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
François Hublet, Goran Zuzic, Tommaso d'Orsi, Jingqiu Ding

24 October 2022

Algorithms & Data Structures — Exercise sheet 5 — HS 22

The solutions for this sheet are submitted at the beginning of the exercise class on 31 October 2022.

Exercises that are marked by * are "challenge exercises". They do not count towards bonus points.

You can use results from previous parts without solving those parts.

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

Given the array $[0, 7, 2, 8, 4, 6, 3, 1]$, 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.*

Below you see four sequences of snapshots, each obtained in consecutive steps of 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 function calls in recursive functions* **(1 point)**.

For each of the following functions g, h, and k, provide an asymptotic bound in big-O notation on the number of calls to f as a function of n. You can assume that n is a power of two.

Algorithm 1

(a)
```
function g(n)
    i ← 1
    while i < n do
        f()
        i ← i + 2
    g(n/2)
    g(n/2)
    g(n/2)
```

Algorithm 2

(b)
```
function h(n)
    i ← 1
    while i < n do
        f()
        i ← i + 1
    k(n)
    k(n)
function k(n)
    i ← 2
    while i < n do
        f()
        i ← i^2
    h(n/2)
```

Exercise 5.4 *Bubble sort invariant.*

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

Algorithm 3 $\textsc{BubbleSort}(A)$

```
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 $\text{INV}(i)$ 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:

(1) $\text{INV}(1)$ holds.

(2) If $\text{INV}(i)$ holds, then $\text{INV}(i + 1)$ holds (for all $1 \leq i < n$).

(3) $\text{INV}(n)$ implies that $\textsc{BubbleSort}(A)$ correctly sorts the array A.

Exercise 5.5 *Guessing a pair of numbers* **(1 point)**.

Alice and Bob play the following game:

- Alice selects two integers $1 \leq a, b \leq 1000$, which she keeps secret
- Then, Alice and Bob repeat the following:
  - Bob chooses two integers (a', b')
  - If $a = a'$ and $b = b'$, Bob wins
  - If $a > a'$ and $b > b'$, Alice tells Bob 'high!'
  - If $a < a'$ and $b < b'$, Alice tells Bob 'low!'
  - Otherwise, Alice does not give any clue to Bob

Bob claims that he has a strategy to win this game in 12 attempts at most.

Prove that such a strategy cannot exist.

Hint: *Represent Bob's strategy as a decision tree. Each edge of the decision tree corresponds to one of Alice's answers, while each leaf corresponds to a win for Bob.*

Hint: *After defining the decision tree, you can consider the sequence $k_0 = 1$, $k_{n+1} = 3k_n + 1$, and prove that $k_n = \frac{3^{n+1}-1}{2}$. The number of leaves in the decision tree of level n should be related k_n.*