Card(i % 4, i % 13 + 1);
    Insertion.sort(deck);

    return deck;
  }

  /**
   * Gathers uniform k-sized sample from deck
   *
   * @param k number of cards in selection
   * @return  a random selection of k cards from the deck
   */
  public static Card[] deal_cards(int k, Card[] deck) {
    Card[] sample = new Card[k];
    StdRandom.shuffle(deck);

    for (int i = 0; i < k; i++) sample[i] = deck[i];

    return sample;
  }

  public static void main(String[] args) {
    int num_cards = Integer.parseInt(args[0]);
    Card[] deck = generate_deck();

```

```java
    StdRandom.shuffle(deck);
    Card[] sample = deal_cards(num_cards, deck);
    for (Card card : sample) System.out.println(card.toString());

  }

}
```

FORRIT 2: Útfærsla á `CardDeal` klasanum

MYND 1 sýnir síðan keyrslu í skel fyrir neðan.

```
*[master][~/Github/TOL203G/HD4/src/V1]$ java CardDeal 3
C-2
C-4
H-8
*[master][~/Github/TOL203G/HD4/src/V1]$ java CardDeal 7
C-J
D-3
S-5
S-Q
C-6
D-9
D-8
*[master][~/Github/TOL203G/HD4/src/V1]$ java CardDeal 15
C-J
D-Q
D-6
S-3
H-9
C-3
S-6
C-10
C-Q
S-10
H-J
C-7
C-6
C-K
C-4
*[master][~/Github/TOL203G/HD4/src/V1]$


[0] 0:nvim- 1:zsh*                                          "Karis-MacBook-Air-2.l" 16:33 09-Feb-23
```

MYND 1: Keyrsla á `CardDeal` í skel fyrir inntök $3, 7, 15$

Verkefni 2

Í útfærslunni á Valröðun í bókinni 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 hvora útgáfu a.m.k. þrisvar sinnum og taka meðaltalið, því það er alltaf einhver breytileiki í keyrslutímanum. Skilið breytta fallinu og niðurstöðum tímamælinganna.

Verkefni 3

Þetta er spurning um hegðun valröðunar og innsetningarröðunar á tilteknu N-staka inntaksfylki:

(a) Öll stökin í fylkinu hafa sama gildi:

  (i) Hversu marga samanburði notar valröðun?

  (ii) Hversu marga samanburði notar innsetningarröðun?

(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?

  (ii) Hversu marga samanburði notar innsetningarröðun?

Lausn

Hluti (a)

(i) Valröðun notar alltaf sama fjölda samanburða því það ítrar frá vísinum `i` og út í enda fylkisins. Fjöldi samanburða er því sem áður $(N-1)+(N-2)+\cdots+2+1 \sim N^2/2$.

(ii) Innsetningarröðun stöðvar alltaf ef `a[j]` $\leq$ `a[j - 1]`, svo í þessu tilviki myndum við alltaf bara líta eitt stak aftur fyrir okkur í röðun á fylkinu. Því er heildarfjöldi samanburða $N-1 \sim N$.

Hluti (b)

(i) Eins og áður sagði er valröðun ekki háð inntaki og því höfum við aftur fjölda samanburða $(N-1)+(N-2)+\cdots+2+1 \sim N^2/2$.

(ii) Látum n og m tákna fjöldann af gildunum A og B í fylkinu, í þeirri röð. Þá er $N = n+m$ heildarföldi staka í fylkinu. Án skerðingar á víðgildni athugum við að ef $n \gg m$ þá erum við nokkurn veginn að fást við (ii) í (a)-lið. Ef svo er ekki og $a \approx m$ þá er fjöldi samanburða áður en stak finnur sinn stað í raðaða fylkinu $\frac{1}{2}(N-i)$. Við fáum því að heildarfjöldi samanburða er $\frac{1}{2}(N-1)+\frac{1}{2}(N-2)+\cdots+1+\frac{1}{2} \sim N^2/4$.

Verkefni 4

Við getum skilgreint röðunarreikniritið *Slembiröðun*, sem virkar þannig að á meðan fylkið er ekki raðað þá veljum við tvo vísa `i` og `h` af handahófi (á milli 0 og $N-1$). Ef stök `a[i]` og `a[j]` eru í langri röð í fylkinu þá víxlum við á þeim og höldum áfram. Forritið þetta reikniriti 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 á tímanum. Skilið Java kóðanum fyrir fallið og tímunum á keyrslunum.

Verkefni 5

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 innsetningarröðun myndi nota á þessu fylki.

Lausn

Hluti (a)

Besta tilvikið kemur upp þegar fylkið er fyrirfram raðað og innri lykkjan keyrir ekki. Kostnaðurinn er einfaldlega að kíkja í gegnum allar hlutrunurnar svo kostnaðurinn er $\Theta(N)$.

Hluti (b)

Setjum upp rakningar af hlutröðunum í TÖFLU 2 og TÖFLU 1 hér fyrir neðan.

	10	9	8	7	6	5	4	3	2	1
$h=4$	10				6				2	
		9				5				1
	2	1	8	7	6	5	4	3	10	9

TAFLA 1: Fyrsta h-röðunin með $h=4$.

Í 4-röðuninni (TAFLA 1 er fjöldi samanburða 6 því við röðum tveimur hlutrunum sem kosta hver um sig 3 samanburði. Næst kemur 1-röðunin í TÖFLU 2 fyrir neðan sem er venjuleg innsetningarröðun, en í henni eru framkvæmdir 25 samanburðir. Til að sjá betri útskýringu, athuga TÖFLU 3.

	2	1	8	7	6	5	4	3	10	9
$h=1$	2	1	8	7	6	5	4	3	10	9
	1	2	3	4	5	6	7	8	9	10

TAFLA 2: Seinni röðunin með $h=1$.

		a[]										
i	j	0	1	2	3	4	5	6	7	8	9	Samanburðir, S
		2	1	8	7	6	5	4	3	10	9	
1	0	1	2	8	7	6	5	4	3	10	9	1
2	2	1	2	8	7	6	5	4	3	10	9	1
3	2	1	2	7	8	6	5	4	3	10	9	2
4	2	1	2	6	7	8	5	4	3	10	9	3
5	2	1	2	5	6	7	8	4	3	10	9	4
6	2	1	2	4	5	6	7	8	3	10	9	5
7	2	1	2	3	4	5	6	7	8	10	9	6
8	8	1	2	3	4	5	6	7	8	10	9	1
9	8	1	2	3	4	5	6	7	8	9	10	2
												$\sum S = 25$

TAFLA 3: Rakning af 1-röðun í Shell röðun með $N = 10$

Prófum nú að bera þetta saman við fjölda samanburða ef við myndum keyra innsetningarröðun á upprunalega fylkið sem gefið var í dæminu:

		a[]										
i	j	0	1	2	3	4	5	6	7	8	9	Samanburðir, S
		10	9	8	7	6	5	4	3	2	1	
1	0	9	10	8	7	6	5	4	3	2	1	1
2	0	8	9	10	7	6	5	4	3	2	1	2
3	0	7	8	9	10	6	5	4	3	2	1	3
4	0	6	7	8	9	10	5	4	3	2	1	4
5	0	5	6	7	8	9	10	4	3	2	1	5
6	0	4	5	6	7	8	9	10	3	2	1	6
7	0	3	4	5	6	7	8	9	10	2	1	7
8	0	2	3	4	5	6	7	8	9	10	1	8
9	0	1	2	3	4	5	6	7	8	9	10	9
												$\sum S = 45$

TAFLA 4: Innsetningarröðun á fylki í öfugri röð með $N = 10$.

Eins og má sjá er sparnaðurinn talsverður við að nota Shell röðun, en það munar 20 samanburðum á röðunarreikniritunum. ■