Data structures and algorithms, exam

(DIT961/2, DIT181/2, DAT038, DAT495, DAT525, and TDA417)

2024-10-30, 14:00–18:00

Teachers: **Jonas Duregård, 031–772 1028**
Peter Ljunglöf, Christian Sattler

We will visit exam rooms continuously throughout the exam.

Notes: *Answer directly on the question sheets!*

If you need additional space, you can use extra pages. Refer to them from the question sheet so that we don't miss them.

Write your anonymous code (not your name) on the first page of the question sheet (and on all extra pages). Don't tear pages out of this question sheet.

You may answer in English or Swedish.

Excessively complicated answers may be rejected. Write legibly, we need to be able to read your answer!

Questions and solutions will be published afterwards here:
https://github.com/ChalmersGU-data-structure-courses/past-exams

Allowed aids: One *hand-written* A4 sheet of notes (double sided).
If you make use of a sheet, you must hand it in along with your answers.

Review: When the exams have been graded, they will be available for review in the CSE Student Office at Johanneberg (EDIT floor 6) during their opening hours. If your course was originally given at Lindholmen (DIT18X, DAT495), the exams will be moved after a couple of weeks to the Lindholmen Student Office in Jupiter floor 4.

If you want to discuss the grading, you can contact the examiner via email. Remarks about the grading should be explained thoroughly. Attach a picture of your answer.

Grading: The six basic questions are graded either **correct** or **incorrect**, while the three advanced questions are awarded 0, 1 or 2 points each.

- To pass the exam, you must pass 5 of the 6 basic questions.

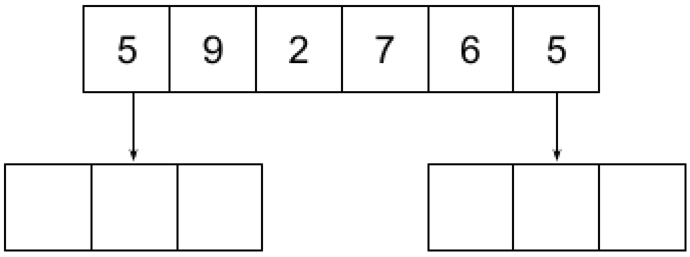
For advanced grades, the following additional requirements must be met:

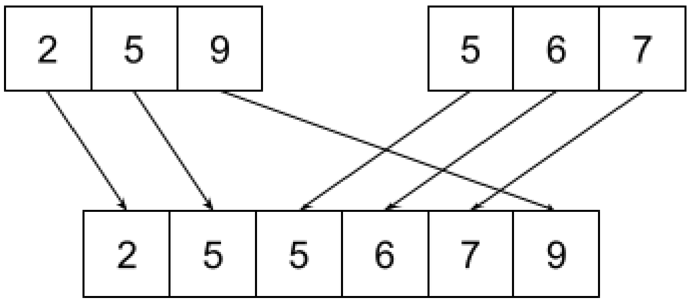
- 4: 2 out of 6 points for the advanced questions.
- 5: 4 out of 6 points for the advanced questions.
- VG: 3 out of 6 points for the advanced questions.

Write your anonymous code (*not* your name) here:______________________________

Basic question 1: Merge sort

Demonstrate Merge sort by completing this illustration (show every split/merge step).





Basic question 2: Hash tables

You have the following linear probing hash table using **modular hashing**:

0	1	2	3	4
A	C		B	D

Here are the hash values for each of the elements (before compression with %):

hash(A) = 9825498235
hash(B) = 2324213
hash(C) = 12214256
hash(D) = 3

a) Double the size of the hash table.
How does the table look after resizing?

0	1	2	3	4	5	6	7	8	9

b) If searching the resized table for E such that hash(E) = 1234, which values are E compared to? (Answer as a list of nodes, e.g. “C, A, B” if your answer is that E is compared to C, A and B in that order)

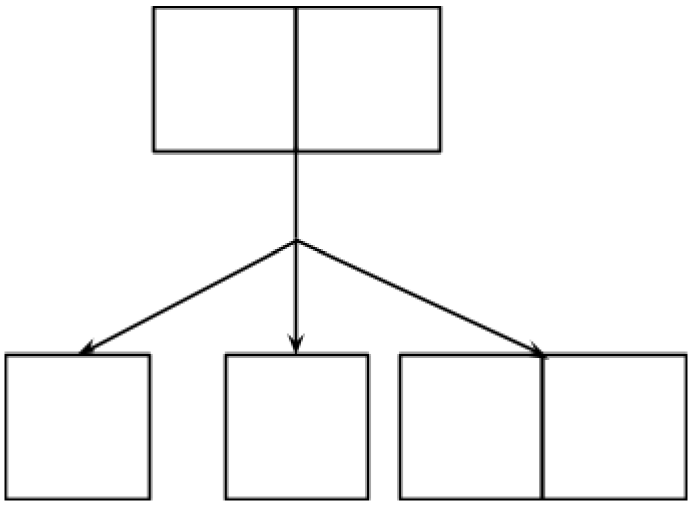
Basic question 3: 2–3 trees

Consider this 2-3 tree structure.

The tree contains the numbers 1,2,3,4,5,6.

a) Write these numbers in the correct boxes in the tree structure.

b) In which of these orders can the numbers have been added if no deletions were done? (checkmark all correct orders)



- ☐ 1, 2, 3, 4, 5, 6
- ☐ 6, 5, 4, 3, 2, 1
- ☐ 2, 1, 4, 3, 6, 5
- ☐ 1, 2, 3, 6, 5, 4
- ☐ 6, 5, 4, 1, 2, 3

Basic question 4: Sets

A set is a collection of elements without duplicates. As an ADT, it is characterised primarily by two operations: Adding values (add) and checking if a value is a member (contains).

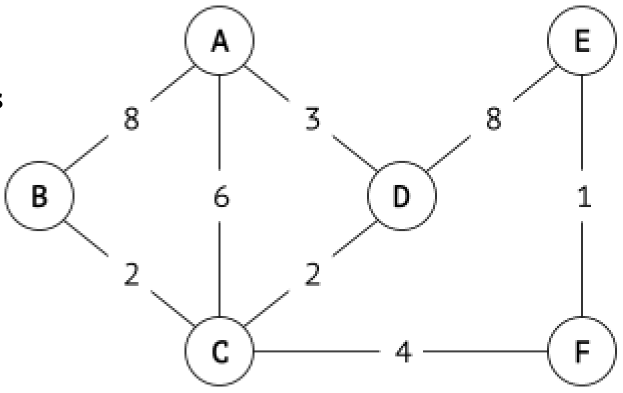
Here are three examples of data structures that can be used to implement sets. For each one, determine the worst case asymptotic complexity of the operations (assuming we use the fastest algorithms we have learned in the course).

	`add(value)`	`contains(value)`
(a) unsorted linked list		
(b) sorted dynamic array		
(c) AVL trees		

Hint: Don't forget the case of adding the same element multiple times.

Optional explanation (e.g. if you make any assumptions about the worst case, or what algorithms you use for (a) and (b)):

Basic question 5: UCS/Dijkstra's algorithm



Using the graph to the right, perform three iterations of UCS search starting in A (until you have discovered the shortest path to three nodes including A itself)

What nodes and total costs have you found?

```
A    :  0
     :
     :
```

What is the contents of the priority queue at this point (edges and total costs)?
This should be after visiting all three nodes listed above, just before you select a fourth node to include in the shortest path tree.
List items in ascending priority (total cost) order.

From-node **To-node** **Total cost**

Basic question 6: Complexity

What is the worst case time complexity of these two functions in the size **n** of the array xs, in O-notation?

```
hasPairSum(xs, k):
  for x1 in xs:
    for x2 in xs:
      if x1+x2 == k and x1!=0 and x2!=0:
        return true
  return false

countPairSums(xs):
  count = 0
  for x in xs:
    if hasPairSum(xs, x):
      count = count+1
  return count
```

hasPairSum: O(   )

countPairSums: O(   )

Short explanation (you should also write notes in the margins of the code):

Advanced question 7: Finding k smallest

Describe an algorithm for finding the k smallest elements (in ascending order) of an array xs with N elements in **worst case O(N log k) time**. You have access to a binary heap implementation that can be used for either a max- or min-priority queue, that you can use without implementing it.

Example: For an array like [3,5,1,4, 2] and k = 3, the algorithm should produce the array [1,2,3].

Hint: After 3,5,1, and 4 you have one too many elements - which one should you throw away?

(a) Justification for the O(N log k) worst case time complexity:

(b) Algorithm (pseudocode):

```
xs = your input array
result = new array of size k  (fill with k smallest elements of xs in ascending order)
```

Advanced question 8: Building bridges

An archipelago of N islands wants to build bridges that connect all the islands. Your task is to sketch a program for finding the cheapest way to do this. A company offers these price estimates:

- Bridges less than 100 metres cost 1.5 million.
- Bridges between 100 and less than 500 metres cost 1 million (cheaper than the 100m ones!)
- Bridges between 500 and less than 1000 metres cost 2 million.
- Bridges 1000 metres or longer cannot be built.

Notes: Every bridge is a straight line between two islands. You don't need to consider bridges crossing other bridges or islands. Islands are tiny enough to be considered points (no area).

Task 1: Your program needs to find if it is possible to connect all islands with bridges, and return a set of bridges with the lowest possible cost to connect all islands (if possible). You can use any algorithms and data structures taught in the course without implementing them. You need to *explain how you encode the problem* using pseudocode or *detailed* text descriptions.

Task 2: What is the complexity of your solution for N islands? (include a short motivation)

Example: The islands are given with 2D-coordinates (in metres), something like:
islands = [Island1: (0,0), Island2: (0,90), Island3: (90, 0), Island4: (90, 90)]
In this case one cheapest set of bridges for Islands 1,2,3, and 4 is: (1-4), (2-3), (1-2) for 3.5 million.

Advanced question 9: Median finding set

Your task is to make a **set** data structure with **add**, **remove** and **contains** operations. It should also be extended with a **constant time median function**.

For full points, add, remove and contains must be **O(log n)**. For partial points, they can be any complexity. The median function must be **O(1)**.

Note: For even sized sets, the median function should give the lesser of the middle values. So the median of {1,2,3,4,5,6} is 3, the median of {5,7,9,11,13} is 9, etc.

You can use standard data structures without implementing them (AVL, hash-tables, sorted lists…)

You do not have to deal with empty sets. Assume the set is non-empty before and after operations.