University of Freiburg
Dept. of Computer Science
Prof. Dr. F. Kuhn
P. Bamberger, P. Schneider

Algorithm Theory
Sample Solution Exercise Sheet 1

Due: Tuesday, 26th of October, 2021, 4 pm

Exercise 1: Sort Functions by Asymptotic Growth *(8 Points)*

Give a sequence of the following functions sorted by asymptotic growth, i.e., for consecutive functions g, f in your sequence, it should hold $g \in \mathcal{O}(f)$. Write "$g \cong f$" between two functions in the sequence if $f \in \mathcal{O}(g)$ *and* $g \in \mathcal{O}(f)$.

***Note**: No formal proofs required, but you loose* 1 *point for each error.*

$\log_2(n!)$	$\sqrt{n}$	$\sqrt{n^3}$	$\log_2(n^2)$
$n^2 + n\log_2(n^2)$	3^n	$n^{\log_2 n}$	$(\log_2 n)^2$
$\log_{10} n$	$10^{100} \cdot n$	$n!$	$n \log_2 n$
$n \cdot 2^n$	n^n	$\sqrt{\log_2 n}$	n^2

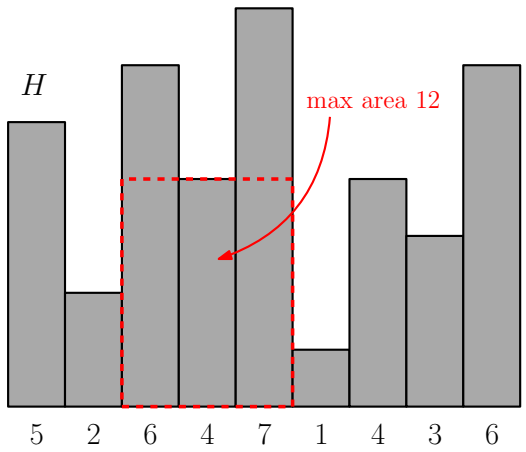
Sample Solution

For clarification, we write $g \lesssim f$ if $g \in \mathcal{O}(f)$, but not $f \in \mathcal{O}(g)$.

$$\begin{array}{ccccccccc} & & \sqrt{\log_2 n} & \lesssim & \log_{10} n & \cong & \log_2(n^2) & \lesssim & (\log_2 n)^2 \\ & \lesssim & \sqrt{n} & \lesssim & 10^{100} n & \lesssim & n\log_2 n & \cong & \log_2(n!) \\ & \lesssim & \sqrt{n^3} & \lesssim & n^2 & \cong & n^2 + n\log_2(n^2) & \lesssim & n^{\log_2 n} \\ & \lesssim & n \cdot 2^n & \lesssim & 3^n & \lesssim & n! & \lesssim & n^n \end{array}$$

Exercise 2: Maximum Rectangle in a Histogram *(12 Points)*

Consider a sequence $h_1, \ldots, h_n$ of positive, integer numbers. This sequence represents a histogram H consisting of n horizontally aligned bars each of width 1, where h_i represents the height of the i^{th} bar. The goal is to find a rectangle of maximum area completely within H (i.e., within the union of bars).



(a) Describe an algorithm that computes a maximum area rectangle in H in time $\mathcal{O}(n^2)$. *(3 Points)*

(b) Describe an algorithm that computes a maximum area rectangle that is within H and also intersects the i^{th} bar in time $\mathcal{O}(n)$ and prove the running time. *(5 Points)*

Remark: correct solutions in $o(n^2)$ grant partial points.

(c) Give an algorithm that uses the divide and conquer principle to compute a maximum area rectangle in H in time $\mathcal{O}(n \log n)$ and prove the running time. *(4 Points)*

Remark: you can use part (b), even if you did not succeed.

Sample Solution

(a) The idea is to compute for any pair i, j the largest rectangle that that starts and ends with the i^{th} and j^{th} bar respectively.

Algorithm 1 `max-area`$(h_1, \ldots, h_n)$

```
max-area ← 0
for i ← 1 to n do
    min-height ← h_i
    for j ← i to n do
        min-height ← min(min-height, h_j)
        max-area ← max(max-area, min-height · (j − i + 1))
return max-area
```

(b) The idea is to start at bar i and extend the rectangle in the direction with the higher bar.

Algorithm 2 `max-area-bar`$(i, h_1, \ldots, h_n)$

```
max-area ← 0
min-height ← h_i
ℓ, r ← i
h_0, h_{n+1} ← −∞                                   ▷ Sentinel elements
while ℓ > 1 or r < n do
    if h_{ℓ−1} ≥ h_{r+1} then
        ℓ ← ℓ − 1
        min-height ← min(min-height, h_ℓ)
    else
        r ← r + 1
        min-height ← min(min-height, h_r)
    max-area ← max(max-area, min-height · (r − ℓ + 1))
return max-area
```

Running time: The operations inside the while loop have constant running time. The while loop has at most n iterations since in each iteration we either increase r or decrease ℓ and both variables together can be increased or decrease at most n times, until they hit 1 or n. Therefore, the running time is $\mathcal{O}(n)$.

Correctness: (Not required though) For completeness sake we give an argument why the algorithm produces the correct result, i.e., the maximum rectangle in H that intersects bar i. This is not so clear (as in part (a)), since we are not checking every possibility, i.e., the area of every rectangle extending from some bar $\ell' \leq i$ to some bar $r' \geq i$, so we can not simply argue that we get the correct result by "brute force".

Instead, we will argue that we always consider a rectangle R that intersects i with maximal area. Assume that R spans from $\ell' \leq i$ to $r' \geq i$. Further assume (w.l.o.g.) that the index ℓ in the algorithm reaches ℓ' before index r reaches r' (the other case is symmetric). That means, we now have $\ell = \ell'$ but $r < r'$. We need to show that r moves to r', before ℓ moves on, so that we consider the area of R.

For a contradiction, assume that $h_{\ell-1} \geq h_{r+1}$ (i.e., $\ell \leftarrow \ell' - 1$ in the algorithm). But then $h_{\ell-1}$ is larger than the height of R, which means we could fit a rectangle R' into H with the same height as R but spanning from $\ell' - 1$ to r' (i.e., larger width). That means R' has strictly larger area than R, a contradiction. That means we always move the right pointer next, until eventually $r \leftarrow r'$ and the area of R will be computed and recorded in max-area.

(c) The idea is to split the Histogram H at the middle bar $m := \lfloor \frac{n}{2} \rfloor$ and take the maximum area of three partial results. The first is the maximum area of a rectangle in $h_1, \ldots, h_{m-1}$ the second is the maximum area intersecting bar m and the third is the maximum area rectangle in $h_{m+1}, \ldots, h_n$. As the partial results cover all possible positions of rectangles, the overall result is correct if the partial results are. The first and third result are obtained recursively, the second with our result from part (b).

Algorithm 3 `max-area-dc`$(h_1, \ldots, h_n)$

```
if n = 0 then
    return 0                                          ▷ check base cases
if n = 1 then
    return h_1
m ← ⌊n/2⌋
area1 ← max-area-dc(h_1, ..., h_{m-1})
area2 ← max-area-bar(m, h_1, ..., h_n)
area3 ← max-area-dc(h_{m+1}, ..., h_n)
return max(area1, area2, area3)
```

Running time: Our recursive function for the running time is $T(n) \leq 2T\left(\frac{n}{2}\right) + \mathcal{O}(n)$. Applying the Master Theorem given in the lecture shows that $T(n) \in \mathcal{O}(n \log n)$.