Eidgenössische
Technische Hochschule
Zürich

Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo
Federal Institute of Technology at Zurich

Departement of Computer Science
Markus Püschel, David Steurer
Gleb Novikov, Tommaso d'Orsi, Ulysse Schaller, Rajai Nasser

25 October 2021

Algorithms & Data Structures — Exercise sheet 5 — HS 21

Exercise Class (Room & TA): _____

Submitted by: _____

Peer Feedback by: _____

Points: ____

Submission: On Monday, 1 November 2021, hand in your solution to your TA *before* the exercise class starts. Exercises that are marked by * are challenge exercises. They do not count towards bonus points.

Exercise 5.1 *Heapsort* **(1 point)**.

Given the array $[3, 6, 5, 1, 2, 4, 8, 7]$, we want to sort it in ascending order using Heapsort.

a) Draw the tree interpretation of the array as a heap, before any call of RestoreHeapCondition.

b) In the lecture you have learned a method to construct a heap from an unsorted array (see also pages 35–36 in the script). Draw the resulting max heap if this method is applied to the above array.

c) Sort the above array in ascending order with heapsort, beginning with the heap that you obtained in (b). Draw the array after each intermediate step in which a key is moved to its final position.

Exercise 5.2 *Sorting algorithms* ***(1 point)***.

Below you see four sequences of snapshots, each obtained during the execution of one of the following algorithms: `InsertionSort`, `SelectionSort`, `QuickSort`, `MergeSort`, and `BubbleSort`. For each sequence, write down the corresponding algorithm.

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

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

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

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

Exercise 5.3 *Counting Operations in Loops II.*

For the following code fragments count how many times the function f is called. Report the number of calls as nested sum, and then simplify your expression in Θ-notation and prove your result.

Hint: *Note that in order to justify your Θ-notation you are required to show two parts: an upper bound on your nested sum as well as a lower bound.*

a) Consider the snippet:

Algorithm 1

```
for j = 1, ..., n do
    k ← 1
    while k ≤ j do
        m ← 1
        while m ≤ j do
            f()
            m ← 2 · m
        k ← 2 · k
```

b) Consider the snippet:

Algorithm 2

```
for j = 1, ..., n do
    for l = 1, ..., 100 do
        k ← 1
        while k^2 ≤ j do
            f()
            f()
            k ← k + 1
```

Exercise 5.4 *Bubble sort invariant.*

Consider the pseudocode of the bubble sort algorithm on an integer array $A[1, \ldots, n]$:

Algorithm 3 BUBBLESORT(A)

```
for 1 ≤ i ≤ n do
    for 1 ≤ j ≤ n − i do
        if A[j] > A[j + 1] then
            t ← A[j]
            A[j] ← A[j + 1]
            A[j + 1] ← t
return A
```

a) Formulate an invariant INV(i) that holds at the end of the i-th iteration of the outer for-loop.

b) Using the invariant from part (a), prove the correctness of the algorithm. Specifically, prove the following three assertions:

   (i) INV(1) holds.

   (ii) If INV(i) holds, then INV$(i+1)$ holds (for all $1 \leq i < n$).

   (iii) INV(n) implies that BUBBLESORT(A) correctly sorts the array A.

Exercise 5.5 *Guessing the parity of a number* **(1 point)**.

Alice and Bob are playing a game where Alice chooses a secret integer $x \in \{1, \ldots, n\}$ and Bob has to guess the parity of x, i.e., Bob has to guess whether x is even or odd. Note that n is a fixed number that Alice and Bob agree on before starting the game. Bob is allowed to ask Alice comparison questions of the form

"Is x greater than y?"

for some $y \in \{1, \ldots, n\}$. Bob is not allowed to ask other forms of questions.

The following is an example of how the game could start:

- Alice and Bob agree on $n = 1000$.
- Alice secretly chooses $x = 541$.
- Bob asks: "Is x greater than 50?"
  - Alice answers "yes".
- Bob asks: "Is x greater than 659?"
  - Alice answers "no".

We emphasize that Bob does not have to guess the exact value of x. He only needs to find the parity of x, and he wishes to achieve this by asking as few questions as possible.

a) Assume that $n = 3$. Devise a strategy of questions for Bob and draw its decision tree

b) Now n is arbitrary. Bob has asked i comparison questions and Alice answered these questions. Let $\mathcal{X}_i \subseteq \{1, \ldots, n\}$ be the collection of all numbers that are consistent with Alice's answers[1]. Show that $\mathcal{X}_i$ is a contiguous subset of $\{1, \ldots, n\}$, i.e., there exist $a, b \in \{1, \ldots, n\}$ such that

$$\mathcal{X} = \Big\{ y \in \{1, \ldots, n\} : \ a \leq y \leq b \Big\}.$$

[1]Consider the example above with $x = 541$. After Bob's two questions, the set $\mathcal{X}_2$ contains $x = 541$ but also contains numbers such as 51, 141, 513 and 659.

Hint: *You can prove this by induction on the number of questions i.*

c) Show that in any strategy of questions that Bob can follow, Bob cannot reliably guess the parity of x without reliably guessing the number x itself.

Hint: *Show that after i questions, Bob cannot reliably guess the parity of x unless $\mathcal{X}_i$ contains a single number.*

d) Show that in any strategy of questions that Bob can follow, the number of questions that are required to reliably guess the parity of x is at least $\lceil \log_2(n) \rceil$ in the worst case.