HÁSKÓLINN Í REYKJAVÍK
REYKJAVIK UNIVERSITY

Reiknirit/Algorithms
T- 301 – REIR

Lokapróf / *Final Exam*

I	
I I	
I I I	
IV	
V	
VI	
VII	
VIII	
IX	
X	
XI	
XII	
XII	
Total	

Kennari/*Instructor*: Magnús M. Halldórsson

Dagsetning/*Date*: Þriðjudagur 15. nóvember 2016 /
Tuesday, November 15, 2016

Tími/*Time*: 9.00 – 12.00 / *9:00 – 12:00*

Hjálpargögn/*Supporting materials*:
*** Eitt blað (báðum megin. Handritað, eða með 10^{+}pt fonti) /**
Single A4-page (Handwritten or with 10^{+}pt font)
*** Einfaldur vasareiknir / *Simple calculator***
(eða með minni útþurrkað/*or with memory cleared*)

Nafn/*Name*:_____

Kt./*ID*: _____

I. (8%) Tímaflækja / Order-of-growth time complexity

A. Hver er tímaflækja hverra eftirfarandi kóðabúta, sem fall af N? Veldu fall úr töflunni til hægri. Nota má fall oftar en einu sinni. / *What is the time complexity of the following code segments, as function of N? Choose a function from the table on the right; a function can be used more than once.*

(a)

```
int x = 1;
for (int j=N; j > 0; j /= 2)
   for (int i=0; i < N; i++)
      x = x*i;
```

Svar/*Answer*: _____

(b)

```
int sum;
for (int j=0; j < N; j++)
   for (int i=0; i < N*N; i++)
        sum += i;
```

Svar/*Answer*: _____

(c)

```
int x = 0;
for (int i=0; i < N; i++)
  for (int j=0; j < 6; j++)
    x += i;
```

Svar/*Answer*: _____

log N
N
N log N
N^2
N^3
N^2 log N
N^3 log N
2^N
N!

(d)

```
public static void f(int N, int a[]) {
  if (N < 1) return;
  a[N] = 0;
  f(N-1, a);
  a[N] = 1;
  f(N-1, a);
}
```

Svar/*Answer*: _____

B. Hver er tímaflækja eftirfarandi kóðabúts sem fall af V and E? / *What is the time complexity of the following code segment as a function of V and E?*

(d)

```
for (int v = 0; v < G.V(); v++)
  for (int w : G.adj(v))
     StdOut.println(v + "-" + w);
```

Svar/*Answer*: _____

II. (8%) Mælingar

(a) Segjum að þú mælir eftirfarandi keyrslutíma reiknirits, þar sem N er fjöldi staka í inntakinu. Mælingarnar eru gerðar á tölvu sem framkvæmir 10^9 aðgerðir á sekúndu.
/ Suppose that you observe the following running times of an algorithm, where N is the number of items in the input. The measurements are done on a machine that performs 10^9 *operations per second.*

N	1,000	2,000	3,000	4,000
Time	0.1 sek	0.4 sek	0.9 sek	1.5 sek

Þú ákvarðar að gögnin séu í samræmi við tímaflækju (í fjölda framkvæmdra aðgerða) á forminu $\sim aN^b$. Ákvarðaðu gildin á a og b.
/ You determine that the data is consistent with time complexity (in the number of operations performed) of the form $\sim aN^b$. *Determine the values of* a *and* b.

Svar/*Answer*: a = _____ b = _____

(b) Segjum að reiknirit nokkuð með tímaflækjuna $\Theta(2^N)$ sé keyrt á inntak af stærð $N = 40$, og tekur keyrslan 2 sek. Nú er óskað eftir því að keyra reikniritið á inntak af stærð $N = 52$ á sömu tölvu. Hve lengi ætti sú keyrsla að taka? */ An algorithm with time complexity* $\Theta(2^N)$ *is run on an instance of size* $N = 40$, *finishing in 2 seconds. We now desire to run it on an instance of size* $N = 52$. *How long should it to take?*

Svar: _____ ***sek/sec.***

(c) Hve mikið minni tekur hlutur af taginu `Wot`?
/ How much memory does an object of type `Wot` *use?*

```
public class Wot {
  private int x;
  private int y;
  private char ch;
  private Wot next;
}
```

Svar: _____ ***bæti/bytes***

III. (4%) Union-Find

Geta eftirfarandi fylki verið útkoma á því að keyra *Quick-Find(QF), Quick-Union (QU) og/eða Weighted Quick-Union (WQU)*? Dragðu hring um alla möguleika sem eiga við.
/ Can the following arrays be the result of Quick-Find(QF), Quick-Union (QU) and/or Weighted Quick-Union (WQU)? Circle all that apply.

i	0	1	2	3	4	5	6	7	8
id[i]	0	0	0	8	1	1	2	8	8

QF QU WQU

i	0	1	2	3	4	5	6	7	8
id[i]	8	8	8	8	8	8	8	8	8

QF QU WQU

i	0	1	2	3	4	5	6	7	8
id[i]	0	1	0	4	1	1	2	0	0

QF QU WQU

i	0	1	2	3	4	5	6	7	8
id[i]	2	7	2	4	1	2	2	8	4

QF QU WQU

i	0	1	2	3	4	5	6	7	8
id[i]	2	7	2	4	1	2	2	8	4

QF QU WQU

IV. (8%) Hrúgur / *Heaps*

Skoðið eftirfarandi max-hrúgu. / *Consider the max-heap given given below.*

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
--	77	23	47	17	20	40	38	8	12	7	11	25	13	6	30	2

(a) Framkvæmdu „eyða stærsta stakinu“, `delMax()`, aðgerðina á hrúguna. Sýndu hrúguna sem kemur út í meðfylgjandi töflu. / *Show the resulting heap in the following table after performing a delete-the-maximum,* `delMax()`, *operation.*

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
--																

(b) Dragið hring um þau stök sem færast til í hrúgunni þegar `insert(88)` er framkvæmt á upphaflegu hrúguna / *on the original heap*. Circle the elements that change location in the heap after `insert(88)` is performed *on the original heap*.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
-	77	23	47	17	20	40	38	8	12	7	11	25	13	6	30	2

V. (10%) Röðunarreiknirit/ Sorting algorithms

Í töflunni að neðan er dálkurinn lengst til vinstri inntak af strengjum og dálkurinn lengst til hægri strengirnir raðaðir. Hver af hinum átta dálkunum er staðan í miðri keyrslu á einu af röðunarreikniritunum (númer 2-9) listuð til hægri. Parið saman dálkana og reikniritin, og notið hverja tölu einu sinni.

/ The leftmost column in the table below is the original input of strings to be sorted, and the rightmost column consists of the strings in sorted order. The other eight columns are the contents at some intermediate step during one of the sorting algorithms (number 2-9) listed on the right. Match each algorithm by writing its number under the corresponding column. Use each number exactly once.

```
 0 hopp blak blak blak anda varp anda blak brun anda
 1 gang slak brun gang blak takt blak brun gang blak
 2 blak anda drif hopp brun taka brun drif blak brun
 3 skor lyft gang kast drif skor drif gang anda drif
 4 varp svif hopp renn gang lyft gang hopp hnit gang
 5 svif drif kast skor geim svif geim kast geim geim
 6 kast klif klif svif hnit svig hopp rauk drif hnit
 7 renn svig rauk varp hopp slak hnit renn hopp hopp
 8 rauk leik renn brun kast rauk kast skor rauk kast
 9 brun stik skor drif keil hnit klif svif renn keil
10 drif spil spil klif klif gang keil taka kast klif
11 taka keil svig rauk leik stik leik varp taka leik
12 spil geim svif spil spil spil lyft klif spil lyft
13 svig hnit taka svig svig kast renn leik svig rauk
14 klif taka takt taka skor klif rauk slak klif renn
15 takt takt varp takt takt renn skor spil takt slak
16 slak gang slak anda slak hopp svif svig slak skor
17 leik renn leik geim taka leik spil takt leik spil
18 keil skor keil hnit varp keil svig anda keil stik
19 geim hopp geim keil svif geim slak geim svif svig
20 hnit varp hnit leik rauk brun stik hnit varp svif
21 anda kast anda lyft renn anda taka keil skor taka
22 lyft rauk lyft slak lyft drif takt lyft lyft takt
23 stik brun stik stik stik blak varp stik stik varp

   0                                          1
---- ---- ---- ---- ---- ---- ---- ---- ---- ----
```

0. Upphaflegt inntak / Original input
1. Raðað / Sorted
2. Selection sort
3. Insertion sort
4. Mergesort (ofans*ækið/top-down*)
5. Mergesort (*neðansækið/bottom-up*)
6. Quicksort (standard, no shuffle)
7. Heapsort
8. LSD radix sort
9. MSD radix sort

VI. (8%) Tætitöflur / *Hashing*

Eftirfarandi lyklar eru settir inn í tóma tætitöflu af stærð sjö (án ‚resizing') , í einhverri röð. Notuð er línuleg könnun (linear probing), með hakkagildum sem sýnd eru hér til hliðar. / *Suppose the following keys are inserted in some order into an initially empty linear-probing hash table of size 7 (assuming no resizing), using the following table of hash values.*

Key	***Hash***
A	6
B	2
C	1
D	6
E	1
F	5
G	0

(a) Hvert af eftirtöldu gæti verið innhald tætifylkisins ef stökin að ofan eru sett inn í einhverri röð? *Which of the following could be the contents of the linear-probing array if the keys are inserted in some order?*

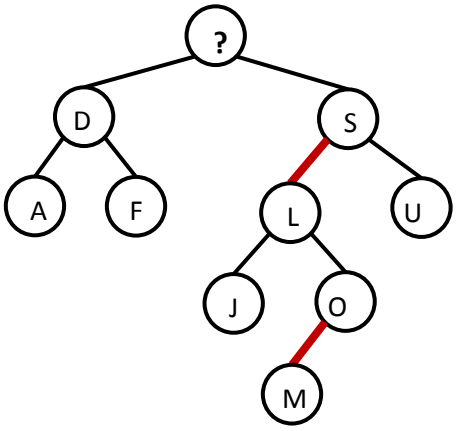
		Mögulegt ***Can occur***	**Ómögulegt** ***Cannot occur***
(i)	<table><tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td></tr><tr><td>A</td><td>D</td><td>E</td><td>C</td><td>G</td><td>F</td><td>B</td></tr></table>	☐	☐
(ii)	<table><tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td></tr><tr><td>G</td><td>C</td><td>E</td><td>B</td><td>A</td><td>F</td><td>D</td></tr></table>	☐	☐
(iii)	<table><tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td></tr><tr><td>B</td><td>C</td><td>D</td><td>E</td><td>G</td><td>F</td><td>A</td></tr></table>	☐	☐

(b) Hverjir eru kostir tætitafla fram yfir rauð-svört tré? / *Which of these are advantages of Hashing over Red-Black trees?*

	Satt/*True*	**Ósatt/*False***
(i) Tæting er viðkvæm fyrir „denial-of-service" árásum. / *Hashing is susceptible to denial-of-service attacks*	☐	☐
(ii) Tæting leyfir raðaðar aðgerðir eins og `range()`,`floor()`,`size()` / *Hashing allows for operations like* `range()`,`floor()`,`size()`.	☐	☐
(iii) Tæting hefur betri versta-falls tímaflækju / *Hashing has better worst-case performance*	☐	☐
(iv) Tæting hefur betri meðaltals-tímaflækju / *Hashing has better average-case time complexity*	☐	☐

VII. (9%) Rauð-svört tré / *Red-black trees*

I. Hér til hliðar er vinstri-vísandi rautt-svart tré, þar sem feitletraðar línur tákna rauða leggi.
/ Below is a left-leaning red-black tree, where bold lines indicate red edges.



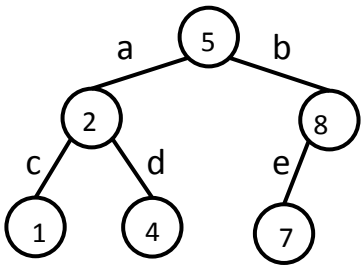
(a) Which of the keys below could be the one labeled with a question mark. Circle all possibilities.

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

(b) Hve marga vinstri snúninga, hægri snúninga, og litavíxl eru notuð til að setja inn hvert af eftirtöldum lykklum inn í upphaflega vinstri-vísandi rauð-svarta tréð að ofan? */How many left rotation, right rotation, and color flip operations would be used to insert each key below into the original LLRB BST above?*

	K	E	B	N
rotateLeft()	1			
rotateRight()	0			
flipColors()	0			

II. Skoðum nú vinstri-vísandi rauð-svarta tréð að neðan*. / Now consider the left-leaning red-black tree below.*

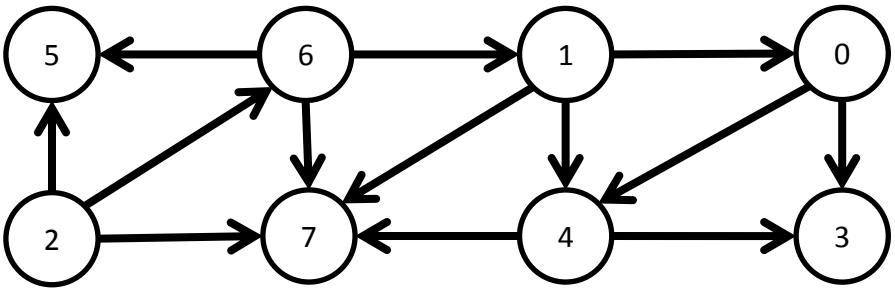


(c) Dragðu hring utan um þá stafina að neðan sem samsvara leggjum sem hljóta að vera rauðir. Krossaðu yfir þá stafi sem samsvara leggjum sem geta ekki verið rauðir. */ Circle the letters of the edges that must be red. Cross out the letters of edges that cannot be red.*

a b c d e

VIII. (10%) Áttuð net/ *Digraphs*

Að neðan er áttað net `G`. Gerðu ráð fyrir að grannlistarnir séu í röð eftir númerum, t.d. grannar nóðu 6 eru í réttri röð: 1, 5, 7. / *Consider the following digraph* `G`*. Assume that the adjacency lists are in order, e.g., the neighbors of node 6 are in the order 1, 5, 7.*



(a) Hver er DFS *postorder* rakning á netinu? / *Give the DFS postorder traversal of the digraph*

Svar:

(b) Hver er DFS *preorder* rakning á netinu? / *Give the DFS preorder traversal of the digraph*

Svar:

(c) Hvaða grannröðun (topsort) finnur reiknirit bókarinnar? /
Give the topological sort (topsort) that the textbook's algorithm constructs.

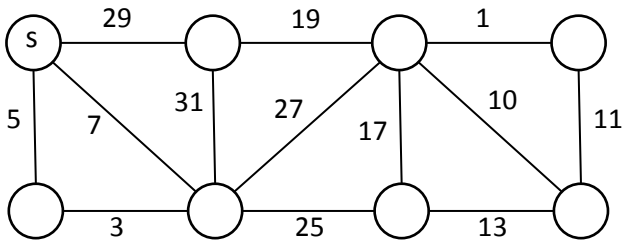
Svar:

(d) Eftir að `BFS(G,0)`, er keyrt (upphafspunkt 0), hvaða gildi eru í fylkjunum `distTo[]` og `edgeTo[]`? / *After we run* `BFS(G,5)`*, with a starting node 0, what values are in the* `distTo[]` og `edgeTo[]` *arrays?*

Svar:

`I`	0	1	2	3	4	5	6	7
`edgeTo[i]`								
`distTo[i]`								

IX. (8%) Minnstu spanntré / *Minimum spanning trees*



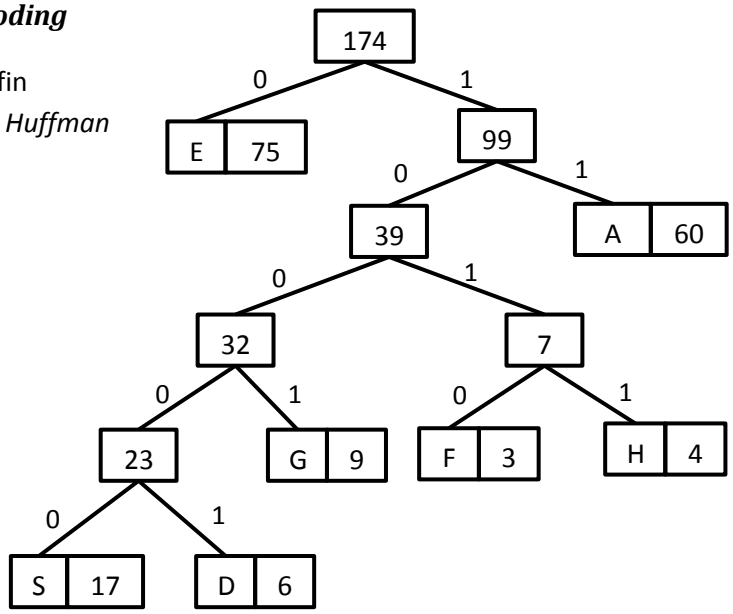
(a) Teldu upp 6 fyrstu leggina sem aðferð Kruskal velur í minnsta spanntré. / *List the first 6 edges that Kruskal's minimum spanning tree algorithm selects.*

(b) Teldu upp 6 fyrstu leggina sem aðferð Prim velur í minnsta spanntré. / *List the first 6 edges that Prim's minimum spanning tree algorithm selects.*

(c) Hvaða staðhæfingar eru sannar? / Which of these claims are true?	**Satt/*True***	**Ósatt/*False***
(i) Ef vægin á leggjunum eru öll mismunandi, þá er minnsta spanntréð ávallt einkvæmt. / *If the weights are distinct, then the minimum spanning tree is always unique*	☐	☐
(ii) Minnsta spanntréð breytist ekki þó að öll vægi séu tvöfölduð. / *The minimum spanning tree always stays unchanged if all the edge weights are doubled.*	☐	☐
(iii) Leggurinn út frá hnúti 0 með minnsta vægið verður að vera í minnsta spanntrénu. / *The minimum weight edge incident on node 0 must be in the spanning tree.*	☐	☐

X. (8%) Huffman kóðun/*Huffman coding*

Skoðum þetta Huffman tré, þar sem laufin geyma staf og tíðni hans. / *Consider this Huffman tree, where leaves store a letter and its frequency.*



(a) Hvernig er þessi bitastrengurinn afkóðaður? */ What is the decoding of this bitstring:*

`100000011101101110001`

Svar/Answer: _____

(b) Hvað er besta og hvað er versta þjöppunarhlutfall sem Huffman tréð að ofan getur náð á strengi sem innihalda einungis stafina A, E, S? / *What is the best and the worst compression ratio achieved by the Huffman tree above on strings containing only the letters A, E and S?*

***Besta/Best**:* _____ ***Versta/Worst**:* _____

(c) Dragðu hring um þá stafi sem yrðu kóðaðir á með lengri eða styttri kóðaorðum en nú þegar notað er besta mögulega forskeytilausa kóðunartré (miðað við þær stafatíðnir sem sýndar eru). / *Circle the letters that would encoded with longer or shorter codewords than now in an optimal prefix-free coding tree (given the frequencies shown in the figure).*

A D E F G H S

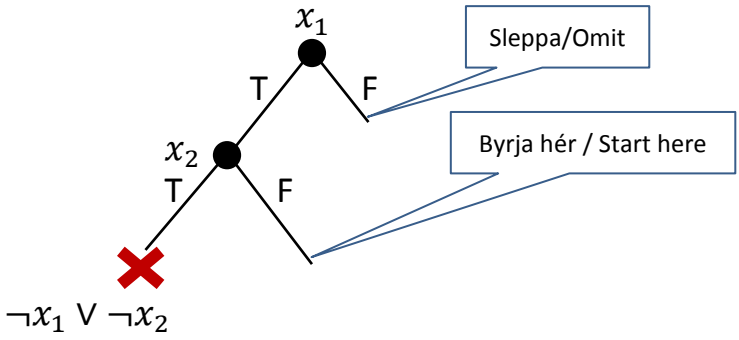
XI. (8%) Satisfiability

Gefnar eru eftirfarandi rökklausur: / *We are given the following propositional clauses:*

$x_1 \vee x_2 \vee x_5$ $\quad \neg x_1 \vee \neg x_2$ $\quad x_1 \vee x_3$ $\quad \neg x_1 \vee \neg x_3$

$x_1 \vee x_2 \vee \neg x_5$ $\quad x_2 \vee \neg x_4$ $\quad \neg x_2 \vee x_4$ $\quad x_3 \vee x_4$ $\quad \neg x_3 \vee \neg x_4$

(a) (5%) Ljúktu við vinstri hliðina á eftirfarandi leitartré sem myndast þegar *backtracking* aðferðinni fyrir Satisfiability er beitt á klaususafnið að ofan. Merktu lauf með þeirri klausu sem myndar mótsögn. / *Complete the left side of the evaluation tree below formed by the backtracking algorithm for Satisfiability. Label each leaf with the clause that forms a contradiction.*

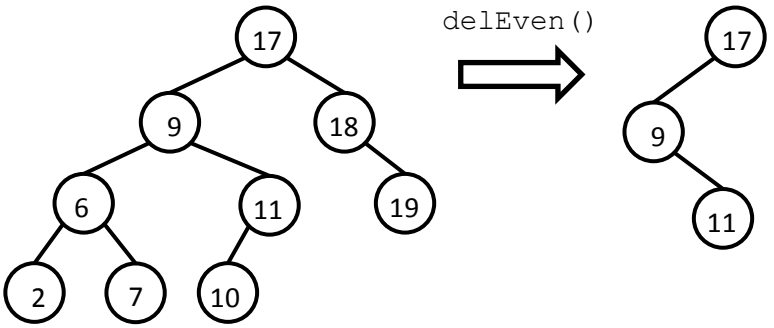


(b) (3%) Teiknaðu nú leitartréð sem myndast þegar „unit propagation“ (DPLL) er beitt með backtracking aðferðinni. Sýndu á hverjum legg þær eindaklausur (unit clauses) sem leiddar eru út, ásamt því að merkja lauf með mótsagnarklausum á sama hátt og áður. / *Now give the evaluation tree formed when applying unit propagation (DPLL) in the backtracking algorithm. Label each edge with the unit clauses derived, as well as marking the leaves with contradicting clauses as before.*

XII. (9%) Implementing Data Structures

Hér fyrir neðan er einfaldað API fyrir tvíleitartré af heiltölum. Útfærðu (in Java) aðferðina `delEven()` sem eyðir úr trénu öllum nóðum sem hafa gildi sem er jöfn tala, og öllum undirtrjám þeirra nóða. Skilgreina má hjálparaðferðir. */ To the right is an API of a simplified binary search tree of integers. Implement in Java the method* `delEven()` *that deletes from the tree all nodes, and their subtrees, whose values are even integers. You may define private methods.*

Dæmi/*Example*:




```
public void delEven() {
```

API:

```
public class BST {

private Node root;

private class Node {
  private int key;
  private Node left, right;
  public Node(int k,
              Node l, Node r);
}

public void delEven();
}
```

(b) (2%) Hver er tímaflækja aðferðar þinnar (big-theta)? / **Svar**: _____

What is the big-theta time complexity of your method?

XIII. (6%) Tímaflækja (Satt-Ósatt/True-False)

Skor fyrir hvorn lið er fjöldi réttra svara fram yfir fyrstu 2. / *The score for each part equals the number of correct answers beyond the first 2.*

(a)

	Satt/True	Ósatt/False
♦ Ef til er reiknirit með margliðu tímaflækju fyrir Satisfiability, þá er $P \neq NP$. *If Satisfiability is polynomial-time solvable, then $P \neq NP$.*	☐	☐
♦ Það er NP-fullkomið verkefni að ákvarða hvort gefið áttað net innihaldi áttaða rás. *Detecting if a directed graph contains a (directed) cycle is an NP-complete problem.*	☐	☐
♦ KMP strengjaleitaraðferðin er hraðvirkari en Boyer-Moore á slembistrengi, en getur þurft að bakka í inntakinu. / *The KMP substring search is faster than Boyer-Moore on random data, but may need to back up in the input.*	☐	☐
♦ Röðunarreiknirit sem byggir á `compareTo()` getur ekki haft $O(N)$ tímaflækju. */ No algorithm that sorts using* `compareTo()` *can have $O(N)$ time complexity*.	☐	☐
♦Það krefst minni tímaflækju að finna miðgildi en að raða. / *Finding the median requires less time complexity than sorting.*	☐	☐

(b) Ýmsar staðhæfingar / *Miscellaneous statement*s.	Satt/True	Ósatt/False
♦ Ef raða þarf fyrst eftir einum lykli og svo eftir öðrum, þá er betra að nota Quicksort en Mergesort. / *If you need to sort first by one key and then by another key, then it is better to use Quicksort than Mergesort.*	☐	☐
♦ DFS og BFS eru jafngild og nýtast á sama hátt við helstu leitarverkefni í óáttuðum netum. / *DFS and BFS are interchangeable in solving the main search problems in graphs.*	☐	☐
♦ Lögun á træ (*trie*) fer ekki bara eftir því hvaða stök eru sett í það heldur líka í hvaða röð það er gert. / *The shape of a trie depends not only on which elements are inserted but also on the order in which they are inserted.*	☐	☐
♦ Burrows-Wheeler umritun á streng samanstendur af heiltölu og runu af stöfum úr strengnum. / *The Burrows-Wheeler transform of a string consists of an integer followed by a sequence of characters from the string.*	☐	☐
♦ Þegar N stökum er ýtt á stafla sem útfærður er með stækkanlegu (*resizable*) fylki, þá er tímaflækja á hverja *push* aðgerð fasti. / *When pushing N elements onto a* `Stack` *implemented as a resizable array, the time complexity of each push operation is a constant.*	☐	☐