Vikublað 7 - Tölv 2

ttb3@hi.is

2. mars 2022

3.2.4

a)

a tréð er gefið sem $[10, 9, 8, 7, 6, 5, 4]$

a gengur upp, ef maður fylgir trénu sést að $10 > 9 > 8 > ... > 5$, þannig maður fylgir bara vinstri greinum trésins þangað til að maður lendir á 5

b)

b tréð er gefið sem $[4, 10, 8, 7, 5, 3]$

nú byrjar maður á 4, þar sem $4 < 5$ er eina leiðin til að finna 5 að fara til hægri, það er gert þegar farið er á 10, næsta tékk er $10 > 8$ þannig farið er til vinstri. Eins og er gengur þetta upp, haldið er áfram að fara eftir vinstri hlið trésins þangað til að komið er að 5, næst í röðinni á eftir 5 er 3 en það gengur ekki upp því $3 < 4$ og ætti þessvegna að vera á vinstri grein frá rótinni og ekki fyrir neðan 5. Þá má sjá að b er röð aðferða sem gengur ekki upp.

c)

c tréð er gefið sem $[1, 10, 2, 9, 3, 8, 4, 7, 6, 5]$

hérna skilar röð aðferða tré sem sikksakkar, byrjar á hægri,vinstri,hægri þangað til að komið er að 6 þá fer maður aftur til vinstri og lendir á 5, rétt gildi fundið, allir glaðir :)

d)

d tréð er gefið sem $[2, 7, 3, 8, 4, 5]$

hérna er tré sem ekki gengur upp, það fer frá 2 til hægri í 7 og frá 7 vinstri til 3 en svo hættir gamanið því það ætti að fara frá 3 til hægri á 8 nema hvað að $8 > 7$ þannig það ætti ekki að geta átt heima í vinstra hluttré 7

d virkar ekki

e)

e tréð er gefið sem $[1, 2, 10, 4, 8, 5]$
röð aðferða verður hægri, hægri, vinstri, hægri, vinstri. það gengur upp

3.2.15

ritháttur:
$Y \rightarrow X$ = fara til hægri frá Y og lenda á X
$Y \leftarrow X$ = fara til vinstri frá Y og lenda á X

a)

floor("Q"), leitar að Q og skilar minnsta stakinu sem finnst á leiðinni
$E > Q : E \rightarrow Q$ þ.e. vegna þess að E er stærra en Q, fara frá E til hægri og lenda á Q. þar sem að Q er það sem leitað var að, skilar Q

b)

fer og athugar rank vinstra hluttrésins, það er 2 þannig þá á að leita í hægra hluttrénu að staki með rank $5 - 2 - 1$, tékkað á rank Q það er 2 og þá er skilað Q
$rank(E) < 5 : E \rightarrow Q, rank(Q) == 2$

c)

voða svipað og a), nema skilar stærsta staki á leið þess að finna stak, $E > Q :\rightarrow Q$ skilar Q

d)

$E > J :\rightarrow Q, Q < J :\leftarrow J, rank(j) == 0$ skilar 0

e)

$E < D \leftarrow D$ telur 1 fyrir rót og 1 fyrir það að lágpunktur fundinn, fara aftur á rót $E > Q :\rightarrow Q$, Q talin, $Q \leftarrow J, J \rightarrow M$ J og M talin, aftur í Q rót, $Q \rightarrow T$ T talið, $T \leftarrow S$ því S er minna en T, S talið, hápunktur fundinn

TopFreq

skrifaði main fall sem tekur inn k, lágmarksfjölda stafa orðs, N, fjölda orða sem á að birta og notar scanner til að taka orð inn af staðalinntaki. Það notar Stopwatch til þess að mæla tímann á föllunum og maður velur á milli þeirra með því að commenta út það kall á fall sem ekki á við.

MAIN

```
public static void main(String[] args) {
    int k = Integer.parseInt(args[0]);
    int N = Integer.parseInt(args[1]);

    Scanner s = new Scanner(System.in, "utf-8");

    Stopwatch time = new Stopwatch();
    ArrayImp(s, k, N);
    // HashImp(s, k, N);
    double elapsed = time.elapsedTime();
    System.out.println(elapsed);

    // ahugaverd og einfold utfaersla a thessu ollu:
    // cat tale.txt | tr " " "\n" | sort | uniq -c | sort -nr | head
        -100 | less
}
```

ArrayImp, A)

array útfærslan notast við arraylist til þess að taka inn orð af staðalinntaki, sortar fylkið og ítrar yfir það til telja endurtekningar. það er svo haldið utan um endurtekin orð með því að skrifa orðið inn í eitt fylki og fjölda endurtekninga í annað. ef orð er endurtekið oftar en ehv orð sem er í geymslufylkjunum, er minna orðið yfirskrifað. vegna þess hvernig er farið yfir geymslufylkin endar maður með minnkandi listas af endurteknum orðum.

```
public static void ArrayImp(Scanner s, int k, int N) {
    List<String> words = new ArrayList<String>();
    while (s.hasNext()) {
        String word = s.next();
        if (word.length() >= k) {
            words.add(word);
        }
    }
    s.close();
    String[] wordsArray = new String[words.size()];
    wordsArray = words.toArray(wordsArray);
    Arrays.sort(wordsArray);
    int counter = 1;
    int[] fjoldi = new int[N];
    String[] ord = new String[N];
    for (int i = 1; i < wordsArray.length; i++) {
        if (wordsArray[i].equals(wordsArray[i - 1]))
            counter++;
        else {
            for (int j = 0; j < ord.length; j++) {
                if (fjoldi[j] < counter) {
                    fjoldi[j] = counter;
                    ord[j] = wordsArray[i - 1];
                    break;
                }
            }
            counter = 1;
        }
    }
    for (int i = 0; i < ord.length; i++) {
        StdOut.println(ord[i] + " " + fjoldi[i]);
    }
}
```

HashImp, B)

breytti útfærslunni úr bókinni varla nema til að nota scanner og geta kallað á það úr main

```
public static void HashImp(Scanner s, int k, int N) {
    ST<String, Integer> st = new ST<String, Integer>();
    while (s.hasNext()) {
        String word = s.next();
        if (word.length() < k)
            continue;
        if (!st.contains(word))
            st.put(word, 1);
        else
            st.put(word, st.get(word) + 1);
    }
    for (int i = 0; i < N; i++) {
        String max = "";
        st.put(max, 0);
        for (String word : st.keys())
            if (st.get(word) > st.get(max))
                max = word;
        StdOut.println(max + " " + st.get(max));
        st.delete(max);
    }
}
```

TÍMAR!!

nr. keyrslu	A	B
nr.1	0.159s	0.189s
nr.2	0.147s	0.212s
nr.3	0.165s	0.190s
nr.4	0.147s	0.220s
nr.5	0.163s	0.214s
meðaltal:	0.156s	0.205s

3.2.21

```
public static BST randomKey(BST x) {
    return select(StdRandom.uniform(x.size()));
}
```