TENTTI

Test exam (previous Final exam)

Takaisin kurssin etusivulle

Aloitettiin	lauantai 4. toukokuu 2024, 21.36
Tila	Palautettu
Valmis	lauantai 4. toukokuu 2024, 21.53
Suoritusaika	17 min 18 sekuntia
Pistettä	42,00/58,00
Arvosana	**7,24** pistettä maksimista 10,00 (**72,41**%)

Kysymys 1

Osittain oikein

Pisteet 6,00 kokonaispisteistä 8,00

Merkitse kysymys

Given these different time complexities:

- O
- O(n!)
- O(1)
- O(n*log n)
- O(log n)
- O(n^2)
- O(2^n)
- O(n + k)

Write them in order from fastest to slowest:

- O(1) ✓
- O(log n) ✓
- O(n) ✗
- O(n*log n) ✗
- O(n + k) ✓
- O(n^2) ✓
- O(2^n) ✓
- O(n!) ✓

Kysymys 2

Oikein

Pisteet 1,00 kokonaispisteistä 1,00

Merkitse kysymys

What is the big O notation of an algorithm that has a runtime that follows the function $f(x)=2n^3+3n^2$

a. O(n^5)

b. O(n^3) ✓

c. O(n^3+n^2)

d. O(n^2)

e. O

Vastauksesi on oikein.

Oikea vastaus on:
O(n^3)

Kysymys 3

Oikein

Pisteet 1,00 kokonaispisteistä 1,00

Merkitse kysymys

What is the time complexity for this block of code?

```
1  input_data = get_input_data()
2  output = []
3  n = len(input_data)
4  for i in range(n):
5      for j in range(i):
6          output.append(j+i)
```

Tentin navigaatio

1 2 3 4 5 6 7
8 9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
29 30 31 32 33 34 35
36 37 38 39 40

Näytä yksi sivu kerrallaan

Lopeta tarkastelu

output.append(i+j)

a. O(2n)

b. $O(n^2)$ ✔

c. O

d. O(i+j)

e. O(log n)

Vastauksesi on oikein.

Oikea vastaus on:
$O(n^2)$

Kysymys 4

Oikein

Pisteet 1,00 kokonaispisteistä 1,00

Merkitse kysymys

What are considered advantages and disadvantages of an array structure?

Remove from the start	Disadvantage ✔
Insert at end	Disadvantage ✔
Find/Search an element based on its value	Disadvantage ✔
Remove from the middle	Disadvantage ✔
Remove from the end	Disadvantage ✔
Access time to an element	Advantage ✔
Insert at the middle	Disadvantage ✔
Insert at start	Disadvantage ✔

Vastauksesi on oikein.

Oikea vastaus on:
Remove from the start → Disadvantage,

Insert at end → Disadvantage,

Find/Search an element based on its value → Disadvantage,

Remove from the middle → Disadvantage,

Remove from the end → Disadvantage,

Access time to an element → Advantage,

Insert at the middle → Disadvantage,

Insert at start → Disadvantage

Kysymys 5

Oikein

Pisteet 1,00 kokonaispisteistä 1,00

Merkitse kysymys

Being "Base" the starting memory position of an array, "index" the 0-index position of an element inside the array and "Element_size" the size in bytes of each element of the array.

How do you calculate the position in memory of the given element of the array based on its index? What the formula would be?

Write the answer without any spaces

Vastaus: Base+(index*Element_size) ✔

Oikea vastaus on: Base+Index Element_size

Kysymys 6

Oikein

Pisteet 1,00 kokonaispisteistä 1,00

Merkitse kysymys

What is the time complexity of array indexing? (Access an element of an array based on its index)

a. $O(n^2)$

b. O(1) ✔

c. O

d. O(log n)

e. O(n*log n)

Vastauksesi on oikein.

Oikea vastaus on:
O(1)

Kysymys 7
Oikein
Pisteet 1,00 kokonaispisteistä 1,00
Merkitse kysymys

You have a dynamic array of 2 elements with 2 extra positions still available. The growth factor is 2. What is the physical size of the array after adding 5 elements to the array?

- a. 8 ✔
- b. 7
- c. 128
- d. 4

Vastauksesi on oikein.

Oikea vastaus on:
8

Kysymys 8
Oikein
Pisteet 1,00 kokonaispisteistä 1,00
Merkitse kysymys

Why can't the content of a Python string be modified?

- a. Because Python strings are implement as static arrays that do not permit changes
- b. Because the Python developers didn't have time to implement that functionality at first and it was kept like that for compability reasons.
- c. Because the memory storage used for the arrays doesn't allow changes after the object creation.
- d. Because depending on the new character, the whole string should be recalculated due to an encoding change ✔

Vastauksesi on oikein.

Oikea vastaus on:
Because depending on the new character, the whole string should be recalculated due to an encoding change

Kysymys 9
Oikein
Pisteet 1,00 kokonaispisteistä 1,00
Merkitse kysymys

What is the time complexity of list indexing? (Access an element of a list based on its index)

- a. O(n*log n)
- b. $O(n^2)$
- c. O(log n)
- d. O👎 ✔
- e. O(1)

Vastauksesi on oikein.

Oikea vastaus on:
O👎

Kysymys 10
Oikein
Pisteet 1,00 kokonaispisteistä 1,00
Merkitse kysymys

What are considered advantages and disadvantages of a doubly linked list structure?

Remove from the end	Advantage ✔
Remove from the start	Advantage ✔
Find/Search an element based on its value	Disadvantage ✔
Insert at start	Advantage ✔
Insert at the middle (position is already located)	Advantage ✔
Access time to an element based on its index	Disadvantage ✔
Remove from the middle (the node is already located)	Advantage ✔

Insert at end | Advantage ✔

Vastauksesi on oikein.

Oikea vastaus on:
Remove from the end → Advantage,

Remove from the start → Advantage,

Find/Search an element based on its value → Disadvantage,

Insert at start → Advantage,

Insert at the middle (position is already located) → Advantage,

Access time to an element based on its index → Disadvantage,

Remove from the middle (the node is already located) → Advantage,

Insert at end → Advantage

Kysymys 11

Väärin

Pisteet 0,00 kokonaispisteistä 1,00

Merkitse kysymys

Given a class that implements a Singly linked list structure, implement a method that reverts the list.

For example, if the list contains [1, 2, 3, 4, 5], after calling the method, the list should contain [5, 4, 3, 2, 1]

Notice that you don't need to create new nodes, just reorganize the existing ones changing their pointers.

The skeleton of the the class is provided

Vastaus: (rangaistus: 100 %)

Tyhjennä vastaus

```
class Node:
    def __init__(self, data):
        self._data = data
        self._next = None

    def next(self):
        return self._next

    def link(self, node):
        self._next = node

    def value(self):
        return self._data


class Singly_Linked_List:
    def __init__(self):
        self._head_node = None

    def append(self, data):
        current = self._head_node
        previous = None
        while current is not None:
            previous = current
            current = current.next()
        new_node = Node(data)
        if previous is None:
            self._head_node = new_node
        else:
            previous.link(new_node)

    def print_list(self):
        current = self._head_node
        print('[', end='')
        while current is not None:
            print(current.value(), end='')
            current = current.next()
            if current is not None:
                print(', ', end='')
        print(']')

    def reverse(self):
        # Implement this method
        pass

```

	Testi	Odotettu	Saatu tulos	
✖	l = Singly_Linked_List() l.append(1) l.append(2) l.append(3)	[5, 4, 3, 2, 1]	[1, 2, 3, 4, 5]	✖

	l.append(4) l.append(5) l.reverse() l.print_list()			
✔	l = Singly_Linked_List() l.reverse() l.print_list()	[]	[]	✔
✔	l = Singly_Linked_List() l.append(1) l.reverse() l.append(2) l.print_list()	[1, 2]	[1, 2]	✔

Ohjelmasi täytyy läpäistä kaikki testit saadaksesi pisteitä.

Näytä erot

▸ **Malliratkaisu (Python3)**

Väärin

Pisteet tälle palautukselle: 0,00/1,00.

Kysymys 12

Oikein

Pisteet 1,00 kokonaispisteistä 1,00

Merkitse kysymys

What is the time complexity of an append operation (insert at the end) on a Singly linked list?

a. O(log n)

b. O👎 ✔

c. O(1)

d. O(n*log n)

e. O(n²)

Vastauksesi on oikein.

Oikea vastaus on:
O👎

Kysymys 13

Oikein

Pisteet 1,00 kokonaispisteistä 1,00

Merkitse kysymys

What is the time complexity of an append operation (insert at the end) on a Doubly linked list?

a. O(1) ✔

b. O(log n)

c. O(n²)

d. O(n*log n)

e. O👎

Vastauksesi on oikein.

Oikea vastaus on:
O(1)

Kysymys 14

Väärin

Pisteet 0,00 kokonaispisteistä 1,00

Merkitse kysymys

Write a class that implements a sorted Doubly linked list. The list is sorted always. When inserting a new element, the new node will be inserted at the right place given its value.

For example, after inserting values 9, 5, 7, 1, 3, the list will contain (in this order): [1, 3, 5, 7, 9]

A skeleton of the class is provided. Implement at least the methods that are defined. You can implement new helper methods if you consider it necessary.

Esimerkiksi:

Testi	Tulos
l = Sorted_Doubly_Linked_List() l.append(9) l.append(5) l.append(7) l.append(1) l.append(3) l.print_list()	[1, 3, 5, 7, 9]

Vastaus: (rangaistus: 100 %)

Tyhjennä vastaus

```
class Node:
    def __init__(self, data):
        self._data = data
        self._next = None
        self._previous = None
    def next(self):
        return self._next
    def previous(self):
        return self._previous
    def link_next(self, node):
        self._next = node
    def link_previous(self, node):
        self._previous = node
    def value(self):
        return self._data


class Sorted_Doubly_Linked_List:
    def __init__(self):
        self._head_node = None

    def print_list(self):
        current = self._head_node
        print('[', end='')
        while current is not None:
            print(current.value(), end='')
            current = current.next()
            if current is not None:
                print(', ', end='')
        print(']')

    def append(self, data):

        # Implement this method
        pass

```

	Testi	Odotettu	Saatu tulos	
✖	l = Sorted_Doubly_Linked_List() l.append(9) l.append(5) l.append(7) l.append(1) l.append(3) l.print_list()	[1, 3, 5, 7, 9]	[]	✖
✖	l = Sorted_Doubly_Linked_List() l.append(1) l.append(2) l.append(3) l.print_list()	[1, 2, 3]	[]	✖
✖	l = Sorted_Doubly_Linked_List() l.append(3) l.append(2) l.append(1) l.print_list()	[1, 2, 3]	[]	✖

Ohjelmasi täytyy läpäistä kaikki testit saadaksesi pisteitä.

Näytä erot

▸ Malliratkaisu (Python3)

Väärin

Pisteet tälle palautukselle: 0,00/1,00.

Kysymys 15

Väärin

Pisteet 0,00 kokonaispisteistä 1,00

Merkitse kysymys

Given the Sorted_Doubly_Linked_List that you implemented in the previous exercise. Implement a function that merges two of them. The function is actually a method in the class, that accepts another object of the same class as parameter and reorganize the links to merge both list over the current object. No new node is created, they are only reorganized. After the merging, both objects point to the same sorted nodes.

For example, given list [1, 3, 5, 7, 9] and [0, 2, 4, 6, 8,]. After merging them, both objects contain the same list [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

Esimerkiksi:

Testi	Tulos
l1 = Sorted_Doubly_Linked_List() l1.append(9) l1.append(5) l1.append(7) l1.append(1) l1.append(3) l2 = Sorted_Doubly_Linked_List() l2.append(2) l2.append(8) l2.append(0) l2.append(6) l2.append(4) l1.merge(l2) l1.print_list() l2.print_list()	[0, 1, 2, 3, 4, 5, 6, 7, 8, 9] [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

Vastaus: (rangaistus: 100 %)

Tyhjennä vastaus

```
1  # Respect the indentation, so the method can be added to the class
2  def merge(self, other):
3      # Implement the method
4      pass
5
```

	Testi	Odotettu	Saatu tulos	
✖	l1 = Sorted_Doubly_Linked_List() l1.append(9) l1.append(5) l1.append(7) l1.append(1) l1.append(3) l2 = Sorted_Doubly_Linked_List() l2.append(2) l2.append(8) l2.append(0) l2.append(6) l2.append(4) l1.merge(l2) l1.print_list() l2.print_list()	[0, 1, 2, 3, 4, 5, 6, 7, 8, 9] [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]	[1, 3, 5, 7, 9] [0, 2, 4, 6, 8]	✖
✖	l1 = Sorted_Doubly_Linked_List() l1.append(9)	[0, 1, 2, 3, 4, 5, 6, 7, 8, 9] [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]	[1, 3, 5, 7, 9] [0, 2, 4, 6, 8]	✖

	l1.append(5) l1.append(7) l1.append(1) l1.append(3) l2 = Sorted_Doubly_Linked_List() l2.append(2) l2.append(8) l2.append(0) l2.append(6) l2.append(4) l2.merge(l1) l1.print_list() l2.print_list()			
✖	l1 = Sorted_Doubly_Linked_List() l2 = Sorted_Doubly_Linked_List() l2.append(2) l2.append(8) l2.append(0) l2.append(6) l2.append(4) l1.merge(l2) l1.print_list() l2.print_list()	[0, 2, 4, 6, 8] [0, 2, 4, 6, 8]	[] [0, 2, 4, 6, 8]	✖
✖	l1 = Sorted_Doubly_Linked_List() l2 = Sorted_Doubly_Linked_List() l2.append(2) l2.append(8) l2.append(0) l2.append(6) l2.append(4) l2.merge(l1) l1.print_list() l2.print_list()	[0, 2, 4, 6, 8] [0, 2, 4, 6, 8]	[] [0, 2, 4, 6, 8]	✖
✖	l1 = Sorted_Doubly_Linked_List() l2 = Sorted_Doubly_Linked_List() l1.append(1) l1.append(3) l1.append(5) l1.append(7) l1.append(9) l1.merge(l2) l1.print_list() l2.print_list()	[1, 3, 5, 7, 9] [1, 3, 5, 7, 9]	[1, 3, 5, 7, 9] []	✖
✖	l1 = Sorted_Doubly_Linked_List() l2 = Sorted_Doubly_Linked_List() l1.append(1) l1.append(3) l1.append(5) l1.append(7) l1.append(9) l2.merge(l1) l1.print_list() l2.print_list()	[1, 3, 5, 7, 9] [1, 3, 5, 7, 9]	[1, 3, 5, 7, 9] []	✖
✔	l1 = Sorted_Doubly_Linked_List() l2 = Sorted_Doubly_Linked_List() l1.print_list() l2.print_list()	[] []	[] []	✔
✖	l1 = Sorted_Doubly_Linked_List() l1.append(0) l1.append(1) l1.append(2) l1.append(3) l1.append(4) l2 = Sorted_Doubly_Linked_List() l2.append(5) l2.append(6) l2.append(7) l2.append(8) l2.append(9) l1.merge(l2) l1.print_list()	[0, 1, 2, 3, 4, 5, 6, 7, 8, 9] [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]	[0, 1, 2, 3, 4] [5, 6, 7, 8, 9]	✖

	l2.print_list()			
✖	l1 = Sorted_Doubly_Linked_List() l1.append(0) l1.append(1) l1.append(4) l1.append(5) l2 = Sorted_Doubly_Linked_List() l2.append(2) l2.append(3) l2.append(6) l2.append(7) l2.append(8) l2.append(9) l1.merge(l2) l1.print_list() l2.print_list()	[0, 1, 2, 3, 4, 5, 6, 7, 8, 9] [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]	[0, 1, 4, 5] [2, 3, 6, 7, 8, 9]	✖

Ohjelmasi täytyy läpäistä kaikki testit saadaksesi pisteitä.

Näytä erot

▸ Malliratkaisu (Python3)

Väärin

Pisteet tälle palautukselle: 0,00/1,00.

Kysymys 16

Oikein

Pisteet 1,00 kokonaispisteistä 1,00

Merkitse kysymys

What is the minimum height of a Binary tree structure of n elements?

a. 1

b. n/2

c. n^2

d. log n ✔

e. n

Vastauksesi on oikein.

Oikea vastaus on:
log n

Kysymys 17

Oikein

Pisteet 1,00 kokonaispisteistä 1,00

Merkitse kysymys

Given that a Binary Search Tree organize the nodes at insertion time based on the value being inserted, can a BST be unbalanced?

Valitse yksi:

Tosi ✔

Epätosi

Oikea vastaus on 'Tosi'.

Kysymys 18

Oikein

Pisteet 1,00 kokonaispisteistä 1,00

Merkitse kysymys

What means that a tree is complete?

a. All nodes have two children

b. All levels have the maximum number of possible nodes

c. If a node has children, it has to have two

d. All levels, except possibly the last, are completely filled, and all nodes are as far left as possible ✔

Vastauksesi on oikein.

Oikea vastaus on:
All levels, except possibly the last, are completely filled, and all nodes are as far left as possible

Kysymys 19

What is the time complexity of finding the minimum and maximum values of a balanced tree of n

Oikein

Pisteet 1,00 kokonaispisteistä 1,00

Merkitse kysymys

elements?

a. O(2*n)

b. $O(n^2)$

c. O

d. O(1)

e. O(log n) ✔

Vastauksesi on oikein.

Oikea vastaus on:
O(log n)

Kysymys 20

Oikein

Pisteet 1,00 kokonaispisteistä 1,00

Merkitse kysymys

How the successor of a node is found?

a. The successor of a node is the left child of the node

b. The successor of a node is the most right descendant starting from the right child of the node

c. The successor of a node is the right child of the node

d. The successor of a node is the most right descendant starting from the left child of the node

e. The successor of a node is the most left descendant starting from the right child of the node ✔

Vastauksesi on oikein.

Oikea vastaus on:
The successor of a node is the most left descendant starting from the right child of the node

Kysymys 21

Väärin

Pisteet 0,00 kokonaispisteistä 1,00

Merkitse kysymys

Write a function that recursively solves the Tower of Hanoi game (follow the link to see the rules if you don't know the game)

A skeleton of the function is provided. The function should be callable with the number of pieces to move as only parameter

The general idea is:

- Move n-1 from source to auxiliary
- Move 1 from source to destination
- Move n-1 from auxiliary to destination

And the base case being when the number of pieces to move is 1.

The function uses a list of 3 lists that work as **stacks**. When moving a piece, it is popped from one stack to be pushed into the other. The function should print the main stack (the list of lists) at every step. That is, first the original position and then after each movement. Finally the function should return the number of moves done to fulfill the task.

Even if this instructions can seem complicated, a solution can be accomplished writing less than 10 lines more than already are.

Esimerkiksi:

Testi	Tulos
print(tower_of_hanoi(3))	[['C', 'B', 'A'], [], []] [['C', 'B'], [], ['A']] [['C'], ['B'], ['A']] [['C'], ['B', 'A'], []] [[], ['B', 'A'], ['C']] [['A'], ['B'], ['C']] [['A'], [], ['C', 'B']] [[], [], ['C', 'B', 'A']] 8

Vastaus: (rangaistus: 10, 20, ... %)

Tyhjennä vastaus

```
1 def tower_of_hanoi(count, stacks=None, source=0, auxiliary=1, destination=2, moves=0):
```

```
def tower_of_hanoi(count, stacks=None, source=0, auxiliary=1, destination=2, moves=0):
    if not stacks:
        stacks = [['ABCDEFGHIJKLMNOPQRSTUVWXYZ'[i] for i in range(count-1, -1, -1)], [], [
        moves = 1
        print(stacks)
    # COMPLETE FROM HERE

```

	Testi	Odotettu	Saatu tulos
✖	print(tower_of_hanoi(3))	[['C', 'B', 'A'], [], []] [['C', 'B'], [], ['A']] [['C'], ['B'], ['A']] [['C'], ['B', 'A'], []] [[], ['B', 'A'], ['C']] [['A'], ['B'], ['C']] [['A'], [], ['C', 'B']] [[], [], ['C', 'B', 'A']] 8	[['C', 'B', 'A'], [], []] None
✖	print(tower_of_hanoi(1))	[['A'], [], []] [[], [], ['A']] 2	[['A'], [], []] None
✖	print(tower_of_hanoi(6))	[['F', 'E', 'D', 'C', 'B', 'A'], [], []] [['F', 'E', 'D', 'C', 'B'], ['A'], []] [['F', 'E', 'D', 'C'], ['A'], ['B']] [['F', 'E', 'D', 'C'], [], ['B', 'A']] [['F', 'E', 'D'], ['C'], ['B', 'A']] [['F', 'E', 'D', 'A'], ['C'], ['B']] [['F', 'E', 'D', 'A'], ['C', 'B'], []] [['F', 'E', 'D'], ['C', 'B', 'A'], []] [['F', 'E'], ['C', 'B', 'A'], ['D']] [['F', 'E'], ['C', 'B'], ['D', 'A']] [['F', 'E', 'B'], ['C'], ['D', 'A']] [['F', 'E', 'B', 'A'], ['C'], ['D']] [['F', 'E', 'B', 'A'], [], ['D', 'C']] [['F', 'E', 'B'], ['A'], ['D', 'C']] [['F', 'E'], ['A'], ['D', 'C', 'B']] [['F', 'E'], [], ['D', 'C', 'B', 'A']] [['F'], ['E'], ['D', 'C', 'B', 'A']] [['F', 'A'], ['E'], ['D', 'C', 'B']] [['F', 'A'], ['E', 'B'], ['D', 'C']] [['F'], ['E', 'B', 'A'], ['D', 'C']] [['F', 'C'], ['E', 'B', 'A'], ['D']] [['F', 'C'], ['E', 'B'], ['D', 'A']] [['F', 'C', 'B'], ['E'], ['D', 'A']] [['F', 'C', 'B', 'A'], ['E'], ['D']] [['F', 'C', 'B', 'A'], ['E', 'D'], []] [['F', 'C', 'B'], ['E', 'D', 'A'], []] [['F', 'C'], ['E', 'D', 'A'], ['B']] [['F', 'C'], ['E', 'D'], ['B', 'A']] [['F'], ['E', 'D', 'C'], ['B', 'A']] [['F', 'A'], ['E', 'D', 'C'], ['B']] [['F', 'A'], ['E', 'D', 'C', 'B'], []] [['F'], ['E', 'D', 'C', 'B', 'A'], []] [[], ['E', 'D', 'C', 'B', 'A'], ['F']] [[], ['E', 'D', 'C', 'B'], ['F', 'A']] [['B'], ['E', 'D', 'C'], ['F', 'A']] [['B', 'A'], ['E', 'D', 'C'], ['F']] [['B', 'A'], ['E', 'D'], ['F', 'C']] [['B'], ['E', 'D', 'A'], ['F', 'C']] [[], ['E', 'D', 'A'], ['F', 'C', 'B']] [[], ['E', 'D'], ['F', 'C', 'B', 'A']]	[['F', 'E', 'D', 'C', 'B', 'A'], [], []] None

		[[], [], ['F', 'C', 'B', 'A']] [['D'], ['E'], ['F', 'C', 'B', 'A']] [['D', 'A'], ['E'], ['F', 'C', 'B']] [['D', 'A'], ['E', 'B'], ['F', 'C']] [['D'], ['E', 'B', 'A'], ['F', 'C']] [['D', 'C'], ['E', 'B', 'A'], ['F']] [['D', 'C'], ['E', 'B'], ['F', 'A']] [['D', 'C', 'B'], ['E'], ['F', 'A']] [['D', 'C', 'B', 'A'], ['E'], ['F']] [['D', 'C', 'B', 'A'], [], ['F', 'E']] [['D', 'C', 'B'], ['A'], ['F', 'E']] [['D', 'C'], ['A'], ['F', 'E', 'B']] [['D', 'C'], [], ['F', 'E', 'B', 'A']] [['D'], ['C'], ['F', 'E', 'B', 'A']] [['D', 'A'], ['C'], ['F', 'E', 'B']] [['D', 'A'], ['C', 'B'], ['F', 'E']] [['D'], ['C', 'B', 'A'], ['F', 'E']] [[], ['C', 'B', 'A'], ['F', 'E', 'D']] [[], ['C', 'B'], ['F', 'E', 'D', 'A']] [['B'], ['C'], ['F', 'E', 'D', 'A']] [['B', 'A'], ['C'], ['F', 'E', 'D']] [['B', 'A'], [], ['F', 'E', 'D', 'C']] [['B'], ['A'], ['F', 'E', 'D', 'C']] [[], ['A'], ['F', 'E', 'D', 'C', 'B']] [[], [], ['F', 'E', 'D', 'C', 'B', 'A']] 64	
✖	print(tower_of_hanoi(8))	[['H', 'G', 'F', 'E', 'D', 'C', 'B', 'A'], [], []] [['H', 'G', 'F', 'E', 'D', 'C', 'B'], ['A'], []] [['H', 'G', 'F', 'E', 'D', 'C'], ['A'], ['B']] [['H', 'G', 'F', 'E', 'D', 'C'], [], ['B', 'A']] [['H', 'G', 'F', 'E', 'D'], ['C'], ['B', 'A']] [['H', 'G', 'F', 'E', 'D', 'A'], ['C'], ['B']] [['H', 'G', 'F', 'E', 'D', 'A'], ['C', 'B'], []] [['H', 'G', 'F', 'E', 'D'], ['C', 'B', 'A'], []] [['H', 'G', 'F', 'E'], ['C', 'B', 'A'], ['D']] [['H', 'G', 'F', 'E'], ['C', 'B'], ['D', 'A']] [['H', 'G', 'F', 'E', 'B'], ['C'], ['D', 'A']] [['H', 'G', 'F', 'E', 'B', 'A'], ['C'], ['D']] [['H', 'G', 'F', 'E', 'B', 'A'], [], ['D', 'C']] [['H', 'G', 'F', 'E', 'B'], ['A'], ['D', 'C']] [['H', 'G', 'F', 'E'], ['A'], ['D', 'C', 'B']] [['H', 'G', 'F', 'E'], [], ['D', 'C', 'B', 'A']] [['H', 'G', 'F'], ['E'], ['D', 'C', 'B', 'A']] [['H', 'G', 'F', 'A'], ['E'], ['D', 'C', 'B']] [['H', 'G', 'F', 'A'], ['E', 'B'], ['D', 'C']] [['H', 'G', 'F'], ['E', 'B', 'A'], ['D', 'C']] [['H', 'G', 'F', 'C'], ['E', 'B', 'A'], ['D']] [['H', 'G', 'F', 'C'], ['E', 'B'], ['D', 'A']] [['H', 'G', 'F', 'C', 'B'], ['E'], ['D', 'A']] [['H', 'G', 'F', 'C', 'B', 'A'], ['E'], ['D']] [['H', 'G', 'F', 'C', 'B', 'A'], ['E', 'D'], []] [['H', 'G', 'F', 'C', 'B'], ['E', 'D', 'A'], []] [['H', 'G', 'F', 'C'], ['E', 'D', 'A'], ['B']] [['H', 'G', 'F', 'C'], ['E', 'D'], ['B', 'A']] [['H', 'G', 'F'], ['E', 'D', 'C'], ['B', 'A']] [['H', 'G', 'F', 'A'], ['E', 'D', 'C'], ['B']] [['H', 'G', 'F', 'A'], ['E', 'D', 'C', 'B'], []] [['H', 'G', 'F'], ['E', 'D', 'C', 'B', 'A'], []] [['H', 'G'], ['E', 'D', 'C', 'B', 'A'], ['F']] [['H', 'G'], ['E', 'D', 'C', 'B'], ['F', 'A']] [['H', 'G', 'B'], ['E', 'D', 'C'], ['F', 'A']] [['H', 'G', 'B', 'A'], ['E', 'D', 'C'], ['F']] [['H', 'G', 'B', 'A'], ['E', 'D'], ['F', 'C']] [['H', 'G', 'B'], ['E', 'D', 'A'], ['F', 'C']] [['H', 'G'], ['E', 'D', 'A'], ['F', 'C', 'B']] [['H', 'G'], ['E', 'D'], ['F', 'C', 'B', 'A']] [['H', 'G', 'D'], ['E'], ['F', 'C', 'B', 'A']] [['H', 'G', 'D', 'A'], ['E'], ['F', 'C', 'B']] [['H', 'G', 'D', 'A'], ['E', 'B'], ['F', 'C']] [['H', 'G', 'D'], ['E', 'B', 'A'], ['F', 'C']] [['H', 'G', 'D', 'C'], ['E', 'B', 'A'], ['F']] [['H', 'G', 'D', 'C'], ['E', 'B'], ['F', 'A']] [['H', 'G', 'D', 'C', 'B'], ['E'], ['F', 'A']] [['H', 'G', 'D', 'C', 'B', 'A'], ['E'], ['F']] [['H', 'G', 'D', 'C', 'B', 'A'], [], ['F', 'E']]	[['H', 'G', 'F', 'E', 'D', 'C', 'B', 'A'], None

[['H', 'G', 'D', 'C', 'B'], ['A'], ['F', 'E']]
[['H', 'G', 'D', 'C'], ['A'], ['F', 'E', 'B']]
[['H', 'G', 'D', 'C'], [], ['F', 'E', 'B', 'A']]
[['H', 'G', 'D'], ['C'], ['F', 'E', 'B', 'A']]
[['H', 'G', 'D', 'A'], ['C'], ['F', 'E', 'B']]
[['H', 'G', 'D', 'A'], ['C', 'B'], ['F', 'E']]
[['H', 'G', 'D'], ['C', 'B', 'A'], ['F', 'E']]
[['H', 'G'], ['C', 'B', 'A'], ['F', 'E', 'D']]
[['H', 'G'], ['C', 'B'], ['F', 'E', 'D', 'A']]
[['H', 'G', 'B'], ['C'], ['F', 'E', 'D', 'A']]
[['H', 'G', 'B', 'A'], ['C'], ['F', 'E', 'D']]
[['H', 'G', 'B', 'A'], [], ['F', 'E', 'D', 'C']]
[['H', 'G', 'B'], ['A'], ['F', 'E', 'D', 'C']]
[['H', 'G'], ['A'], ['F', 'E', 'D', 'C', 'B']]
[['H', 'G'], [], ['F', 'E', 'D', 'C', 'B', 'A']]
[['H'], ['G'], ['F', 'E', 'D', 'C', 'B', 'A']]
[['H', 'A'], ['G'], ['F', 'E', 'D', 'C', 'B']]
[['H', 'A'], ['G', 'B'], ['F', 'E', 'D', 'C']]
[['H'], ['G', 'B', 'A'], ['F', 'E', 'D', 'C']]
[['H', 'C'], ['G', 'B', 'A'], ['F', 'E', 'D']]
[['H', 'C'], ['G', 'B'], ['F', 'E', 'D', 'A']]
[['H', 'C', 'B'], ['G'], ['F', 'E', 'D', 'A']]
[['H', 'C', 'B', 'A'], ['G'], ['F', 'E', 'D']]
[['H', 'C', 'B', 'A'], ['G', 'D'], ['F', 'E']]
[['H', 'C', 'B'], ['G', 'D', 'A'], ['F', 'E']]
[['H', 'C'], ['G', 'D', 'A'], ['F', 'E', 'B']]
[['H', 'C'], ['G', 'D'], ['F', 'E', 'B', 'A']]
[['H'], ['G', 'D', 'C'], ['F', 'E', 'B', 'A']]
[['H', 'A'], ['G', 'D', 'C'], ['F', 'E', 'B']]
[['H', 'A'], ['G', 'D', 'C', 'B'], ['F', 'E']]
[['H'], ['G', 'D', 'C', 'B', 'A'], ['F', 'E']]
[['H', 'E'], ['G', 'D', 'C', 'B', 'A'], ['F']]
[['H', 'E'], ['G', 'D', 'C', 'B'], ['F', 'A']]
[['H', 'E', 'B'], ['G', 'D', 'C'], ['F', 'A']]
[['H', 'E', 'B', 'A'], ['G', 'D', 'C'], ['F']]
[['H', 'E', 'B ...snip... B', 'A']]
[['F'], ['G', 'D', 'C'], ['H', 'E', 'B', 'A']]
[['F', 'A'], ['G', 'D', 'C'], ['H', 'E', 'B']]
[['F', 'A'], ['G', 'D', 'C', 'B'], ['H', 'E']]
[['F'], ['G', 'D', 'C', 'B', 'A'], ['H', 'E']]
[['F', 'E'], ['G', 'D', 'C', 'B', 'A'], ['H']]
[['F', 'E'], ['G', 'D', 'C', 'B'], ['H', 'A']]
[['F', 'E', 'B'], ['G', 'D', 'C'], ['H', 'A']]
[['F', 'E', 'B', 'A'], ['G', 'D', 'C'], ['H']]
[['F', 'E', 'B', 'A'], ['G', 'D'], ['H', 'C']]
[['F', 'E', 'B'], ['G', 'D', 'A'], ['H', 'C']]
[['F', 'E'], ['G', 'D', 'A'], ['H', 'C', 'B']]
[['F', 'E'], ['G', 'D'], ['H', 'C', 'B', 'A']]
[['F', 'E', 'D'], ['G'], ['H', 'C', 'B', 'A']]
[['F', 'E', 'D', 'A'], ['G'], ['H', 'C', 'B']]
[['F', 'E', 'D', 'A'], ['G', 'B'], ['H', 'C']]
[['F', 'E', 'D'], ['G', 'B', 'A'], ['H', 'C']]
[['F', 'E', 'D', 'C'], ['G', 'B', 'A'], ['H']]
[['F', 'E', 'D', 'C'], ['G', 'B'], ['H', 'A']]
[['F', 'E', 'D', 'C', 'B'], ['G'], ['H', 'A']]
[['F', 'E', 'D', 'C', 'B', 'A'], ['G'], ['H']]
[['F', 'E', 'D', 'C', 'B', 'A'], [], ['H', 'G']]
[['F', 'E', 'D', 'C', 'B'], ['A'], ['H', 'G']]
[['F', 'E', 'D', 'C'], ['A'], ['H', 'G', 'B']]
[['F', 'E', 'D', 'C'], [], ['H', 'G', 'B', 'A']]
[['F', 'E', 'D'], ['C'], ['H', 'G', 'B', 'A']]
[['F', 'E', 'D', 'A'], ['C'], ['H', 'G', 'B']]
[['F', 'E', 'D', 'A'], ['C', 'B'], ['H', 'G']]
[['F', 'E', 'D'], ['C', 'B', 'A'], ['H', 'G']]
[['F', 'E'], ['C', 'B', 'A'], ['H', 'G', 'D']]
[['F', 'E'], ['C', 'B'], ['H', 'G', 'D', 'A']]
[['F', 'E', 'B'], ['C'], ['H', 'G', 'D', 'A']]
[['F', 'E', 'B', 'A'], ['C'], ['H', 'G', 'D']]
[['F', 'E', 'B', 'A'], [], ['H', 'G', 'D', 'C']]
[['F', 'E', 'B'], ['A'], ['H', 'G', 'D', 'C']]
[['F', 'E'], ['A'], ['H', 'G', 'D', 'C', 'B']]
[['F', 'E'], [], ['H', 'G', 'D', 'C', 'B', 'A']]
[['F'], ['E'], ['H', 'G', 'D', 'C', 'B', 'A']]
[['F', 'A'], ['E'], ['H', 'G', 'D', 'C', 'B']]
[['F', 'A'], ['E', 'B'], ['H', 'G', 'D', 'C']]

[['F'], ['E', 'B', 'A'], ['H', 'G', 'D', 'C']]
[['F', 'C'], ['E', 'B', 'A'], ['H', 'G', 'D']]
[['F', 'C'], ['E', 'B'], ['H', 'G', 'D', 'A']]
[['F', 'C', 'B'], ['E'], ['H', 'G', 'D', 'A']]
[['F', 'C', 'B', 'A'], ['E'], ['H', 'G', 'D']]
[['F', 'C', 'B', 'A'], ['E', 'D'], ['H', 'G']]
[['F', 'C', 'B'], ['E', 'D', 'A'], ['H', 'G']]
[['F', 'C'], ['E', 'D', 'A'], ['H', 'G', 'B']]
[['F', 'C'], ['E', 'D'], ['H', 'G', 'B', 'A']]
[['F'], ['E', 'D', 'C'], ['H', 'G', 'B', 'A']]
[['F', 'A'], ['E', 'D', 'C'], ['H', 'G', 'B']]
[['F', 'A'], ['E', 'D', 'C', 'B'], ['H', 'G']]
[['F'], ['E', 'D', 'C', 'B', 'A'], ['H', 'G']]
[[], ['E', 'D', 'C', 'B', 'A'], ['H', 'G', 'F']]
[[], ['E', 'D', 'C', 'B'], ['H', 'G', 'F', 'A']]
[['B'], ['E', 'D', 'C'], ['H', 'G', 'F', 'A']]
[['B', 'A'], ['E', 'D', 'C'], ['H', 'G', 'F']]
[['B', 'A'], ['E', 'D'], ['H', 'G', 'F', 'C']]
[['B'], ['E', 'D', 'A'], ['H', 'G', 'F', 'C']]
[[], ['E', 'D', 'A'], ['H', 'G', 'F', 'C', 'B']]
[[], ['E', 'D'], ['H', 'G', 'F', 'C', 'B', 'A']]
[['D'], ['E'], ['H', 'G', 'F', 'C', 'B', 'A']]
[['D', 'A'], ['E'], ['H', 'G', 'F', 'C', 'B']]
[['D', 'A'], ['E', 'B'], ['H', 'G', 'F', 'C']]
[['D'], ['E', 'B', 'A'], ['H', 'G', 'F', 'C']]
[['D', 'C'], ['E', 'B', 'A'], ['H', 'G', 'F']]
[['D', 'C'], ['E', 'B'], ['H', 'G', 'F', 'A']]
[['D', 'C', 'B'], ['E'], ['H', 'G', 'F', 'A']]
[['D', 'C', 'B', 'A'], ['E'], ['H', 'G', 'F']]
[['D', 'C', 'B', 'A'], [], ['H', 'G', 'F', 'E']]
[['D', 'C', 'B'], ['A'], ['H', 'G', 'F', 'E']]
[['D', 'C'], ['A'], ['H', 'G', 'F', 'E', 'B']]
[['D', 'C'], [], ['H', 'G', 'F', 'E', 'B', 'A']]
[['D'], ['C'], ['H', 'G', 'F', 'E', 'B', 'A']]
[['D', 'A'], ['C'], ['H', 'G', 'F', 'E', 'B']]
[['D', 'A'], ['C', 'B'], ['H', 'G', 'F', 'E']]
[['D'], ['C', 'B', 'A'], ['H', 'G', 'F', 'E']]
[[], ['C', 'B', 'A'], ['H', 'G', 'F', 'E', 'D']]
[[], ['C', 'B'], ['H', 'G', 'F', 'E', 'D', 'A']]
[['B'], ['C'], ['H', 'G', 'F', 'E', 'D', 'A']]
[['B', 'A'], ['C'], ['H', 'G', 'F', 'E', 'D']]
[['B', 'A'], [], ['H', 'G', 'F', 'E', 'D', 'C']]
[['B'], ['A'], ['H', 'G', 'F', 'E', 'D', 'C']]
[[], ['A'], ['H', 'G', 'F', 'E', 'D', 'C', 'B']]
[[], [], ['H', 'G', 'F', 'E', 'D', 'C', 'B', 'A']]
256

Ohjelmasi täytyy läpäistä kaikki testit saadaksesi pisteitä.

Näytä erot

▸ **Malliratkaisu (Python3)**

Väärin

Pisteet tälle palautukselle: 0,00/1,00.

Kysymys 22

Oikein

Pisteet 1,00 kokonaispisteistä 1,00

Merkitse kysymys

What is the worst case time complexity for access time on a hash table?

a. O👎 ✔

b. O(n*log n)

c. O(log n)

d. O(n^2)

e. O(1)

Vastauksesi on oikein.

Oikea vastaus on:
O👎

Kysymys 23
Oikein
Pisteet 1,00 kokonaispisteistä 1,00
Merkitse kysymys

How collisions in a hash table can be avoided?

a. Selecting a high quality hash algorithm
b. They can't be avoided. At best the can be dealt with. ✔
c. Incrementing the size of the storing array from the beginning
d. Incrementing the size of the storing array when it's near full

Vastauksesi on oikein.
Oikea vastaus on:
They can't be avoided. At best the can be dealt with.

Kysymys 24
Oikein
Pisteet 1,00 kokonaispisteistä 1,00
Merkitse kysymys

What is the average case time complexity for access time on a hash table?

a. O
b. O(log n)
c. $O(n^2)$
d. O(n*log n)
e. O(1) ✔

Vastauksesi on oikein.
Oikea vastaus on:
O(1)

Kysymys 25
Oikein
Pisteet 1,00 kokonaispisteistä 1,00
Merkitse kysymys

Which of the hash table collision-handling schemes could tolerate a load factor above 1 and which could not?

Open addressing: No, it can not tolerate a load factor above 1 ✔
Chaining: Yes, it can tolerate a load factor above 1 ✔

Vastauksesi on oikein.
Oikea vastaus on:
Open addressing → No, it can not tolerate a load factor above 1,
Chaining → Yes, it can tolerate a load factor above 1

Kysymys 26
Oikein
Pisteet 1,00 kokonaispisteistä 1,00
Merkitse kysymys

Write the second entry of a hash table that results from using the hash function, f(x) = (5*n + 3) mod 8, to hash the keys 50, 27, 59, 1, 43, 52, 40, 63, 9 and 56, assuming collisions are handled by chaining. Write the result as a Python list.

Vastaus: [27,49,53] ✔

Oikea vastaus on: [27, 49 ,53]

Kysymys 27
Oikein
Pisteet 1,00 kokonaispisteistä 1,00
Merkitse kysymys

Write the hash table that results from using the hash function, f(x) = (5*n + 3) mod 8, to hash the keys 50, 27, 59, 1, 43, 52, 40 and 63, assuming collisions are handled by linear addressing. Write the result as a Python list.

Vastaus: [1,63,27,59,43,50,40,52] ✔

Oikea vastaus on: [1, 63, 27, 59, 43, 50, 40, 52]

Kysymys 28
Oikein
Pisteet 1,00 kokonaispisteistä 1,00
Merkitse kysymys

What is the worst-case time

After putting n entries in an initially empty hashtable, with collisions resolved by chaining...

What is the best case?

What is the best case time scenario? O(1) ✔

What is the worst-case time scenario? O(n) ✔

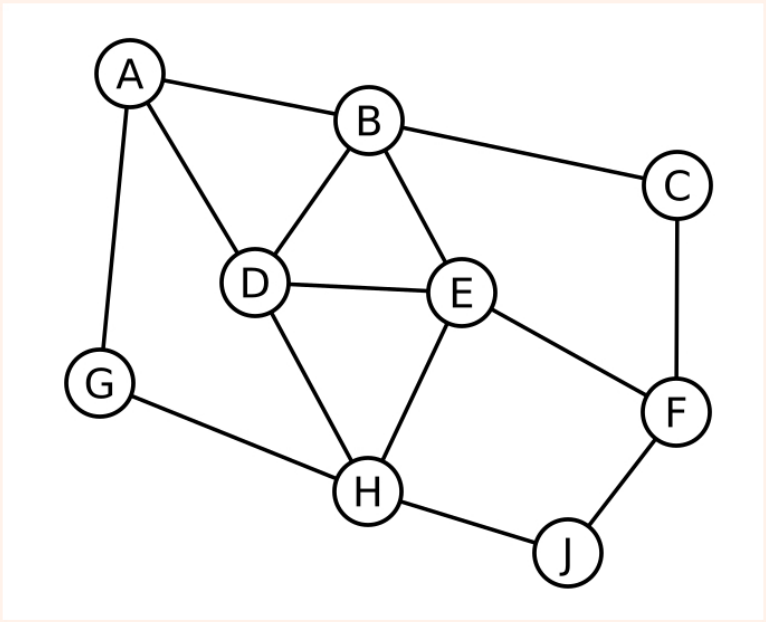
Vastauksesi on oikein.

Oikea vastaus on:
What is the best case time scenario? → O(1),

What is the worst-case time scenario? → O(n)

Kysymys 29
Oikein
Pisteet 4,00 kokonaispisteistä 4,00
Merkitse kysymys

Given the following graph:



How many vertices has the graph in total? 9 ✔

How many edges has the graph in total? 14 ✔

What is the degree of vertex D? 4 ✔

How many vertices are adjacent to vertex E? 4 ✔

Kysymys 30
Osittain oikein
Pisteet 1,00 kokonaispisteistä 2,00
Merkitse kysymys

Given the following adjacency map of a graph:

```
graph = {
    A: {B: (A, B), D: (A, D), G: (A, G)},
    B: {A: (A, B), C: (B, C), D: (B, D), E: (B, E)},
    C: {B: (B, C), F: (C, F)},
    D: {A: (A, D), B: (B, D), E: (D, E), H: (D, H)},
    E: {B: (B, E), D: (D, E), F: (E, F), H: (E, H)},
    F: {C: (C, F), E: (E, F), J: (F, J)},
    G: {A: (A, G), H: (G, H)},
    H: {D: (D, H), E: (E, H), G: (G, H), J: (H, J)},
    J: {F: (F, J), H: (H, J)}
}
```

Are vertices C and D adjacents? no ✔

What is the minimum number of edges to go from vertex C to vertex G? 2 ✖

Kysymys 31

Given a matrix map (implemented in Python as a list of lists) that represents a series of nodes and

Väärin

Pisteet 0,00 kokonaispisteistä 1,00

Merkitse kysymys

their connections (two nodes are connected if they are neighbours, including diagonally). Write a function that finds the number of independent groups of nodes. 1 or more nodes are independent if they are not connected with other nodes.

An example matrix can be:

```
map = [
    [1, 1, 1, 0, 1],
    [1, 1, 0, 0, 1],
    [0, 0, 0, 0, 1],
    [1, 0, 1, 0, 0],
    [1, 1, 0, 0, 1]
]
```

In this matrix there are 4 different groups of nodes. Notice that 2 nodes can be neighbours also in diagonal, and hence, they belong to the same group.

The idea is to maintain a list of visited nodes and to traverse the positions of the matrix, checking if the position contains a node (value is 1) and is not visited, increment a group counter and apply a DFS function on it, that visits all connected nodes to this one. The function will finally return the counter giving the number of independent groups.

You can create helper functions you consider necessary. For example, the DFS function is a probably a good idea to be implemented on its own recursive function.

The DFS function should check the 8 possible neighbours positions that contain a node (value is 1) and are not visited, and apply DFS recursively on them. DFS should update the list of visited nodes. The function should take into account that possible neighbour's possition is out the matrix.

Note: DFS = Deep-First Search

Esimerkiksi:

Testi	Tulos
map = [[1, 1, 1, 0, 1], [1, 1, 0, 0, 1], [0, 0, 0, 1, 1], [1, 0, 0, 0, 0], [1, 1, 1, 0, 1]] print(get_groups(map))	4

Vastaus: (rangaistus: 100 %)

Tyhjennä vastaus

```
1 def DFS(coord, visited, map):
2     # coord is a tuple with map's indices (x, y)
3     pass
4
5 def get_groups(map):
6     pass
```

	Testi	Odotettu	Saatu tulos	
✖	map = [[1, 1, 1, 0, 1], [1, 1, 0, 0, 1], [0, 0, 0, 1, 1], [1, 0, 0, 0, 0], [1, 1, 1, 0, 1]] print(get_groups(map))	4	None	✖
✖	map = [[1, 0, 1, 0, 1], [0, 1, 0, 1, 0], [1, 0, 1, 0, 1], [0, 1, 0, 1, 0], [1, 0, 1, 0, 1]] print(get_groups(map))	1	None	✖
✖	map = [[1, 1, 1, 1, 1], [1, 1, 1, 1, 1], [1, 1, 1, 1, 1], [1, 1, 1, 1, 1], [1, 1, 1, 1, 1]] print(get_groups(map))	1	None	✖
✖	map = [[0, 0, 0, 0, 0], [0, 0, 0, 0, 0], [0, 0, 0, 0, 0], [0, 0, 0, 0, 0], [0, 0, 0, 0, 0]] print(get_groups(map))	0	None	✖
✖	map = [[1, 0, 0, 0, 1], [0, 1, 0, 1, 0], [0, 0, 0, 0, 0], [0, 1, 0, 1, 1], [1, 1, 0, 1, 1]] print(get_groups(map))	4	None	✖
✖	map = [[0, 0, 0, 0, 0], [0, 1, 1, 1, 0], [0, 1, 0, 1, 0], [0, 1, 1, 1, 0], [0, 0, 0, 0, 0]] print(get_groups(map))	1	None	✖
✖	map = [[1, 0, 1, 0, 1], [0, 0, 0, 0, 0], [1, 0, 1, 0, 1], [0, 0, 0, 0, 0], [1, 0, 1, 0, 1]] print(get_groups(map))	9	None	✖

Ohjelmasi täytyy läpäistä kaikki testit saadaksesi pisteitä.

Näytä erot

▸ Malliratkaisu (Python3)

Väärin
Pisteet tälle palautukselle: 0,00/1,00.

Kysymys 32
Väärin
Pisteet 0,00 kokonaispisteistä 1,00
Merkitse kysymys

You are considering traveling to a certain country for your holidays. You already have a list of cities you want to visit and your plan is to stay in one of the cities and visit the rest of them one city a day.

As you want your trip to be as cheap as possible and you don't have any city preference to stay in, you plan to write a function that given a graph with the cities, the connection between them and the cost associated to travel between them, gives you what is the best city to stay in. That is, the city that has a minimum cost to travel to the rest of the cities.

Given the graph object and the dijkstra algorithm you studied in the course, write a modified dijkstra function and main function that gives you the answer. Your function should accept the graph as a parameter and call your modified dijkstra function on each of the vertices (cities). The modified dijkstra function should return the table with the costs, so you can collect the costs for all cities and calculate which one of them has a smaller value for for traveling to the rest of cities (the minimum sum of all costs for each city). Your function should return a tuple containing the name of the city (the value of the vertex) and the total cost of traveling to the rest of the cities from that one.

Esimerkiksi:

Testi	Tulos

A = Vertex('A')
B = Vertex('B')
C = Vertex('C')
D = Vertex('D')
E = Vertex('E')
 	
AB = Edge(A, B, 93)
AC = Edge(A, C, 21)
BE = Edge(B, E, 81)
CD = Edge(C, D, 11)
CE = Edge(C, E, 18)
DE = Edge(D, E, 33)
 	
am = {
  A: {B: AB, C: AC},
  B: {A: AB, E: BE},
  C: {A: AC, D: CD, E: CE},
  D: {C: CD, E: DE},
  E: {B: BE, C: CE, D: DE},
}
 	
g = Graph(am)
print(get_best_city(g))
	('C', 149)

Vastaus: (rangaistus: 100 %)

Tyhjennä vastaus

```
1 def get_best_city(graph):
2     pass
```

	Testi	Odotettu	Saatu tulos

✖	A = Vertex('A')
	B = Vertex('B')
	C = Vertex('C')
	D = Vertex('D')
	E = Vertex('E')
	F = Vertex('F')
	G = Vertex('G')
	H = Vertex('H')
	J = Vertex('J')
	
	AB = Edge(A, B, 25)
	AD = Edge(A, D, 3)
	AG = Edge(A, G, 17)
	BC = Edge(B, C, 2)
	BD = Edge(B, D, 73)
	BE = Edge(B, E, 84)
	CF = Edge(C, F, 79)
	DE = Edge(D, E, 47)
	DH = Edge(D, H, 10)
	EF = Edge(E, F, 73)
	EH = Edge(E, H, 15)
	FJ = Edge(F, J, 48)
	GH = Edge(G, H, 38)
	HJ = Edge(H, J, 72)
	
	am = {
	A: {B: AB, D: AD, G: AG},
	B: {A: AB, C: BC, D: BD, E: BE},
	C: {B: BC, F: CF},
	D: {A: AD, B: BD, E: DE, H: DH},
	E: {B: BE, D: DE, F: EF, H: EH},
	F: {C: CF, E: EF, J: FJ},
	G: {A: AG, H: GH},
	H: {D: DH, E: EH, G: GH, J: HJ},
	J: {F: FJ, H: HJ}
	}
	
	g = Graph(am)
	
	print(get_best_city(g))
		('D', 296)	***Run error***
			Traceback (most recent call last):
			File "__tester__.python3", line 9, in <module>
			A = Vertex('A')
			NameError: name 'Vertex' is not defined

Testaaminen lopetettiin virheen vuoksi.
Ohjelmasi täytyy läpäistä kaikki testit saadaksesi pisteitä.

Näytä erot

▸ **Malliratkaisu (Python3)**

Väärin
Pisteet tälle palautukselle: 0,00/1,00.

Kysymys 33
Oikein
Pisteet 2,00 kokonaispisteistä 2,00
Merkitse kysymys

Given a heap like: [1, 16, 4, 21, 18, 42, 13, 25, 43, 68, 44, 57, 44, 39, 16], which are the two direct children of element with value 4?

First child: 42 ✔

Second Child: 13 ✔

Kysymys 34
Oikein

What is the time complexity of pop operation on a Heap structure?

Pisteet 1,00 kokonaispisteistä 1,00

Merkitse kysymys

a. O(1) ✔

b. $O(n^2)$

c. O

d. O(n*log n)

e. O(log n)

Vastauksesi on oikein.

Oikea vastaus on:
O(1)

Kysymys 35

Oikein

Pisteet 1,00 kokonaispisteistä 1,00

Merkitse kysymys

What does each pop() call return within the following sequence of priority queue operations?

- add(5)
- add(4)
- add(7)
- add(1)
- pop
- add(3)
- add(6)
- pop
- pop
- add(8)
- pop
- add(2)
- pop
- pop

Write the answer as a comma separated list of numbers

Vastaus: 1,3,4,5,2,6 ✔

Oikea vastaus on: 1, 3, 4, 5, 2, 6

Kysymys 36

Oikein

Pisteet 1,00 kokonaispisteistä 1,00

Merkitse kysymys

Associate the following search algorithms with their **average** time complexity

Interpolation search: O(log(log n)) ✔

Binary Search: O(log n) ✔

Linear search: O(n) ✔

Vastauksesi on oikein.

Oikea vastaus on:
Interpolation search → O(log(log n)),

Binary Search → O(log n),

Linear search → O(n)

Kysymys 37

Oikein

Pisteet 1,00 kokonaispisteistä 1,00

Merkitse kysymys

The best case time complexity for the Quicksort algorithm is O(n*log n). The average case is also near that. What is the complexity time if the array is already sorted and the first element of the partition is always selected as the pivot?

a. O(1)

b. O

c. O(log n)

d. $O(n^2)$ ✔

e. O(n*log n)

Vastauksesi on oikein.

Oikea vastaus on:
$O(n^2)$

Kysymys 38
Ei vastattu
Kokonaispisteistä 7,00
Merkitse kysymys

For every step of the Selection sort algorithm write the list state:

1. [15, 49, 50, 28, 9, 42, 30, 39]
2. ✖
3. ✖
4. ✖
5. ✖
6. ✖
7. ✖
8. ✖

(Respect the spaces in the list)

Kysymys 39
Oikein
Pisteet 1,00 kokonaispisteistä 1,00
Merkitse kysymys

All of the following sorting algorithms have a worst case time complexity of $O(n^2)$, but one of them has clearly worst performance due to making too many switches. Which one?

- Insertion sort
- Quicksort
- Bubble sort
- Selection sort

a. Selection sort
b. Insertion sort
c. Quicksort
d. Bubble sort ✔

Vastauksesi on oikein.
Oikea vastaus on:
Bubble sort

Kysymys 40
Oikein
Pisteet 1,00 kokonaispisteistä 1,00
Merkitse kysymys

The naive implementation of Heapsort is of **space** complexity O👎 . What is the space complexity of the common implementation?

a. O(1) ✔
b. $O(n^2)$
c. O(log n)
d. O(n*log n)
e. O👎

Vastauksesi on oikein.
Oikea vastaus on:
O(1)

Lopeta tarkastelu

◄ Chapter 10 - Exercises

Siirry...