Answers of Algorithms and Data Structures, Exam 12 August 2022

The questions start on page 6.

It is strongly preferred that you fit your answers on these sheets. If you really must, you can use a separate sheet of paper instead. Please indicate that clearly.

Your name:

	1a	1b	1c	1d	1e	1f	1g	1h	2c	2d	2e	2f	3d	3e
A	A	A	A	A	A	A	A	A	A	A	A	A	A	A
B	B	B	B	B	B	B	B	B	B	B	B	B	B	B
C	C	C	C	C	C	C	C	C	C	C	C	C	C	C
D	D	D	D	D	D	D	D	D	D	D	D	D	D	D

2a

2b

2g

2h

2i

3a

3b

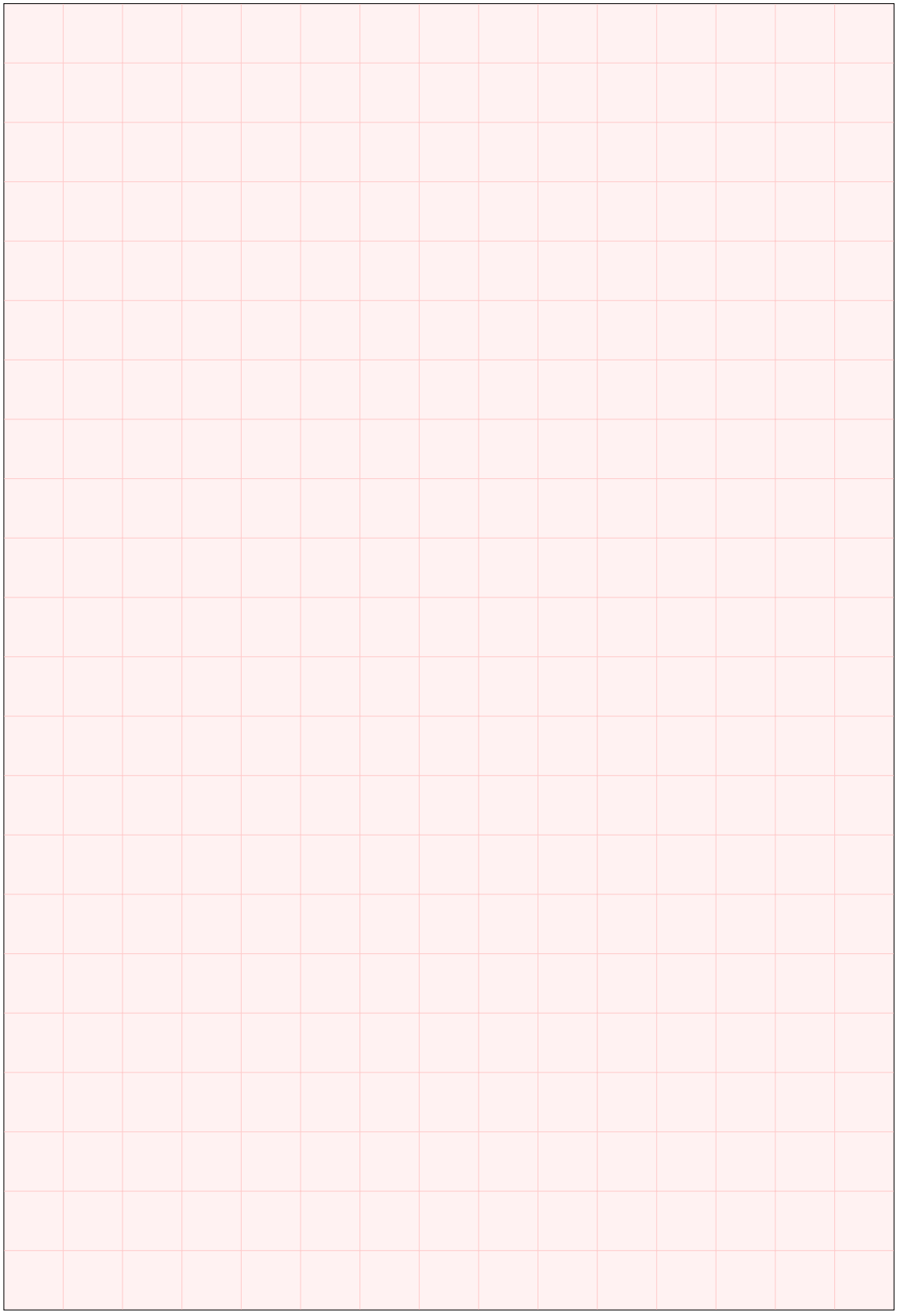
3c

3f

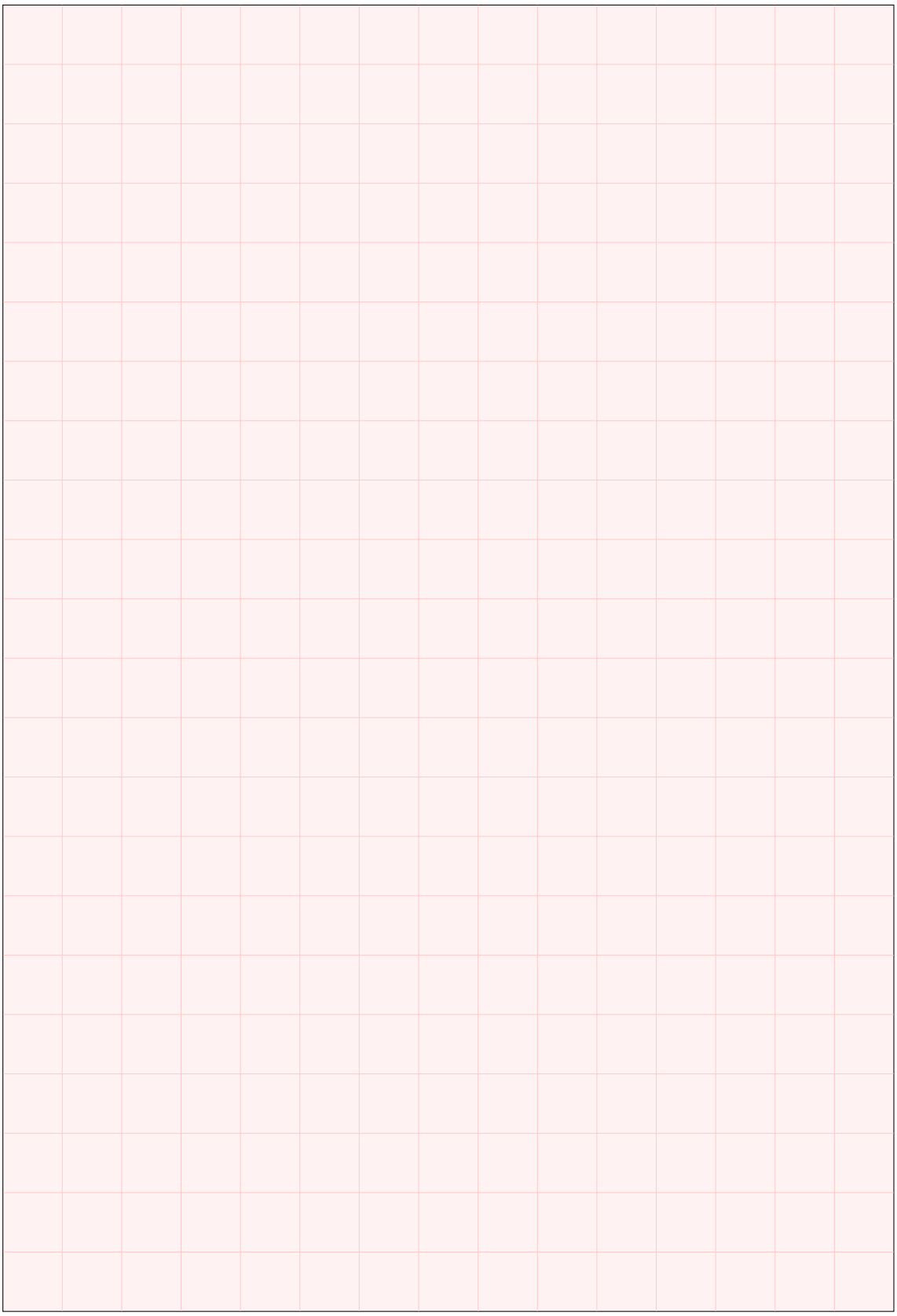
3g

4a

4b



4c



Algorithms and Data Structures

Exam 12 August 2022

Riko Jacob and Holger Dell, ITU

Instructions

What to bring. You can bring any written aid you want. This includes the course book and a dictionary. In fact, these two things are the only aids that make sense, so I recommend you bring them and only them. But if you want to bring other books, notes, print-out of code, old exams, or today's newspaper you can do so. (It won't help.) Electronic devices like mobile phones, pocket calculators, computers or e-book readers are not allowed.

Answering multiple-choice questions. In the multiple-choice questions, there is one and only one correct answer. However, to demonstrate partial knowledge, you are allowed to check 2 or more boxes, but this earns you less than full points for that question.

number of checked boxes	0	1	2	3	4
points if correct answer checked		1	0.5	0.21	0
points if correct answer not checked	0	−0.33	−0.5	−0.62	

In particular, the best thing is to only check the correct answer, and the worst thing is to check all answers but the correct one. If you don't check anything (or check *all* boxes) your score is 0. Also, if you check boxes at random, your expected score is 0. (Just to make sure: a question that is not multiple-choice cannot give you negative points.)

Where to write. Mark your answers on pages 1–5. If you really have to, you may use separate sheets of paper for the free text questions instead, but please be clear about it (cross out everything and write "see separate paper, page 1" or something like that.) For the love of all that is Good and Holy, write legibly. Hand in pages 1–5, and any separate sheet(s) of paper. Do not hand in pages 6–14, they will not be read.

Exam questions

1. Analysis of algorithms

(a) (*1 pt.*) Which pair of functions satisfy $f(N) \sim g(N)$?

- A $f(N) = N$ and $g(N) = N/2 + \log N$
- B $f(N) = \sqrt{N} + \log N$ and $g(N) = \sqrt{N} + 2\log N$
- C $f(N) = 2\sqrt{N} + \log N$ and $g(N) = \sqrt{N} + \log N$
- D $f(N) = N(\log N)$ and $g(N) = N(\log N) + N^2$

(b) (*1 pt.*) Which pair of functions satisfy $f(N) \in O(g(N))$?

- A $f(N) = 2\sqrt{N} + (\log N)$ and $g(N) = \sqrt{N}$
- B $f(N) = 4N^3$ and $g(N) = N^2 + 5N$
- C $f(N) = (N+1) \cdot (N-1)$ and $g(N) = (N-1) \cdot \log(N-1)$
- D $f(N) = (\log N) \cdot (\log N)$ and $g(N) = \log N$

(c) (*1 pt.*) How many stars are printed?

```
# python3
i = 1
while 2*i < N:
    stdio.write("*")
    i = i + 1
```

```
// java
for (int i = 1 ; 2*i < N; i = i + 1)
    StdOut.print("*");
```

A $\sim \log_2 N$ B $\sim N/2$ C $\sim N$ D $\sim 2N$

(d) (*1 pt.*) How many stars are printed? (Choose the smallest correct estimate.)

```
# python3
i = 1
while i < N:
    j = i
    while j > 0:
        stdio.write("*")
        j = j - 1
    i = i + 1
```

```
// java
for (int i = 1; i < N; i = i + 1) {
  int j = i;
  while ( j > 0) {
    StdOut.print("*");
    j = j - 1;
  }
}
```

A $O(\log N)$ B $O(N)$ C $O(N \log N)$ D $O(N^2)$

(e) (*1 pt.*) How many stars are printed? (Choose the smallest correct estimate.)

```
# python3
j = N
if N < 5:
    j = N * N
while j > 0:
    stdio.write("*")
    j = j - 2
```

```
// java
int j = N;
if (N < 5)
    j = N * N;
while (j > 0) {
    StdOut.print("*");
    j = j - 2;
}
```

A $O(\log N)$ B $O(N)$ C $O(N \log N)$ D $O(N^2)$

(f) (*1 pt.*) Find a recurrence relation for the number of arithmetic operations (additions, subtractions, and multiplications) performed by the following recursive method. The base case is $T(0) = T(1) = 0$ in all cases.

```
# python3
def r(N):
  if N > 1:
    return 2 * r(N-1) + N
  else:
    return 2
```

```
// java
static int r(int N) {
  if (N > 1)
    return 2 * r(N-1) + N;
  else
    return 2;
}
```

A $T(N) = T(N-1) + 3$
B $T(N) = 2 * T(N-1) + N \cdot N$
C $T(N) = T(N-1) + T(N-2) + 5$
D $T(N) = 2 \cdot T(N-1) + 3$

(g) (*1 pt.*) Assume I have a function `f(int K)` that runs in amortised linear time in K, but quadratic worst case time in K. What is the worst-case running time of the following piece of code? (Choose the smallest correct estimate.)

```
# python3
for i in range(N):
    f(N)
```

```
// java
for (int i = 0; i < N; i = i + 1)
    f(N);
```

A Linear in N, i.e. in $O(N)$.
B Quadratic in N, i.e. in $O(N^2)$.
C Cubic in N, i.e. in $O(N^3)$.
D Impossible to say from the information given.

(h) (*1 pt.*) What is the asymptotic running time of the following piece of code? (Choose the smallest correct estimate.) Remember that in both languages, string concatenation takes time proportional to the resulting string.

```
# python3
def f(x):
    return x + "A"

x = "A"
for i in range(N):
    x = f(x)
print(x)
```

```
// java
static String f(String x)
   { return x + "A"; }

String x = "A";
for (int i = 0; i < N; i = i + 1)
    x = f(x);
System.out.println(x);
```

A linear in N B linearithmic in N C quadratic in N D even slower

```python
class Node:
    def __init__(self, val, next):
        self.val = val
        self.next = next


class M:
    def __init__(self):
        self.first = None
        self.cur = None  # if not None, points to a node with positive value

    def insert(self, val):
        N = Node(val, self.first)
        self.first = N
        self.cur = N
        self.advance()

    def advance(self):
        while self.cur is not None and self.cur.val <= 0:
            self.cur = self.cur.next

    def getNextPositive(self):
        if self.cur is None:
            raise Exception("No more positive values")
        val = self.cur.val
        self.cur = self.cur.next
        self.advance()
        return val
```

Figure 1: Class `M`, python3 version.

```
1  public class M
2  {
3      class Node
4      {
5          int val;
6          Node next;
7      }
8
9      Node first;
10     Node cur; // if non-null, points to a node with positive value
11
12     public void insert(int val)
13     {
14         Node N = new Node();
15         N.val = val;
16         N.next = first;
17         first = N;
18         cur = N;
19         advance();
20     }
21
22     private void advance()
23     {
24         while (cur != null && cur.val <= 0) cur = cur.next;
25     }
26
27     public int getNextPositive()
28     {
29         if (cur == null)
30             throw new RuntimeException("No more positive values");
31         int val = cur.val;
32         cur = cur.next;
33         advance();
34         return val;
35     }
36 }
```

Figure 2: Class `M`, java version.

2. Class M. The next few questions all concern the class defined in Fig. 2 and Fig. 1. We let N denote the number of elements in the data structure.

(a) (*1 pt.*) What is the output of the following piece of code?

Python:

```
m = M()
m.insert(6)
m.insert(-2)
m.insert(4)
m.insert(-3)
stdio.write(m.getNextPositive())
stdio.write(", ")
stdio.write(m.getNextPositive())
```

Java:

```
M m = new M();
m.insert(6);
m.insert(-2);
m.insert(4);
m.insert(-3);
StdOut.print(m.getNextPositive());
StdOut.print(", ");
StdOut.print(m.getNextPositive());
```

(b) (*1 pt.*) Draw the data structure after the following operations: (This means "*at the end* of the operations," not "*after each* operation," so you need to draw only a single picture. Make sure to include all instance variables.)

Python:

```
m = M()
m.insert(6)
m.insert(-2)
m.insert(4)
m.insert(-3)
stdio.write(m.getNextPositive())
```

Java:

```
M m = new M();
m.insert(6);
m.insert(-2);
m.insert(4);
m.insert(-3);
StdOut.print(m.getNextPositive());
```

(c) (*1 pt.*) What is the worst-case running time of `insert`? (Choose the smallest correct estimate.)

A $O(1)$. B $O(\log N)$. C $O(N)$. D $O(N^2)$.

(d) (*1 pt.*) What is the worst-case running time of `getNextPositive()`? (Choose the smallest correct estimate.)

A $O(1)$. B $O(\log N)$. C $O(N)$. D $O(N^2)$.

(e) (*1 pt.*) What is the total worst-case running time of N consecutive calls to `getNextPositive()`? (Choose the smallest correct estimate.)

A $O(1)$. B $O(\log N)$. C $O(N)$. D $O(N^2)$.

(f) (*1 pt.*) What is the total worst-case running time of N calls, each to either `insert()` or `getNextPositive()`, with any arguments (in any order)? (Choose the smallest correct estimate.)

A $O(1)$. B $O(\log N)$. C $O(N)$. D $O(N^2)$.

(g) (*1 pt.*) Write a method `int pop()` that removes and returns the most recently inserted element (or raises a sensible exception), just like in a Stack data structure. In particular, the intended functionality is that the following piece of code prints `3` and then `5`:

```
# Python
m = M();
m.insert(5);
m.insert(-4);
m.insert(3);
print(m.pop());
print(m.getNextPositive());
```

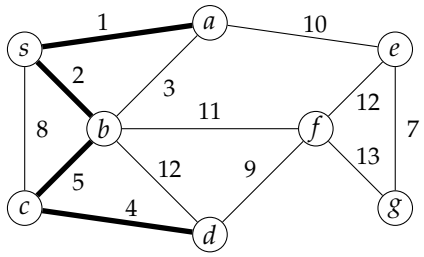
```
// Java
M m = new M();
m.insert(5);
m.insert(-4);
m.insert(3);
StdOut.println(m.pop());
StdOut.println(m.getNextPositive());
```

Don't change any other methods in `M` *and don't introduce new instance variables to* `M`.

(h) (*1 pt.*) Write a method `hasTwoNextPositive()` (for the original `M`, without `pop`) that returns `true` if "there is at least two positive value left", that is, if the next two calls to `getNextPositive()` would not raise an exception. *Don't change any other methods in* `M` *and don't introduce new instance variables to* `M`.

(i) (*1 pt.*) Can `hasTwoNextPositive()` (defined in the previous question, for the original `M`, without `pop`) be implemented so as to run in worst-case constant time, possibly by modifying other parts of `M`? If yes, explain how. If no, give an argument.

3. Operation of common algorithms and data structures.
We run three graph algorithms on the following undirected weighted graph G_{example}:



Edge weights are written next to the edges, and four edges are *thick*, which indicates that they have already been added by the algorithm.

(a) (*1 pt.*)
Suppose first that Prim's algorithm was started on this graph G_{example}. Which edge will Prim's algorithm add next? (Reminder: An edge between vertices u and v is written $\{u, v\}$.)

(b) (*1 pt.*) Suppose now that Kruskal's algorithm was started on this graph G_{example}. Which edge will Kruskal's algorithm add next?

(c) (*1 pt.*) Suppose now that Dijkstra's algorithm was started at vertex s in this graph G_{example}. Which edge will be added next?

(d) (*1 pt.*) Consider the following sequence of operations on a FIFO-queue, where a number i means `enqueue(`i`)` and "`*`" means `dequeue()`. The data structure is initially empty.

`6 * 9 7 4`

I now perform another `dequeue()`. What is returned?

A 6 B 9 C 7 D 4

(e) (*1 pt.*) Let me sort the sequence `7 4 6 9 4 5 2 4` using insertion sort in increasing order. Which one of the following intermediary states of the array can arise at some time during the algorithm?

A `2 4 6 9 4 5 7 4` B `4 4 6 7 9 5 2 4` C `5 4 6 9 4 7 2 4` D `4 7 2 9 4 5 6 4`

(f) (*1 pt.*) Consider a hash table of size 11 with linear probing. The hash function $h : \mathbb{N} \to \{0, 1, \ldots, 10\}$ with

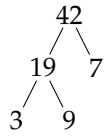
$$h(x) = (2x) \bmod 11$$

is used to insert integers into the hash table. The hash table has already been partially filled as follows:

0	1	2	3	4	5	6	7	8	9	10
33					30		42			

Add the integers 5, 6, 19, 16 in this order into the hash table and draw the result on the answer sheet.

(g) (*1 pt.*) Depicted here is a max oriented heap with five elements:



We now call DeleteMax() and Insert(10) in this order. Draw the resulting max oriented heap after the second operation.

4. Design of algorithms

Marie just arrived in Berlin during a heat wave. She wants to walk from the main station to her hotel. Due to different surface features, such as asphalt, vegetation, lakes, and houses, temperatures are very different across the city.

The city consists of S street segments and I intersections. For each street segment s, Marie knows its length ℓ_s. For each intersection i, Marie knows the temperature t_i at this intersection, and we assume this temperature to stay the same over time. Finally, the temperature at a point on a street segment is assumed to be the temperature of the closer of the two intersections the street segment is connecting.

(a) (1 *Pt.*) Suppose Marie has a maximum temperature T that she is willing to tolerate. Can you help her find a shortest walk that does not traverse any intersection of temperature larger than T? Explain your algorithm and state its running time using the parameters S and I.

(b) (2 *Pt.*) Suppose Marie has some specific health concerns regarding the temperature: While she can actually tolerate any temperature, she cannot tolerate if the temperature curve fluctuates too much. For this reason, she is only willing to take a walk along the intersections $i_1, \ldots, i_k$ if the temperature is increasing from the start intersection i_1 (the main station) to some intermediate intersection i_m, and then decreasing until the final intersection i_k (Maries hotel) is reached. More formally, the temperature curve along the walk must be *bitonic*, that is, it must satisfy the following for some m:

$$t_{i_1} \leq t_{i_2} \leq \cdots \leq t_{i_{m-1}} \leq t_{i_m} \geq t_{i_{m+1}} \geq \cdots \geq t_{i_k} .$$

Note that $m = 1$ or $m = k$ may be possible.

Can you help Marie find a shortest bitonic walk? Explain your algorithm and state its running time using the parameters S and I.

(c) (2 *Pt.*) Suppose Marie is in the situation of (b), but she notices that the temperatures at all intersections are different from each other. Moreover, the intersections are already given in sorted order from smallest to largest temperature. Explain an algorithm to find a shortest bitonic walk in time $O(S + I)$.

In all questions, use existing algorithms and data structures that were introduced in the course as much as you can, write clearly, preferably using pseudocode, and be brief.