Algorithms and Data Structures II.
Written Exam – Example Questions 2022

Rules for the Exams

Dear Student,

You get theoretical questions (definitions, theorems, time complexity calculations), and some other questions about illustrating algorithms learned this semester. Most of these algorithms have different variants. You must present them as you have seen them in the classroom.

Write the structograms, definitions, and theorems, and illustrate each algorithm the way you learned in my lessons. You have patterns for illustration in Teams and Canvas. The time complexity calculations of the algorithms must be given in detail.

For each exercise about illustrating algorithms follow the following process:

- Carefully copy the input data given in the exercises.
- You must use the data given in your exercise.
- Illustration on another input will not be accepted.
- You draw your illustrations on some sheets of originally blank paper.

Best wishes, Tibor Asvanyi

DC Data Compression

DC.1 The Huffman coding

`<http://en.wikipedia.org/wiki/Huffman_coding>`

DC.1.1 We would like to compress the text

MATEKFELELETEMKETTESLETT

with Huffman coding. Draw the Huffman tree, give the Huffman code of each letter, the Huffman code of the text, and the length of the Huffman code in bits. Show the letters of the original text in the Huffman code of it.

DC.1.2 Solve Exercise DC.1.1 with the following text.

EMESE MAI SMSE NEM NAIV MESE

DC.1.3 We have compressed a text with Huffman coding. Given an internal node of the Huffman tree, number 0 labels the edge which goes to its child with a smaller frequency value, and number 1 labels the edge which goes to its child with a higher frequency value. The frequencies of the different characters are the following ones. R:2, T:4, O:5, G:7, E:13, and A:17. The beginning of the compressed code is

0111 1100 0100 1101 0

without spaces. (Blanks are included here only to increase readability.) Draw the Huffman tree and decompress the code above. What are the codes of the characters above? What is the decompressed text?

DC.2 The Lempel–Ziv–Welch (LZW) algorithm

`<http://en.wikipedia.org/wiki/Lempel-Ziv-Welch>`

DC.2.1 We have compressed a text with the Lempel-Ziv-Welch algorithm. The text contains the letters A', 'B', and 'C'. Their codes are 1, 2, and 3 in turn. While building the dictionary, for the code of each new word the first unused positive integer was selected. In this way, we have received the following code.

1 2 4 3 5 6 9 7 1

Give the original text and the complete dictionary.

DC.2.2 Solve Exercise DC.2.1 with the following LZW code.

1 2 4 3 5 8 1 10 11 1

DC.2.3 Illustrate LZW compression on text $CBABABABCBABABAB$.

DC.2.4 Solve Exercise DC.2.1 with the following LZW code.

1 2 3 4 6 5 9 7 11

AVL trees

Note: { [(1) 2] 3 [(4 {5}) 6 (7)] } is the parenthesized form of the following tree.

```
    3
   / \
  2   6
 /   / \
1   4   7
     \
      5
```

AVL.1 Given an initially empty AVL tree t. Illustrate the insertion of numbers 1 2 7 3 5 6 8 9 4 into it in the given order. Show the balances of each node on your illustrations. When you balance a subtree, make sure, which subtree is balanced.

AVL.2 Given the AVL tree $t = \{\ [\ (1)\ 2\ ]\ 3\ [\ (\ 7\ \{9\}\ )\ 10\ (11)\ ]\ \}$. Draw the tree with its balances. Illustrate the deletion of numbers 2 1 7 from t in the given order. Show the balances of each node on your illustrations. When you balance a subtree, make sure, which subtree is balanced.

AVL.3 Given the AVL tree $t = \{\ [\ (1)\ 2\ ]\ 3\ [\ (\ 7\ \{9\}\ )\ 10\ (11)\ ]\ \}$. Draw the tree with its balances. Illustrate the removal of its minimal and maximal key, **in each case on the original tree**. Show the balances of each node on your illustrations. When you balance a subtree, make sure, which subtree is balanced.

AVL.4 Given the AVL tree $t = \{\ [\ (2)\ 4\ (\ \{6\}\ 8\ \{10\}\ )\ ]\ 12\ [\ 14\ (16)\ ]\ \}$. Draw the tree with its balances. Illustrate the insertion of keys 1, 5, and 15, **in each case on the original tree**. Show the balances of each node on your illustrations. When you balance a subtree, make sure, which subtree is balanced.

AVL.5 Given the AVL tree $t = \{\ [\ (2)\ 4\ ]\ 6\ [\ (\ \{8\}\ 10\ \{12\}\ )\ 14\ (16)\ ]\ \}$. Draw the tree with its balances. Illustrate the removal of its minimal and maximal key, **in each case on the original tree**. Show the balances of each node on your illustrations. When you balance a subtree, make sure, which subtree is balanced.

AVL.6 We already know BSTs. Present the definitions characterizing AVL trees.

AVL.7 Define Fibonacci trees. What do you know about the size of a Fibonacci tree with height h? Why Fibonacci trees are important?

AVL.8 What theorem do you know about the height of a balanced binary search tree? Why this theorem is important?

AVL.9 What theorem do you know about the height of a balanced binary search tree? Present the outline of its proof.

AVL.10 What kind of balancing rotations do you know? Draw the four general rotation schemes which are important during insertion.

AVL.11 Draw the two general rotation schemes which are not important during insertion. Why are these important?

AVL.12 Draw the structogram of procedure AVLinsert() as we learned in the lecture. (Do not draw the structograms of its subroutines.)

AVL.13 Draw the structogram of procedure leftSubTreeGrown() as we learned in the lecture. (Do not draw the structograms of its subroutines.)

AVL.14 Draw the structogram of procedure rightSubTreeGrown() as we learned in the lecture. (Do not draw the structograms of its subroutines.)

AVL.15 Draw the structogram of balancing rotation $(++,+)$ as we learned in the lecture.

AVL.16 Draw the structogram of balancing rotation $(++,-)$ as we learned in the lecture.

AVL.17 Draw the structogram of balancing rotation $(++,=)$ as we learned in the lecture.

AVL.18 Draw the structogram of balancing rotation $(--,+)$ as we learned in the lecture.

AVL.19 Draw the structogram of balancing rotation $(--,-)$ as we learned in the lecture.

AVL.20 Draw the structogram of balancing rotation $(--,=)$.

AVL.21 Draw the structograms of procedures AVLremMin() and leftSubTreeShrunk() as we learned in the lecture. (Do not draw the structograms of their other subroutines.)

AVL.22 Draw the structogram of procedure rotate_PP() as we learned in the lecture. (Do not draw the structograms of its subroutines.)

AVL.23 Draw the structogram of procedure AVLdel() as we learned in the lecture. (Do not draw the structograms of its subroutines.)

AVL.23 Draw the structograms of procedures AVLdelRoot() and rightSubTreeMinToRoot() as we learned in the lecture. (Do not draw the structograms of their other subroutines.)

B+ trees

Note for the subsequent exercises:
– The general schemes of the textual representation of a non-leaf B+ tree of degree 4 are
(T1 k1 T2), **(T1 k1 T2 k2 T3)**, and **(T1 k1 T2 k2 T3 k3 T4)** where **Ti** is the ith subtree, and **ki** is a split key between the ith and the (i+1) subtree.
- The general schemes of the textual representation of a leaf of a B+ tree of degree 4 are **(k1 k2)**, and **(k1 k2 k3)** where **ki** is a key of the set represented with the B+ tree.
- In the example trees we vary three kinds of pairs of brackets: (), [], and {} depending on the level of a subtree.

B+1 Given the following B+ tree of degree 4.
{ [(1 2 4) 5 (6 8)] 10 [(10 12 14) 15 (16 18) 20 (20 22) 24 (24 26)] }
Draw the tree. Illustrate the insertion of keys 5, 3, and 11, **in each case on the original tree**.

B+2 Given the following B+ tree of degree 4.
{ [(2 4) 5 (6 8)] 10 [(10 12 14) 15 (16 18) 20 (20 22)] }
Draw the tree. Illustrate the deletion of keys 18, 22, and 2, **in each case on the original tree**.

B+3 Given the following B+ tree of degree 4.
{ [(2 3 4) 5 (6 7)] 16 [(16 18) 19 (20 22)] }
Draw the tree. Illustrate the insertion of key 1 and the deletion of key 16, **in each case on the original tree**.

B+4 Given the following B+ tree of degree 4.
[(9 10) 11 (12 13 14) 15 (15 16) 18 (19 20)]
Draw the tree. Illustrate the insertion of key 11, then the deletion of keys 9 and 19, **in each case on the original tree**.

B+5 Given the following B+ tree of degree 4.
[(9 10) 11 (12 13) 14 (15 16 17) 18 (19 20)]
Draw the tree. Illustrate the insertion of key 14, then the deletion of keys 10 and 20, **in each case on the original tree**.

B+6 Given the following B+ tree of degree 4.
{ [(1 2) 3 (4 5)] 7 [(11 15 20) 27 (27 30)] }
Draw the tree. Illustrate the insertion of key 18, then the deletion of keys 30 and 4, **in each case on the original tree**.

32 String Matching

32.1 The naive string-matching algorithm

32.1-1 Show the comparisons the naive string matcher makes for the pattern $P = 0001$ in the text $T = 000010001010001$.

32.1-2 Suppose that all characters in the pattern P are different. Show how to accelerate NAIVE-STRING-MATCHER to run in time $O(n)$ on an n-character text T.

32.x The Quick Search algorithm

`<http://aszt.inf.elte.hu/~asvanyi/ds/Quick Searching. ppt>`

32.x.1 Compute $shift[0..1]$ for the pattern $P = 000$. Show the comparisons the Quick Search string matcher makes for this pattern in the text $T = 000010001010001$.

32.x.2 Compute $shift['A'..'F']$ for the pattern
$P = ABABACD$. Show the comparisons the Quick Search string matcher makes for this pattern in the text
$T = BABAEABABAFDBABABACD$.

32.x.3 Compute $shift['A','C','G','T']$ for the pattern
$P = GCAGAGAG$. Show the comparisons the Quick Search string matcher makes for this pattern in the text
$T = GCATCGCAGAGAGTATACAGTACG$.

32.4 The Knuth-Morris-Pratt algorithm

32.4-1 Compute $next[1..19]$ (also called $prefix$ or π function) for the pattern $ababbabbabbababbabb$.

32.4.2a Compute $next[1..4]$ for the pattern $P = 0001$. Show the comparisons the Knuth-Morris-Pratt string matcher makes for this pattern in the text $T = 000010001010001$.

32.4.2b Compute $next[1..5]$ for the pattern $P = abaab$. Show the comparisons the Knuth-Morris-Pratt string matcher makes for this pattern in the text $T = aaababaabaababaab$.

32.4.2c Compute $next[1..6]$ for the pattern $P = aabaab$. Show the comparisons the Knuth-Morris-Pratt string matcher makes for this pattern in the text $T = aaabaabaabaababaab$.

32.4.2d Compute $next[1..6]$ for the pattern $P = babbab$. Show the comparisons the Knuth-Morris-Pratt string matcher makes for this pattern in the text $T = ababbabbababbababbabb$.

32.4.2e Compute $next[1..7]$ for the pattern $P = ABABAKI$.
Show the comparisons the Knuth-Morris-Pratt string matcher makes for this pattern in the text
$T = BABALABABATIBABABAKI$.