Algorithm Theory, Tutorial 1

Improving the QLever Search Engine

Johannes Kalmbach

University of Freiburg

johannes.kalmbach@gmail.com

October 2018

General Hints

- Contact tutor (`johannes.kalmbach@gmail.com`) for questions concerning corrections etc.
- Contact forum (`daphne.informatik.uni-freiburg.de`) for everything else
- Suggestion: Submit in groups of two (better for understandable algorithms)
- Submit readable solutions (LaTeX as pdf, CLEAN handwriting (+ good scan if necessary))
- Spend enough time on exercise sheets and writeup (you and I have to understand your submission).

Algorithm Writeups

- Pseudocode, limit to important aspects
- Reader must be able to understand and implement it.
- E.g "Split Array A in two evenly-sized halves L and R"

Ex 1a: Prove or disprove: $n! \in \Omega(n^2)$

Ex 1b: Prove or disprove: $\sqrt{n^3} \in O(n \log n)$

Ex 1b: Prove or disprove: $\sqrt{n^3} \in O(n \log n)$

Ex 1b: Prove or disprove: $\sqrt{n^3} \in O(n \log n)$

Ex 1c: Prove or disprove: $2^{\sqrt{\log_2 n}} \in \Theta(n)$

Ex 1c: Prove or disprove: $2^{\sqrt{\log_2 n}} \in \Theta(n)$

Ex 1c: Prove or disprove: $2^{\sqrt{\log_2 n}} \in \Theta(n)$

Sort by asymptotic growth

n^2	$\sqrt{n}$	$2^{\sqrt{n}}$	$\log(n^2)$
$2^{\sqrt{\log_2 n}}$	$\log(n!)$	$\log(\sqrt{n})$	$(\log n)^2$
$\log n$	$10^{100} n$	$n!$	$n \log n$
$2^n / n$	n^n	$\sqrt{\log n}$	n

Sort by asymptotic growth

Sort by asymptotic growth

Ex 3: Master Theorem

See master solution, just table lookup

Ex 4: Peak Elements

You are given an array $A[1 \ldots n]$ of n integers and the goal is to find a peak element, which is defined as an element in A that is equal to or bigger than its direct neighbors in the array. Formally, $A[i]$ is a peak element if $A[i-1] \leq A[i] \geq A[i+1]$. To simplify the definition of peak elements on the rims of A, we introduce *sentinal-elements* $A[0] = A[n+1] = -\infty$.

a) Give an algorithm with runtime $O(\log n)$ (measured in the number of read operations on the array) which returns the position i of a peak element.

b) Prove that your algorithm always returns a peak element, give a recurrence relation for the runtime and use it to prove the runtime.

Why/When do peak elements exist?

Claim: If $A[0] < A[1]$ and $A[n] > A[n+1]$ then $A[1 \dots n]$ contains a peak element.

Prove by contradition: Assume that $A[1 \dots n]$ contains no peak, then

Constructing the algorithm

Idea: Make the considered array smaller while ensuring that it still contains a peak element:

Constructing the algorithm

Proof of correctness

Two parts: Algorithm always terminates and only looks at arrays that contain peak elements.

Proof of correctness

Proof of correctness

Ex 5, Frequent Numbers

You are given an Array $A[0 \dots n-1]$ of n integers and the goal is to determine frequent numbers which occur at least $n/3$ times in A. There can be at most three such numbers, if any exist at all.

a. Give an algorithm with runtime $O(n \log n)$ (measured in number of array entries that are read) based on the divide and conquer principle that outputs the frequent numbers (if any exist).
b. Argue why your algorithm is correct, give a recurrence relation for the runtime and use it to prove the runtime.

First Try: HashMap

Divide and Conquer

Sorting and Counting