Exercise sheet 1.

## **Advanced Algorithms**

WiSe 2020/21

Instructor: László Kozma

**Due** 12:00, November 13th, 2020

## Exercise 1 Asymptotic notation

10 Points

The purpose of this exercise is to deepen our understanding of asymptotic notation. Revisit the definitions where necessary and answer in a precise/rigorous way.

Let f, g be almost everywhere positive functions.

- (a) (3 pts) Show that if f is a polynomial of degree k then
  - $f(n) \in \Theta(n^k)$ ,
  - $f(n) \in o(n^{\ell})$  for  $\ell > k$ ,
  - $f(n) \in \omega(n^{\ell})$  for  $\ell < k$ .
- (b) (3 pts) For numbers,  $x \not< y$  and  $x \ge y$  are equivalent. In some sense, we may view  $o(\dots)$  and  $\Omega(\dots)$  as the extensions of < and  $\ge$  to (growth rates of) functions. However,  $f \notin o(g)$  does not imply  $f \in \Omega(g)$ . Find an example that shows this.

(Bonus 2 pts) Can you give an example where both f and g are monotone increasing functions?

(c) (4 pts) Order the following 12 functions by their asymptotic growth rates, motivating your answer. By  $\epsilon$  we denote an arbitrary constant, s.t.  $0 < \epsilon < 1$ .

$$3^{\sqrt{\log_2 n}}, \quad n^3 + 2n, \quad n^{\sqrt{n}}, \quad n^{1-\epsilon}, \quad n^{5/4} \log_2 n, \quad (1+\epsilon)^n, \quad n/\log_2 \log_2 n, \quad \epsilon n,$$

$$n^{7/5}, \quad \sqrt{n^{\log_2 n}}, \quad n^{\log_2 \log_2 n}, \quad (\log_2 n)^{\log_2 n}.$$

## Exercise 2 Randomized algorithm

4 Points

Suppose we have a randomized (Monte Carlo) algorithm that returns the correct solution with probability  $\frac{1}{\ln n}$ , and suppose that we can easily verify whether the solution is correct. How many (independent) runs should we perform, to obtain the correct solution with probability at least  $(1 - \frac{1}{n^2})$ ?

The purpose of this exercise is to show that a computational model with unit cost operations and no restriction on the word sizes would be too powerful and unrealistic. In particular, you are to show that the problem of factoring a large integer into its prime components would, in such a machine, become easy. This problem is believed to be difficult, and many widely used methods in cryptography (e.g. the RSA system) rely on this assumption.

Suppose (for the purpose of the exercise) an integer RAM with no word-size restriction and unit cost operations  $+, -, \times, div$ .

- (a) (2 pts) Show that given integers A and N the number  $A^N$  can be computed in  $O(\log N)$  time.
- (b) (2 pts) Show that given natural numbers N and K the binomial coefficient  $\binom{N}{K}$  can be computed in  $O(\log N)$  time. Hint: Consider  $(A+1)^N$  for large A.
- (c) (2 pts) Show that given natural number N the number N! can be computed in  $O(\log^2 N)$  time.

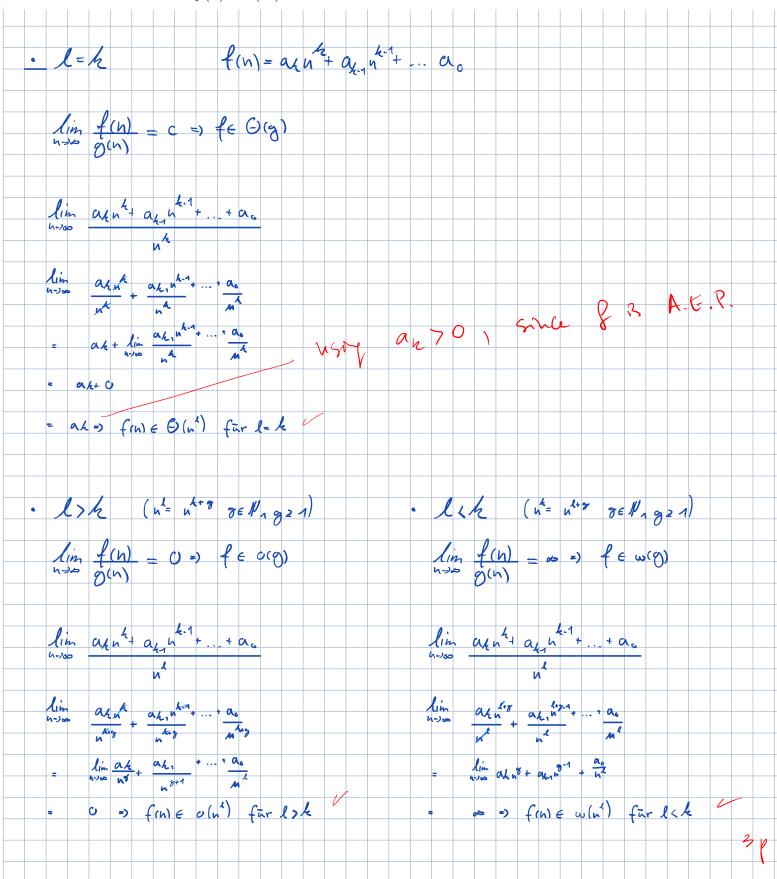
  Hint: Recursion.
- (d) (2 pts) Show that in  $O(\log^2 N)$  time it can be tested whether N is a prime number. Hint: Consider whether N divides (N-1)!.
- (e) (2 pts) Show that in  $O(\log^3 N)$  time a non-trivial factor of N can be found, provided N is not prime. For this you may assume the existence of a routine that computes the GCD (Greatest Common Divisor) of two numbers X and Y in time  $O(\log(\min\{X,Y\}))$ .
- (f) (2 pts) Show that the prime factorization of N can be found in time  $O(\log^4 N)$ .
- (g) (Bonus 2 pts) Can you improve some of the indicated asymptotic running times?

Total: 26 points. Have fun with the solutions!

The purpose of this exercise is to deepen our understanding of asymptotic notation. Revisit the definitions where necessary and answer in a precise/rigorous way.

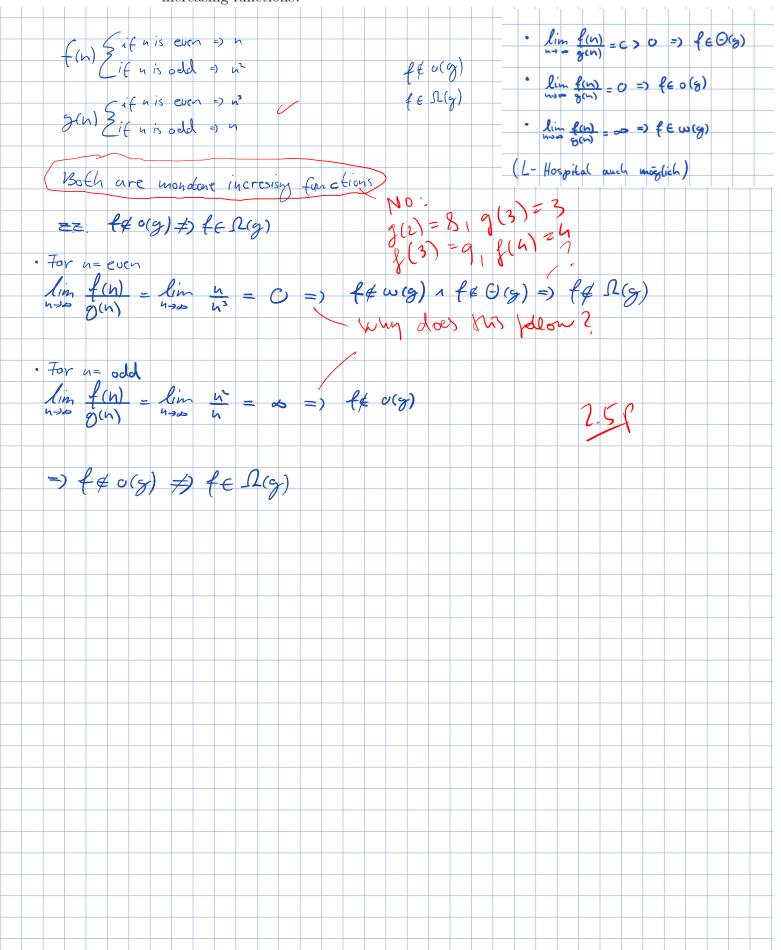
Let f, g be almost everywhere positive functions.

- (a) (3 pts) Show that if f is a polynomial of degree k then
  - $f(n) \in \Theta(n^k)$ ,
  - $f(n) \in o(n^{\ell})$  for  $\ell > k$ ,
  - $f(n) \in \omega(n^{\ell})$  for  $\ell < k$ .

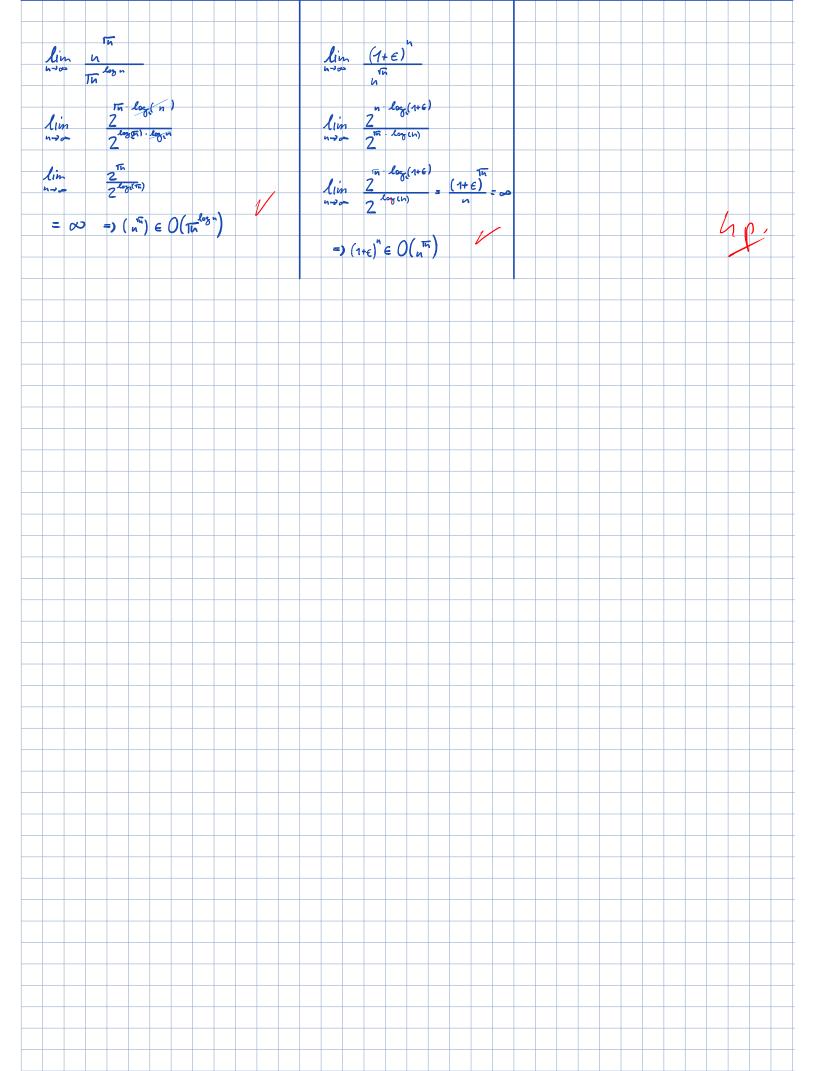


(b) (3 pts) For numbers,  $x \not< y$  and  $x \ge y$  are equivalent. In some sense, we may view  $o(\dots)$  and  $\Omega(\dots)$  as the extensions of < and  $\ge$  to (growth rates of) functions. However,  $f \notin o(g)$  does not imply  $f \in \Omega(g)$ . Find an example that shows this.

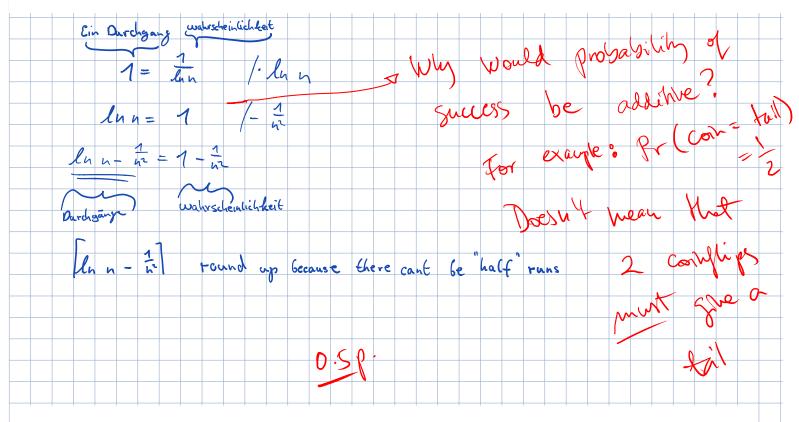
(Bonus 2 pts) Can you give an example where both f and g are monotone increasing functions?



(c) (4 pts) Order the following 12 functions by their asymptotic growth rates, motivating your answer. By $\epsilon$ we denote an arbitrary constant, s.t. $0 < \epsilon < 1$ . $3^{\sqrt{\log_2 n}},  n^3 + 2n,  n^{\sqrt{n}},  n^{1-\epsilon},  n^{5/4} \log_2 n,  (1+\epsilon)^n,  n/\log_2 \log_2 n,  \epsilon n,$ From 6ig 60 small $n^{7/5},  \sqrt{n^{\log_2 n}},  n^{\log_2 \log_2 n},  (\log_2 n)^{\log_2 n}.$	
From big to small $n^{7/5}$ , $\sqrt{n}^{\log_2 n}$ , $n^{\log_2 \log_2 n}$ , (log	$(g_2 n)^{\log_2 n}$ .
$C < \alpha < \beta $ $O < \alpha < b$ $log_n n \in G(log_n)$ $1 < A < \beta$ $A < \beta < (log_n) $	K' K K' np k'n
lin N log log N lin 3	lin n = lin n · log.log.n
$\lim_{n\to\infty} \frac{2^{\log n} \log_2 n}{2^{\log n} \log_2 n} = \lim_{n\to\infty} \frac{3^{\log n} n}{3^{\log n} (n) \cdot (1-\epsilon)}$ $\lim_{n\to\infty} \frac{2^{\log n} \log_2 n}{2^{\log n} \log_2 n} \cdot \log_2 (n) = 1$ $\lim_{n\to\infty} \frac{2^{\log n} \log_2 n}{2^{\log n} \log_2 n} \cdot \log_2 (n)$	( n=e; O(loglogin))  = 0 => n=e; o(Vloglogin)
= = = 3 3 tojin & o (n=6)  = nogelyen & G (logen logen)  lin ne noon //logelogen	lin No 195
lin KE · log. log. n  lin E · log. log. n  lin E · log. log. n  = 0	lin no logue  lin logue = 0
= = = ) NE E O( 1/leg-leg-n) = ) NE E O( 1/leg-leg-n)	=) n'lyn e o (n')
lin n + 2n  1 1 1 1 2n  1 2 2n	lin 2  log 2(In) · log in  lin 2  log 2(In) · log in
(din $3 < l_{g_1 l_{g_1 n}}) \Rightarrow (n^3 \stackrel{?}{\leftarrow} o(n^{l_{g_1 l_{g_2 n}}}))$ (lin $n^{\frac{r_1}{r_1}} \stackrel{?}{\rightarrow} = \infty$ $(2n \stackrel{?}{\leftarrow} o(n^{l_{g_1 l_{g_2 n}}}))$	lin 2 logita) - lin In  no 2 logitagin ho login  (In & O(login) = 10 E O(n logitagin)



Suppose we have a randomized (Monte Carlo) algorithm that returns the correct solution with probability  $\frac{1}{\ln n}$ , and suppose that we can easily verify whether the solution is correct. How many (independent) runs should we perform, to obtain the correct solution with probability at least  $(1-\frac{1}{n^2})$ ?



Exercise 3 RAM model

12 Points

The purpose of this exercise is to show that a computational model with unit cost operations and no restriction on the word sizes would be too powerful and unrealistic. In particular, you are to show that the problem of factoring a large integer into its prime components would, in such a machine, become easy. This problem is believed to be difficult, and many widely used methods in cryptography (e.g. the RSA system) rely on this assumption.

Suppose (for the purpose of the exercise) an integer RAM with no word-size restriction and unit cost operations  $+, -, \times, div$ .

- (a) (2 pts) Show that given integers A and N the number  $A^N$  can be computed in  $O(\log N)$  time.
- - (d) (2 pts) Show that in  $O(\log^2 N)$  time it can be tested whether N is a prime Hint: Consider whether N divides (N-1)!.
  - (e) (2 pts) Show that in  $O(\log^3 N)$  time a non-trivial factor of N can be found, provided N is not prime. For this you may assume the existence of a routine that computes the GCD (Greatest Common Divisor) of two numbers X and Y in time  $O(\log(\min\{X,Y\}))$ .
  - (f) (2 pts) Show that the prime factorization of N can be found in time  $O(\log^4 N)$ .
  - (g) (Bonus 2 pts) Can you improve some of the indicated asymptotic running times?

