CHALMERS
EXAMINATION / TENTAMEN

Course code/kurskod	Course name/kursnamn			
DIT962	DATAstrukturer			
Anonymous code Anonym kod		Examination date Tentamensdatum	Number of pages Antal blad	Grade Betyg
DIT962-0022-CPK		2023/01/06	TES	4

[redacted]

* I confirm that I've no mobile or other similar electronic equipment available during the examination. Jag intygar att jag inte har mobiltelefon eller annan liknande elektronisk utrustning tillgänglig under eximinationen.

Solved task Behandlade uppgifter No/nr		Points per task Poäng på uppgiften	Observe: Areas with bold contour are to completed by the teacher. Anmärkning: Rutor inom bred kontur ifylles av lärare.
1	X	G	
2	X	G	
3	X	G	
4	X	G	
5	X	G	
6	X	G	
7	X	G	
8	X	G	
9			
10	X	G	
11			
12			
13			
14			
15			
16			
17			
Bonus poäng			
Total examination points Summa poäng på tentamen		8/1	

DIT962-0022-CPK

Basic question 1: Complexity

Here is a simple function that finds the pair of elements in the input list *xs* whose sum is closest to *n*:

```
function find(n: int, xs: list of ints) -> pair of ints:
    pairs = new list of (pair of ints)

    for i from 0 to length(xs):
        for j from i+1 to length(xs):
            pairs.add((xs[i], xs[j]))

    closest = pairs[0]

    for each p in pairs:
        if abs(n - sum(p)) < abs(n - sum(closest)):
            closest = p

    return closest
```

← n Körningar

← (n-1) körningar så $\frac{n(n-1)}{2}$ (blir n-1 för varje körning)

← pairs har $\frac{(n)(n-1)}{2}$ element så denna har också $\frac{(n)(n-1)}{2}$

What is the asymptotic complexity of the function `find` in the number of elements *N* of `xs`? You may assume that adding an element to a list and all arithmetic operations (including calculating the sum of a pair) are O(1).

Write your answer in O-notation. Be as exact and simple as possible. Justify why the complexity of the function has this order of growth.

Answer

Complexity: $O(n^2)$ ✓

Justification (you can also add notes directly in the code above):

Vi kan direkt se att den första loopen kör $\frac{(n)(n-1)}{2}$ gånger då j blir i+1 i varje körning! Så första delen har n^2 komplexitet

Sedan kör en annan loop length (pairs) gånger men eftersom vi vet att pairs har $\frac{(n)(n-1)}{2}$ element. Totalt blir komplexiteta

$$\frac{2n(n-1)}{2} = n(n-1) = O(n^2) =$$

ok

Basic question 2: Sorting

Perform a quicksort partitioning of the following array, using the median of the first, middle and last element as pivot:

0	1	2	3	4	5	6	7	8
63	65	35	5	30	10	47	75	4

Note: quicksorting the left and right parts of the partition is not part of this question.

Answer

Write down how the array looks after the partitioning:

0	1	2	3	4	5	6	7	8
5	4	10	30	63	35	47	75	65

What sequence of swaps did you make when partitioning the array?

Swap (0, 4) //30 - 65 så pivot blir första elementet

Swap (1, 8) //4 - 65 båda pekarna pekade på "dåliga" elemen så vi byter plats på dem

Swap (0, 3) //30 - 5 pekarna hade samma värde nämligen 3 så vi byter plats på pivot och low

If you used a different algorithm from that of your course, explain it here:

Jag minns inte vilken vi använde men den jag använde kallas hoare partition. Poängen är att du har två pekare lo och hi som börjar på första (efter pivot) och sista elementet.

Om hi pekar på ett element < pivot så säger vi att den är dålig. Samma gäller för lo fast > pivot om båda är dåliga byter vi plats på dem. Om elementen är "bra" så går vi bara i riktningen av den motsatta pekaren. När hi och lo möts byter vi plats på lo och pivot och vi är klara!

Basic question 3: Bug disinfection

A dynamic array should have *amortised* constant complexity, O(1), for adding and removing elements at the end. Here is an attempt at implementing a dynamic array (of strings):

```
class DynamicArray<Elem>:
    internal: array of strings
    size: int

    const sizeIncrease = 100

    method addLast(x: Elem):
        if this.size >= length(this.internal):
            this.resize()
        this.internal[this.size] = x
        this.size += 1

    method resize():
        maxSize = this.size + this.sizeIncrease
        oldInternal = this.internal
        this.internal = new array with maxSize cells
        for k from 0 to (but not including) maxSize:
            this.internal[k] = oldInternal[k]
```

(Note that the loop for k includes the starting index 0, but not the ending index maxSize.)

Unfortunately, two bugs have sneaked into the code somewhere, so it doesn't behave as it should.

Answer

Mark the places in the code where the bugs are. Explain what the problem is and how to fix it:

Bug 1: oldInternal[k] will get index out of bounds error. Because k can be larger than oldInternal.length

Fix by making for loop go to oldInternal.length instead of Maxsize - 1 ✓

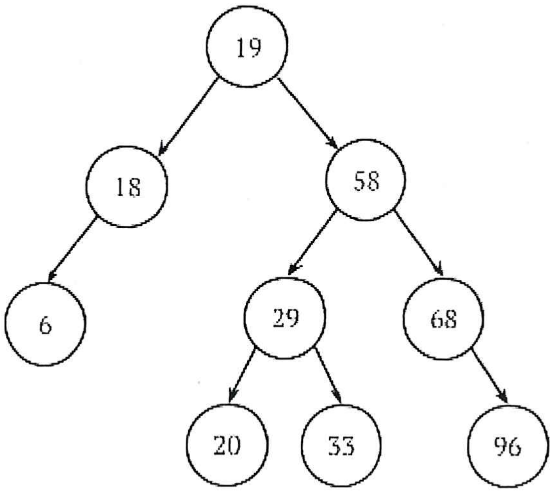
Bug 2: Complexity is not O(1) because when n gets large it will only add 100 elements at a time which is too little slow.

Fixe by making sizeIncrease = 2 and make the + in resize a multiplication sign! ✓

Ingen aning varför jag skrev på engelska

Basic question 4: Search trees

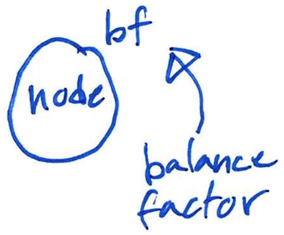
Consider the AVL tree to the right. In this question you are going to insert 25 into the tree using the AVL insertion algorithm. Additionally, you should annotate every node in the resulting tree with its AVL *balance factor* (height of the right subtree minus the height of the left subtree).

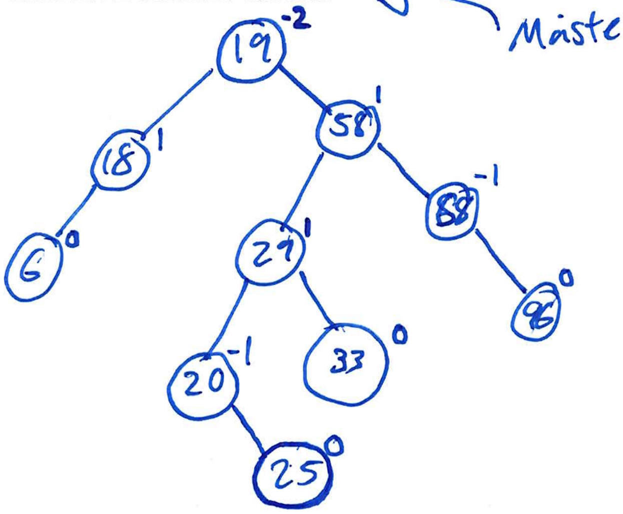


Answer

How does the tree look after you have inserted 25, but before rebalancing? Annotate the nodes in the resulting tree with their AVL balance factor.

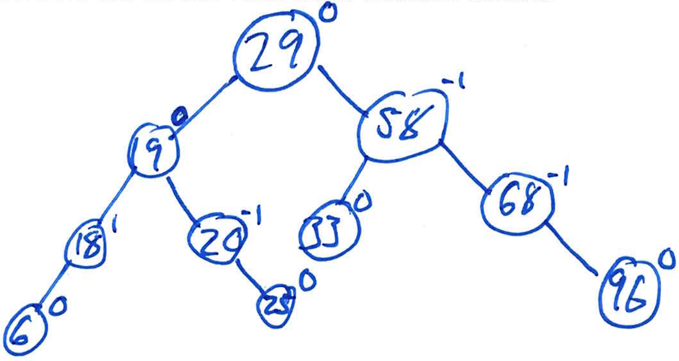
Måste roteras då |bf| > 1





How does the tree look after rebalancing? Again, annotate the nodes with their balance factor.

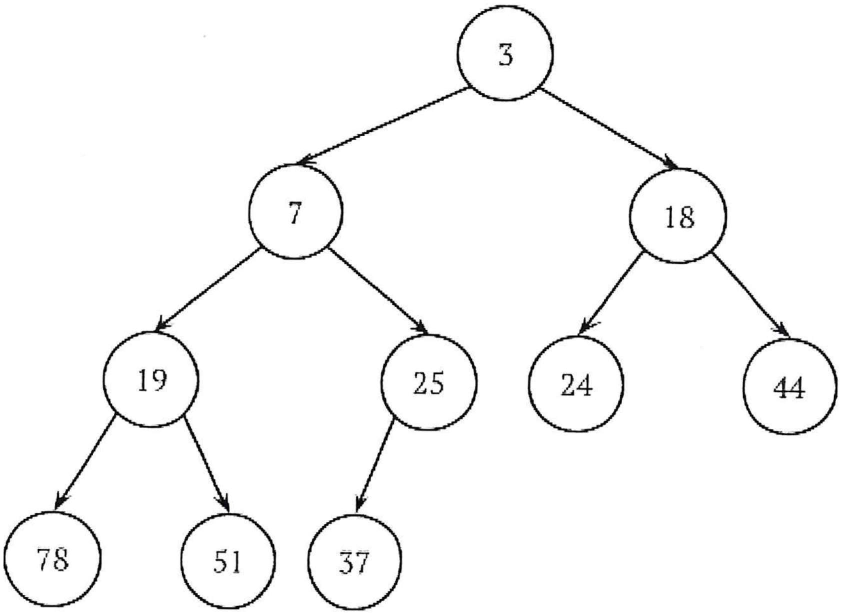
Vi kan se att vi måste först högerroterar 29 upp till 58 sedan 29 med 19 med en vänsterrotation! Efter som ~~detta~~ invarianten bryts från en insertion i RL!



G

Basic question 5: Priority queues

You are given a minimum priority queue implemented as the following binary heap:



Answer

How does the above heap look when represented as an array?
You can choose whether to start at index 0 or index 1.

0	1	2	3	4	5	6	7	8	9	10
3	7	18	19	25	24	44	78	51	37	

Now remove the minimum element from the priority queue. How does the array look afterwards?

0	1	2	3	4	5	6	7	8	9	10
7	19	18	37	25	24	44	78	51		

25 24

Vi byter plats på 3 och 37 sedan tar bort 3 och Siftar ner 37

8

Basic question 6: Hash tables

Suppose we have the following *open-addressing* hash table using *linear probing*. We are going to store strings (array of characters) consisting of the letters A, C, and T in the hash table. The hash code for the characters is: *h*(A) = 1, *h*(C) = 3, and *h*(T) = 5. The hash code for a string is the sum of the hash code of all characters in the string. For example, the hash code of ATT is:

$$h(\text{ATT}) = 1+5+5 = 11$$

The hash table uses the hash code to calculate an index by taking the remainder of the hash code divided by the size of the array (also called modular hashing):

$$index(\text{ATT}) = h(\text{ATT}) \% 10 = 11 \% 10 = 1$$

0	1	2	3	4	5	6	7	8	9
ACT	TT		C	CA	TTT	CC	AT		CCC

The hash table was created by adding the elements C, AT, CA, CC, TT, ACT, CCC, TTT **in an unknown order**. The array has never been resized, and no elements have been deleted. In what orders could the elements have been added?

A) CC, CA, AT, C, CCC, TTT, ACT ,TT

B) TT, CCC, CC, TTT, AT, ACT, C, CA

C) TT, CC, CA, TTT, AT, CCC, ACT, C

D) C, ACT, TTT, TT, CC, CCC, AT, CA

E) CA, CCC, ACT, C, TTT, CC, TT, AT

Determine which of these orders are possible (there may be several, or even none). For the others, explain why they are impossible.

Answer

Which orders are possible?

A) och E) ✓

Explain why the others are impossible:

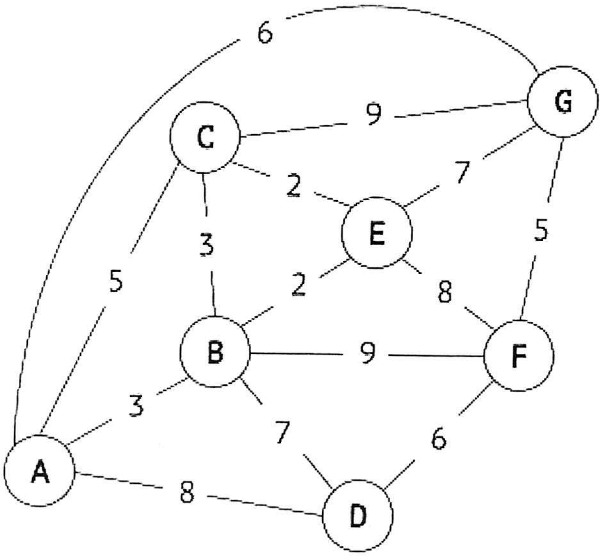
B) och C) sätter in "TT" på plats 0 så de kan inte stämma

D) sätter "C" rätt men sätter sedan "ACT" på plats 9 så det är också fel

Basic question 7: Graphs

Perform Dijkstra's algorithm on the graph to the right, starting in node **A**. Show each step of the algorithm: which node is removed from the priority queue, which node(s) are added to the priority queue, and what the priority queue looks like after each iteration.

Write the priority queue like this: "`X:4, Y:6, Z:8`", where the numbers are the priorities.



Answer

Removed node	Added node(s)	Priority queue after adding new nodes
–	A	A:0
~~A~~	~~C,B,D~~	~~B:3, C:5, D:8~~
~~B~~	~~E,F~~	~~E:2, C:5, D:8, F:9~~
~~E~~	~~G~~	~~C:5, G:7, D:8~~
A:0	B,C,D,G	B:3, C:5. G:6, D:8
B:3	C,E,F,D	C:5, E:5, C:6, G:6, D:8, D:10. F:12
C:5	G,E	E:5, G:6, E:7, D:8, D:10, F:12, G:14
E:5	F,G	G:6, D:8, D:10, F:12; G:14, G:14, F:15
G:6	F	D:8, D:10, F:11, F:12, F:15
D:8	F	F:11, F:12, F:14, F:15
F:11	—	—

Båda C försvinner

Båda E försvinner

Tre G försvinner

två D försvinner

Note that you will not fill all rows in the table.

P.G.A Plats så lägger jag inte till edges till noder som redan är visiterade!

O.BS!

Om jag tar bort en nod och det finns andra ~~[illegible]~~ vägar av större vikt till den noden tar jag bort dem också!

8

Basic question 8: Fill-in the blanks

Here is an iterative implementation of insertion into a set, represented as a binary search tree. However, someone has stolen parts of the code. Please repair it by filling in the blanks.

```
class BSTNode:
    value : number
    left  : BSTNode
    right : BSTNode

class BSTSet:
    root : BSTNode

    method add(value : number):
        parent = NULL
        node = this.root
        while node is not NULL:
            parent = node
            if value < node.value:
                node = node.left
            else if value > node.value:
                node = node.right
            else: // the value is already in the set
                return
        newnode = new BSTNode(value, NULL, NULL)
        if parent is NULL:
            this.root = newnode
        else if value < parent.value:
            Parent.left = newnode
        else: // now we know that value > parent.value
            Parent.right = newnode        ✓
```

Answer

Fill in all blanks so that the resulting code is correct.

Advanced question 9: Complexity

The following function `indexes` takes an integer *x* as input and returns a list of integers:

```
function indexes(x: int) -> list of ints:
    l = new list of ints; i = 0; n = 1

    while n <= x:
        if (x & n) > 0:    // bitwise and-operation to check
            l.addLast(i)       // if i-th bit is set, takes constant time
        i = i + 1
        n = n * 2

    return l
```

Alltså tar den log(x) tid
som mest kan l ha log(x) element

The function converts the decimal number x to a binary number (a binary number uses 2 as a base instead of 10, and consists of ones and zeros), and checks at which places (indexes) the binary digit is one. For example, the decimal number 13 is 1101 in the binary number system, and our function would then return the list [0,2,3], because the 0-th, 2-nd, and 3-rd places are one.

We use the `indexes` function in the following `powerset` function:

```
function powerset(xs: list of ints) -> list of list of ints:
    ps = new list of list of ints

    for i from 0 to 2^(length of xs):
        p = new list of ints

        for each j in indexes(i):
            p.add(xs[j])

        ps.add(p)

    return ps
```

2^n komplexitet

that returns the 'power set' of the input list xs, that is all sub-lists including the empty list and the input list itself. For example, calling `powerset` on `[1,2,3]` gives:

```
[[],[1],[2],[1,2],[3],[1,3],[2,3],[1,2,3]]
```

Asymptotically, what are the **time complexities** of the `indexes` and `powerset` functions, related to the size of their input? Assume that adding to a list and indexing is O(1). The bitwise 'and' (&) operator is O(1) as well.

Write your answers in O-notation. Be as exact and simple as possible. Explain the key reasons why each complexity has the order of growth you give.

Answer

Please answer on a separate page.

Advanced question 10: Disjoint arrays

Design an algorithm that takes two arrays, and returns true if the arrays are disjoint, i.e. have no elements in common.

You may freely use standard data structures and algorithms from the course in your solution, without explaining how they are implemented.

Your algorithm should take O(*n* log(*m*)) time, where *n* is the size of the larger array and *m* is the size of the smaller array.

Write down your algorithm as pseudocode (but Python, Java or Haskell are also fine). Explain which standard data structures and algorithms from the course that you have used.

Answer

```
            Ord a =>
disjoint :: [a] -> [a] -> Bool
disjoint xs ys                                  fromList ys
    | len xs > len ys = not (or (go xs ~~ys~~))
    | otherwise = not (or (go ys ~~xs~~))
      where ord a                               fromlist xs
         go :: [a] -> AVL a -> [Bool]
         go [] _ = False
         go (x:xs) tree = search x tree : go xs tree
```

Gär den mindre listan till ett AVL träd och kollar sedan om varje element från den längre finns i den kortare. Search har komplexitet O(logm) och körs n gånger vi kör också length och fromList men dem är båda billigare och försinner

1

Advanced question 11: Median Set

Your task is to create a data structure that can store a set of comparable elements, with special support for retrieving the *median* element in constant time. Note that we don't have any duplicates in a set. If the set is non-empty and the n elements in the set are:

$$x_0, x_1, \ldots, x_{n-1} \text{ with } x_{i-1} < x_i \text{ for all } 0 < i < n$$

then the median element is:

$$x_{\frac{n-1}{2}}$$

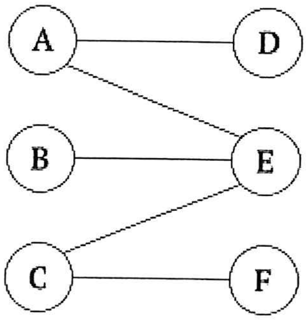
The other standard operations on a set should be supported as well. To summarise, the data structure should support the following operations, with time complexities as stated:

- add, O(log n)
- remove, O(log n)
- contains, O(log n)
- getMedian, O(1) (precondition: the set is non-empty)

You may freely use any standard data structure or algorithm from the course. Write down your data structure and the implementation of the operations using pseudo code, but you may use Java, Python or Haskell if you are more comfortable with that. Justify why your implementation of the operations has the required complexity.

Answer

Advanced question 12: Bipartite graphs



An undirected graph is *bipartite* iff the nodes can be divided into two disjoint subsets V_0 and V_1 such that all edges have one endpoint in V_0 and another in V_1. To the right is an example of a bipartite graph with V_0 = {A,B,C} and V_1 = {D,E,F}.

Design an efficient algorithm that takes a connected graph as input and returns true if and only if the graph is bipartite. By efficient we mean that its runtime complexity should be *better* than quadratic, $O(N^2)$. Also describe which data structure(s) you use to implement the graph.

What is the asymptotic runtime complexity of your algorithm, in terms of the size N? (The size of the graph is the total number of vertices and edges, or $N = |V| + |E|$). Explain your reasoning.

Answer