

[My Courses](#) / [My courses](#) / [Algorithms and Data Structures, MSc \(Spring 2023\)](#) / [Old exams](#) / [Re-exam 13 August 2021](#)

Started on Wednesday, 10 May 2023, 10:23

State Finished

Completed on Wednesday, 10 May 2023, 13:08

Time taken 2 hours 45 mins

Information

PART 1: Analysis of Algorithms

Question 1

Complete

Marked out of 1.00

Which pair of functions satisfies $f(N) \sim g(N)$?

- ☐ a. $f(N) = \log(N^2) + 1$ and $g(N) = (\log N)^2 + 17$
- ☐ b. $f(N) = (N + 1)(N - 1)$ and $g(N) = (0.5N + 0.5)(0.5N - 0.5)$
- ☐ c. $f(N) = (N + 1)(3N + \log N)$ and $g(N) = N^2 + N \log N$
- ☒ d. $f(N) = N \log(N^2) + N$ and $g(N) = 2N \log N + 1$

Question 2

Complete

Marked out of 1.00

Which pair of functions satisfy $f(N) = O(g(N))$?

- ☐ a. $f(N) = N + 2N^2$ and $g(N) = N + 2$
- ☐ b. $f(N) = 3N^2 \log N + 3N$ and $g(N) = 2N^2 + 3N$
- ☒ c. $f(N) = 2N/\log(N) + 2\log(N)$ and $g(N) = N + 7$
- ☐ d. $f(N) = 2 + \log N$ and $g(N) = 17$

Question 3

Complete

Marked out of 1.00

How many stars are printed?

```
#python3
for i in range(2*N):
    print('*')
```

```
//java
for(int i = 0; i < 2*N; i++){
    System.out.print("*");
}
```

- ☐ a. $\sim N/2$
- ☐ b. $\sim 4N^2$
- ☒ c. $\sim 4N$
- ☐ d. $\sim 2N$

Question 4

Complete

Marked out of 1.00

How many stars are printed? (Choose the *smallest* correct estimate.)

```
#python3
for i in range(2*N):
    j = i
    while j > 0:
        print('*')
        j = j // 2
```

```
//java
for (int i = 0; i < 2*N; i++){
    int j = i;
    while (j > 0){
        System.out.print("*");
        j = j / 2;
    }
}
```

- ☐ a. $O(\log N)$
- ☐ b. $O(N^2)$
- ☒ c. $O(N \log N)$
- ☐ d. $O(N)$

Question 5

Complete

Marked out of 1.00

Take a look at the following recursive program.

```
#python3
def r(N):
    if N <= 1:
        return 1
    return N + r(N - 1) + r(N - 2) + 7
```

```
//java
public static int r(int N){
    if (N <= 1){
        return 1;
    }
    return N + r(N - 1) + r(N - 2) + 7;
}
```

Let $A(N)$ denote the number of arithmetic operations (that is, additions and subtractions) performed by a call to $r(N)$. The base case is $A(0) = A(1) = 0$. Give a recurrence relation for $A(N)$ for $n \geq 1$.

- ☐ a. $A(N) \sim N + 5$
- ☐ b. $A(N) = A(2N - 3) + 3$
- ☐ c. $A(N) = A(N - 1) * A(N - 2) * 3$
- ☐ d. $A(N) = A(N - 1) + A(N - 2) + N + 7$
- ☒ e. $A(N) = A(N - 1) + A(N - 2) + 5$
- ☐ f. $A(N) = A(N) + A(N - 1) + A(N - 2) + A(7)$
- ☐ g. $A(N) = A(N - 1) + A(N - 2) + 3$

Question 6

Complete

Marked out of 1.00

Consider the following data structure. (It only has a single method and doesn't do anything interesting.)

```
#python3
class A:
    def __init__(self, n):
        self.counter = n
        self.n = n

    def mymethod(self):
        self.counter -= 1
        if self.counter == 0:
            self.counter = 1
            while self.counter < self.n:
                self.counter *= 2
```

```
//java
public class A{
    int counter;
    int n;

    public A(int n){
        this.counter = n;
        this.n = n;
    }

    public void mymethod() {
        this.counter -= 1;
        if (this.counter == 0) {
            this.counter = 1;
            while (this.counter < this.n) {
                this.counter *= 2;
            }
        }
    }
}
```

Assume N and K are positive integers. Create a new A-object:

```
#python3
a = A(N)
```

```
//java
a = new A(N);
```

Then follow K calls `a.mymethod()`. What is the running time of `mymethod`?

- ☐ a. $O(K)$ in amortized sense, $O(N)$ for a single call.
- ☐ b. $O(1)$ in both the amortized sense and for a single call.
- ☐ c. $O(K)$ in both the amortized sense and for a single call.
- ☒ d. $O(1)$ in amortized sense, $O(\log N)$ for a single call.
- ☐ e. $O(\log N)$ in both the amortized sense and for a single call.
- ☐ f. $O(\log K)$ in both the amortized sense and for a single call.
- ☐ g. $O(N)$ in amortized sense, $O(\log K)$ for a single call.
- ☐ h. $O(1)$ in amortized sense, $O(\log K)$ for a single call.
- ☐ i. $O(N)$ in both the amortized sense and for a single call.
- ☐ j. $O(N)$ in amortized sense, $O(K)$ for a single call.

PART 2: Mystery class

This question is about the data structure defined by the following code snippets:

```
# python
class Elem:
    def __init__(self, key, val):
        self.key = key
        self.val = val
        self.next = None

    def __lt__(self, other):
        return self.val < other.val

class Mystery:
    def __init__(self):
        self.head = None

    def insert(self, key, val):
        e = Elem(key, val)
        e.next = self.head
        self.head = e

    def unlink(self, elem):
        if elem == self.head:
            self.head = self.head.next
            return
        cur = self.head
        while cur.next != elem:
            cur = cur.next
        cur.next = elem.next

    def extract(self):
        minElem = None
        cur = self.head
        while cur:
            if not minElem or minElem > cur:
                minElem = cur
            cur = cur.next
        self.unlink(minElem)
        return minElem.key
```

```
// java
public class Mystery<Key, Value extends Comparable<Value>> {
    private Elem head;

    private class Elem implements Comparable<Elem>{
        Key key;
        Value val;
        Elem next;
        public Elem(Key key, Value val){
            this.key = key;
            this.val = val;
            this.next = null;
        }
        public int compareTo(Elem other){
            return this.val.compareTo(other.val);
        }
    }

    public Mystery(){
        this.head = null;
    }

    public void insert(Key key, Value val){
        Elem e = new Elem(key, val);
        e.next = this.head;
        this.head = e;
    }

    private void unlink(Elem elem){
        if(elem == this.head) {
            this.head = this.head.next;
            return;
        }
        Elem cur = this.head;
        while(cur.next != elem){
            cur = cur.next;
        }
        cur.next = elem.next;
    }

    public Key extract(){
        Elem minElem = null;
        Elem cur = this.head;
        while(cur != null){
            if( minElem == null || minElem.compareTo(cur) > 0) { // "minElem > cur", see python code
                minElem = cur;
            }
            cur = cur.next;
        }
        this.unlink(minElem);
        return minElem.key;
    }
}
```

Question 7

Not answered

Marked out of 1.00

Consider the following operations:

```
# python3
m = Mystery()
m.insert("A", 3)
m.insert("B", 10)
m.insert("C", 1)
print(m.extract())
```

```
// java
Mystery<String,Integer> m = new Mystery<String,Integer>();
m.insert("A", 3);
m.insert("B", 10);
m.insert("C", 1);
System.out.println(m.extract());
```

What is printed? (Your answer is a single character.)

Answer:

Question 8

Not answered

Marked out of 1.00

Which data type is implemented by class **Mystery**?

- ☐ a. A symbol table
- ☐ b. An unordered set of key-value pairs
- ☐ c. A priority queue
- ☐ d. A stack

Question 9

Not answered

Marked out of 1.00

What is the worst-case running time of **insert**?

- ☐ a. Constant
- ☐ b. Linearithmic in the number of stored elements
- ☐ c. Linear in the number of stored elements
- ☐ d. Logarithmic in the number of stored elements

Question 10

Not answered

Marked out of 1.00

What is the worst-case running time of **extract**?

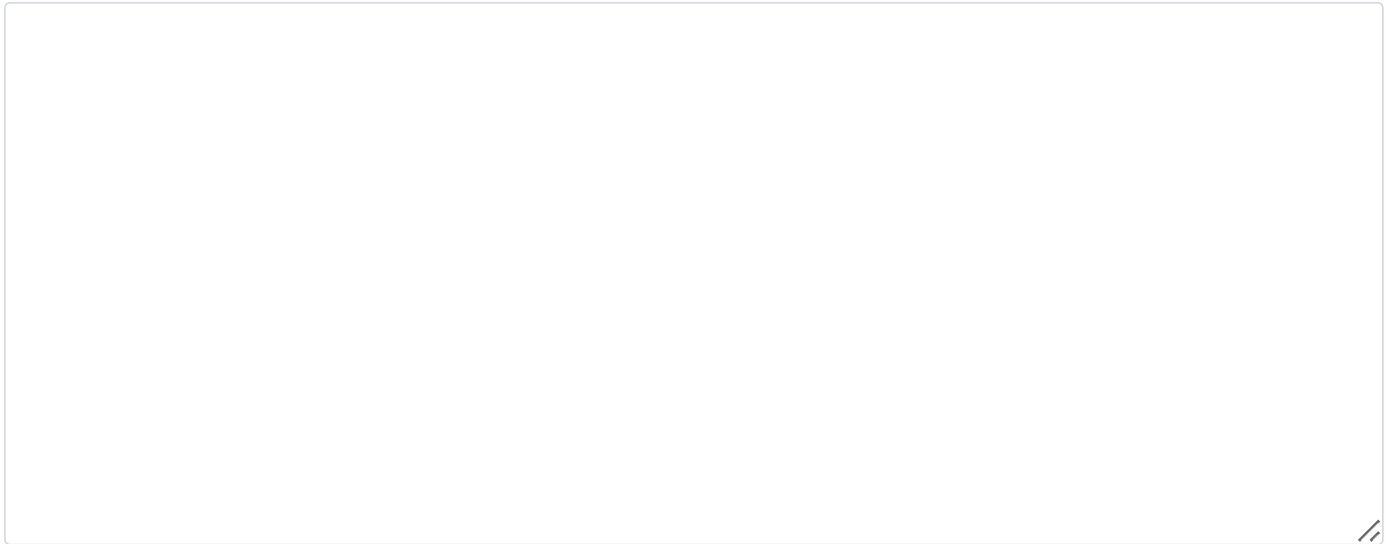
- ☐ a. Logarithmic in the number of stored elements
- ☐ b. Linear in the number of stored elements
- ☐ c. Constant
- ☐ d. Linearithmic in the number of stored elements

Question 11

Not answered

Marked out of 1.00

Implement an additional method **size()** in Python or Java that reports the number of elements stored in the data structure **without changing any other parts of the code**. Give a short description of your idea as a comment.



Question 12

Not answered

Marked out of 1.00

Give a succinct description how you would change other methods to implement `size()` with a **constant running time** in the worst-case.

Question 13

Not answered

Marked out of 1.00

Say I carry out the following code

```
# python3
m = Mystery()

for i in range(N):
    m.insert("A", 12)
    m.extract()
```

```
//java
Mystery<String,Integer> m = new Mystery<String,Integer>();
for(int i=0;i<N;i+=1){
    m.insert("A", 12);
    m.extract();
}
```

What is the running time of carrying out the whole code-fragment, including the **for**-loop, in terms of N ?

- ☐ a. Linear in N
- ☐ b. Constant
- ☐ c. Logarithmic in N
- ☐ d. Quadratic in N

Question 14

Not answered

Marked out of 1.00

Say I replace

```
#python3
def __lt__(self, other):
    return self.val < other.val
```

```
//java
public int compareTo(Elem other){
    return this.val.compareTo(other.val);
}
```

with

```
#python3
def __lt__(self, other):
    return self.val > other.val
```

```
//java
public int compareTo(Elem other){
    return other.val.compareTo(this.val);
}
```

Which statement is correct?

- ☐ a. The stack behaves as a queue.
- ☐ b. The priority queue is a MaxPQ
- ☐ c. The symbol table is hashing-based
- ☐ d. The `extract()` operation runs in constant time.

Information

PART 3: Algorithms and Data Structures

The following questions all concern the operation of standard algorithms and data structures.

The names refer to the definition and implementation found in the course book, Sedgewick and Wayne's Algorithms (4th ed.).

Question 15

Complete

Marked out of 1.00

Consider a Queue **Q** and perform the following operations:

```
Q.enqueue(4)
Q.enqueue(5)
print(Q.dequeue())
print(Q.dequeue())
Q.enqueue(6)
print(Q.size())
```

What is printed? Provide the answer *as a single line* and put a blank to separate lines.

Answer: 4 5 1

Question 16

Complete

Marked out of 1.00

Insert the keys **S O M M E R B A D** in that order into an initially empty binary search tree (Algorithm 3.3 in [SW]).

What is the maximal depth of a node in the tree after carrying out all insertions? Reminder: The depth of a node is the number of edges on the path from the root to the node.

Answer: 5

Question 17

Complete

Marked out of 1.00

Here is a (max-)heap with 5 elements.

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

If we represent this heap with an array **A** (with the conventions in [SW]), which element is in **A[4]**?

- ☐ a. 7
- ☐ b. 4
- ☐ c. 5
- ☒ d. 1
- ☐ e. 3

Question 18

Complete

Marked out of 1.00

As an example, consider the key-value pairs

A:3
C:2
D:8
B:1

inserted into a left-leaning red-black tree-based symbol table in that order (Algorithm 3.4 in [SW]). Let's agree to draw the result like this:

```

      C:2
     /  \
    B:1  D:8
   //
  A:3

```

(// represent a red link.)

As a question, starting from an **empty** data structure (**NOT** starting from the tree drawn above), insert the key-value pairs

C:1
O:4
R:3
O:2
N:1
A:5

in that order and draw the result in the above style.

```

      O:2
     //  \
    C:1  R:3
   //  \ \
  A:5  N:1

```

Question 19

Complete

Marked out of 1.00

Assume I sort the letters **T O R D E N** using Insertion sort. Which of the following situations *does not* arise at any time during the algorithm? (Multiple answers allowed, i.e. each line is graded individually.)

- ☐ a. DORTEN
- ☒ b. EORDTN
- ☐ c. DOERTN
- ☒ d. ODTREN
- ☐ e. ORDTEN

Question 20

Complete

Marked out of 1.00

Let's sort 4 strings **CAC**, **DBA**, **AAB**, **BAA** using LSD string sort (algorithm 5.1 in [SW], with $N = 4$, $W = 3$). Here is a partial trace in the style of the book:

input	d=2	d=1	d=0
CAC	DBA		
DBA		*	
AAB			
BAB			

Which string is in the cell marked *?

Answer:

Question 21

Complete

Marked out of 1.00

Let's consider the following undirected weighted graph given as a list of edges where the weight is the last number in a line:

```
0 1 5
0 3 2
1 5 7
1 3 1
0 2 4
2 3 8
5 6 8
3 6 3
```

I am running an algorithm on this graph, and this algorithm picked first the edge 0-3, then 1-3, and next 0-2. Which algorithm am I running?

- ☐ a. DFS
- ☐ b. Kruskal
- ☐ c. Prim
- ☒ d. Dijkstra
- ☐ e. BFS
- ☐ f. None of these algorithms

Question 22

Complete

Marked out of 1.00

Why do we carry out a random shuffle before sorting an array with Quick sort?

- ☐ a. To make sure that we pick the median as partitioning element.
- ☐ b. To guarantee that the expected height of the recursion tree is logarithmic.
- ☒ c. To avoid Quick sort having to deal with identical neighboring entries.
- ☐ d. To simplify the code in the partitioning method, because we don't need to handle corner-cases.
- ☐ e. To guarantee that the partitioning element is not the smallest or largest array element.
- ☐ f. Because the worst-case running time is then $O(n \log n)$.

Part 4: Data Structure Design

Story

Doctor D just opened up her shiny new vaccination clinic. According to the rules in her country, patients can just show up at her clinic to get a vaccine shot. Each patient gets assigned an identifier $i \in \{1, \dots, N\}$ when entering the clinic, starting from 1. Additionally, each patient has a severity level from 1 (lowest) to M (highest), and among all patients available, D has to give the shot to the person with highest severity. In case of equality, she has to give the shot to the person that first arrived at the clinic among the people with highest severity.

Lines are getting long, and it can happen that a person receives an increase/decrease in severity *while they are waiting*. Doctor D is in dire need of someone designing a clever data structure to help her bring order to her clinic! (Spoiler: *It's you!*)

Notation

We model the i -th person that entered the clinic as the tuple (i, s_i) , where $s_i \in \{1, \dots, M\}$ is the severity of person $i \in \{1, \dots, N\}$. In total, there are at most N people that will ever enter the clinic, and both N and M are given as input parameters.

Your data structure should support three methods: **enter**(i, s_i), **update**(i, Δ), and **treatNext**(). A person *entering* the clinic is modelled by a call to **enter** with two arguments i and s_i . An update of the severity is modelled by a call **update** with arguments $i \in \{1, \dots, N\}$ and $\Delta \in \{-M, \dots, M\}$ such that person i has severity $s_i + \Delta$. (You can assume that $s_i + \Delta$ will always be between 1 and M .) You have to implement a method **treatNext**() that reports the person to get the vaccine next. This person should no longer be part of your data structure after the call.

Example

You can think about the input received by your program as follows: (behind # is a comment)

```
3 120 # at most 3 people will enter, max severity is 120
ENTER 1 110 # person with identifier (id) 1 enters, her severity is 110.
ENTER 2 100 # person with id 2 enters, her severity is 100.
TREAT # Person with id 1 should be treated
ENTER 3 90 # person with id 3 enters, severity is 90.
UPDATE 2 -20 # Person with id 2 has severity 80
TREAT # Person with id 3 should be treated
TREAT # Person with id 2 should be treated
```

Remarks

For all of the following tasks, describe *briefly* and *precisely*, in prose, how to efficiently solve this problem. You are *very welcome* to use existing data structures and algorithms where this makes sense, please use terminology from the course text book.

Asymptotically faster is better.

On a separate line, state the asymptotic worst-case running time of the different methods of your data structure (**enter**, **update**, **treatNext**) in terms of the parameters given in the problem description. (See the section on **Notation** in the problem description.) Use the formulation "The running time of **enter** is ..." followed by a big-Oh expression. If you need, use \wedge and $_$ for superscripts and subscripts, such as x^2 for x^2 and $\log_2 y$ for $\log_2 y$. Reduce your O-expression as much as possible. That is, avoid writing $O(2n + 4n^2 + 2)$ when you mean $O(n^2)$.

Question 23

Complete

Marked out of 2.00

Let's say there will not occur any **update** operation, people show up with their final severity. Discuss the implementation of **enter** and **treatNext**.

As stated above: Make the running time explicit and keep the description short and precise. Make use of existing algorithms or data structures from the text book.

Description: We use ... as the main data structure ... **enter**(id,s_i): **treatNext**(): Running time: The running time of ``enter`` is $O(\dots)$ where [insert description of what parameters mean in the context of the problem description] The running time of ``treatNext`` is $O(\dots)$ where

Question 24

Not answered

Marked out of 1.00

In this task, you have to support **enter**, **treatNext**, and **update**.

Discuss how to implement these operations. If you implement **enter** and **treatNext** identically to your solution in Question 23, do not repeat the description.

Question 25

Not answered

Marked out of 2.00

The end of the day is near, but the government says that everyone with severity at least S has to be vaccinated today. Let's fix that there are T such patients in the clinic. Discuss how to implement two methods `count(S)` and `people(S)`. `count(S)` returns the *number* of patients in line with severity at least S (this means it should return the number T), and `people(S)` returns the set of identifiers i of people in line with severity at least S .

If you implement `enter`, `treatNext`, or `update` as in Question 24, do not repeat their discussion.