In-class Test
COMS10007 Algorithms 2018/2019

12.03.2019

Reminder: $\log n$ denotes the binary logarithm, i.e., $\log n = \log_2 n$. We also write $\log^c n$ as an abbreviation of $(\log n)^c$.

1 O-notation

1. Let $f : \mathbb{N} \rightarrow \mathbb{N}$ be a function. Define the set $\Omega(f(n))$. **(3 pts)**

Proof.

$$\Omega(f(n)) = \{g(n) : \text{There exist positive constants } c \text{ and } n_0 \text{ such that } 0 \leq cf(n) \leq g(n) \text{ for all } n \geq n_0\}$$

□

2. Give a formal proof of the statement: **(2 pts)**

$$10 \log n \in O(\log^2 n) \ .$$

Proof. We need to find constants c, n_0 such that $10 \log n \leq c \log^2 n$, for every $n \geq n_0$. The previous inequality is equivalent to $\frac{10}{c} \leq \log n$, which in turn gives $2^{\frac{10}{c}} \leq n$. We can hence for example select $c = 10$ and $n_0 = 2$. □

3. For each of the following statements, indicate whether it is true of false: (no justification needed) **(1 pt each)**

   (a) $n \in (n^2)$ true

   (b) $\log n \in O(n^3)$ true

   (c) $\log n \in O(\sqrt{\log n})$ false

   (d) $n! \in O(2^n)$ false

   (e) $2^{\sqrt{\log n}} = O(\log^2 n)$ false

   (f) $f(n) \in O(g(n))$ implies $g(n) \in \Omega(f(n))$ true

   (g) $f(n) \notin O(g(n))$ implies $g(n) \in O(f(n))$ false

2 Sorting Algorithms

Let A be an array of length n with $A[i] = A[j]$, for every $0 \leq i, j \leq n-1$.

1. What is the runtime of Heapsort on A? **(1 pt)**

   $\Theta(n)$

2. What is the runtime of Mergesort on A? **(1 pt)**

   $\Theta(n \log n)$

3. What is the runtime of Insertionsort on A? **(1 pt)**

   $\Theta(n)$

4. What are the best-case and worst-case runtimes of Mergesort? **(2 pts)**

   Both are $\Theta(n \log n)$

5. Illustrate how the Mergesort algorithm sorts the following array (for example using a recursion tree): **(2 pts)**

   9 3 2 7 1 6 11 4

   See for example slide 10 of lectures 6/7.

3 Loop-Invariant

Consider the following algorithm:

Algorithm 1

Require: A is an array of n positive integers, x is an integer

```
1: c ← 0
2: for i ← 0, 1, ..., n − 1 do
3:     if A[i] < x then
4:         c ← c + 1
5:     end if
6: end for
7: return c
```

1. Consider the for-loop of the algorithm. One of the following options is a correct loop-invariant:

   At the beginning of iteration i (i.e., after i is updated in Line 2 and before the code in Lines 3 and 4 is executed) ...

   (a) ... $c = |\{j \ : \ 0 \leq j < i \text{ and } A[j] < x\}|$
   (b) ... $c = |\{j \ : \ 0 \leq j \leq i \text{ and } A[j] < x\}|$
   (c) ... $c = |\{j \ : \ 0 \leq j < i \text{ and } A[j] \leq x\}|$
   (d) ... $c = |\{j \ : \ 0 \leq j \leq i \text{ and } A[j] \leq x\}|$

   State which one is correct. **(2 pts)**

   (a), i.e., $c = |\{j \ : \ 0 \leq j < i \text{ and } A[j] < x\}|$, is correct.

2. *Initialization:* Consider the correct invariant. Argue that at the beginning of the first iteration, i.e. when $i = 0$, the loop-invariant holds. **(1 pt)**

> *Proof.* At the beginning of the first iteration (when $i = 0$), the loop invariant states that
> $$c = |\{j \ : \ 0 \leq j < 0 \text{ and } A[j] < x\}| = |\{\}| = 0 \ ,$$
> since there is no j such that $0 \leq j < 0$. This holds since c is initialized to 0 in the line just before the loop. □

3. *Maintenance:* Consider the correct invariant. Suppose that the loop invariant holds at the beginning of iteration i. Argue that the loop-invariant then also holds at the beginning of iteration $i + 1$. **(2 pt)**

> *Proof.* Let c_i be the value of c at the beginning of iteration i. Then we have $c_i = |\{j \ : \ 0 \leq j < i \text{ and } A[j] < x\}|$. We need to show that $c_{i+1} = |\{j \ : \ 0 \leq j < i + 1 \text{ and } A[j] < x\}|$. Suppose first that $A[i] < x$. Then the algorithm increments c, i.e., we have $c_{i+1} = c_i + 1$. Observe further that:
> $$\begin{aligned} |\{j \ : \ 0 \leq j < i+1 \text{ and } A[j] < x\}| &= |\{j \ : \ 0 \leq j < i \text{ and } A[j] < x\}| \\ &+ |\{j \ : \ j = i \text{ and } A[j] < x\}| = c_i + 1 \ , \end{aligned}$$
> using the assumption $A[i] < x$. The invariant thus holds in this case.
> Next, suppose that $A[i] \geq x$. Then the algorithm does not change c, i.e., we have $c_{i+1} = c_i$. Observe further that:
> $$\begin{aligned} |\{j \ : \ 0 \leq j < i+1 \text{ and } A[j] < x\}| &= |\{j \ : \ 0 \leq j < i \text{ and } A[j] < x\}| \\ &+ |\{j \ : \ j = i \text{ and } A[j] < x\}| = c_i \ , \end{aligned}$$
> using the assumption $A[i] \geq x$. The invariant thus holds in this case.
> Since the invariant holds in both cases, the invariant always holds. □

4. *Termination:* What does the algorithm compute? Argue that this follows from the loop invariant. **(1 pt)**

> *Proof.* The algorithm computes the number of elements of the input array that are smaller than x. This can be seen by plugging in the value $i = n$ into the invariant (the state after the last iteration or before iteration $i = n$ that is never executed), which yields $c = |\{j \ : \ 0 \leq j < n \text{ and } A[j] < x\}|$. □