From: Jonny Lam & Thore Brehmer

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 $x \geq y$ are equivalent. In some sense, we may view $o(\dots)$ and $\Omega(\dots)$ as the extensions of $<$ and $\geq$ 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 ϵ 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 $(1 - \frac{1}{n^2})$?

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.

(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!

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 $\ell < k$.

• $l = k$ $\qquad f(n) = a_k n^k + a_{k-1} n^{k-1} + \dots a_0$

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = c \Rightarrow f \in \Theta(g)$$

$$\lim_{n\to\infty} \frac{a_k n^k + a_{k-1} n^{k-1} + \dots + a_0}{n^k}$$

$$\lim_{n\to\infty} \frac{a_k n^k}{n^k} + \frac{a_{k-1} n^{k-1}}{n^k} + \dots + \frac{a_0}{n^k}$$

$$= a_k + \lim_{n\to\infty} \frac{a_{k-1} n^{k-1}}{n^k} + \dots + \frac{a_0}{n^k}$$

$$= a_k + 0$$

$$= a_k \Rightarrow f(n) \in \Theta(n^l) \text{ für } l = k$$ ✓

using $a_k > 0$, since f is A.E.P.

• $l > k$ $\quad (n^l = n^{k+g} \quad g \in \mathbb{N}, g \geq 1)$

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0 \Rightarrow f \in o(g)$$

$$\lim_{n\to\infty} \frac{a_k n^k + a_{k-1} n^{k-1} + \dots + a_0}{n^l}$$

$$\lim_{n\to\infty} \frac{a_k n^k}{n^{k+g}} + \frac{a_{k-1} n^{k-1}}{n^{k+g}} + \dots + \frac{a_0}{n^{k+g}}$$

$$= \lim_{n\to\infty} \frac{a_k}{n^g} + \frac{a_{k-1}}{n^{g+1}} + \dots + \frac{a_0}{n^l}$$

$$= 0 \Rightarrow f(n) \in o(n^l) \text{ für } l > k$$ ✓

• $l < k$ $\quad (n^k = n^{l+g} \quad g \in \mathbb{N}, g \geq 1)$

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty \Rightarrow f \in \omega(g)$$

$$\lim_{n\to\infty} \frac{a_k n^k + a_{k-1} n^{k-1} + \dots + a_0}{n^l}$$

$$\lim_{n\to\infty} \frac{a_k n^{l+g}}{n^l} + \frac{a_{k-1} n^{l+g-1}}{n^l} + \dots + \frac{a_0}{n^l}$$

$$= \lim_{n\to\infty} a_k n^g + a_{k-1} n^{g-1} + \frac{a_0}{n^l}$$

$$= \infty \Rightarrow f(n) \in \omega(n^l) \text{ für } l < k$$ ✓

3P

(b) (3 pts) For numbers, $x \not< y$ and $x \geq y$ are equivalent. In some sense, we may view $o(\dots)$ and $\Omega(\dots)$ as the extensions of $<$ and $\geq$ 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?

$$f(n) \begin{cases} \text{if } n \text{ is even} \Rightarrow n \\ \text{if } n \text{ is odd} \Rightarrow n^2 \end{cases}$$

$$g(n) \begin{cases} \text{if } n \text{ is even} \Rightarrow n^3 \\ \text{if } n \text{ is odd} \Rightarrow n \end{cases}$$

✓

$f \notin o(g)$
$f \in \Omega(g)$

- $\lim_{n \to \infty} \frac{f(n)}{g(n)} = c > 0 \Rightarrow f \in \Theta(g)$
- $\lim_{n \to \infty} \frac{f(n)}{g(n)} = 0 \Rightarrow f \in o(g)$
- $\lim_{n \to \infty} \frac{f(n)}{g(n)} = \infty \Rightarrow f \in \omega(g)$

(L-Hospital auch möglich)

Both are monotone incresing functions

NO:
$g(2) = 8,\ g(3) = 3$
$f(3) = 9,\ f(4) = 4$
?

zz. $f \notin o(g) \not\Rightarrow f \in \Omega(g)$

- For $n =$ even

$$\lim_{n \to \infty} \frac{f(n)}{g(n)} = \lim_{n \to \infty} \frac{n}{n^3} = 0 \Rightarrow f \notin \omega(g) \wedge f \notin \Theta(g) \Rightarrow f \notin \Omega(g)$$

why does this follow?

- For $n =$ odd

$$\lim_{n \to \infty} \frac{f(n)}{g(n)} = \lim_{n \to \infty} \frac{n^2}{n} = \infty \Rightarrow f \notin o(g)$$

2.5P

$$\Rightarrow f \notin o(g) \not\Rightarrow f \in \Omega(g)$$

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

$$\overset{11}{3^{\sqrt{\log_2 n}}},\quad \overset{5}{n^3 + 2n},\quad \overset{2}{n^{\sqrt{n}}},\quad \overset{10}{n^{1-\epsilon}},\quad \overset{7}{n^{5/4}\log_2 n},\quad \overset{1}{(1+\epsilon)^n},\quad \overset{9}{n/\log_2\log_2 n},\quad \overset{8}{\epsilon n},$$

$$\overset{6}{n^{7/5}},\quad \overset{3}{\sqrt{n}^{\log_2 n}},\quad \overset{4}{n^{\log_2\log_2 n}},\quad \overset{4}{(\log_2 n)^{\log_2 n}}.$$

From big to small
1, 2, ... 11

Very good.

$0 < \alpha < \beta$
$0 < a < b$
$1 < A < B$

$a, b > 1$
$\log_a n \in \Theta(\log_b n)$

$$A \overset{1}{\text{"}\prec\text{"}} (\log n)^\alpha \overset{2}{\text{"}\prec\text{"}} (\log n)^\beta \overset{3}{\text{"}\prec\text{"}} n^a \overset{4}{\text{"}\prec\text{"}} n^a (\log n)^\alpha \overset{5}{\text{"}\prec\text{"}} n^b \overset{6}{\text{"}\prec\text{"}} A^n \overset{7}{\text{"}\prec\text{"}} n^a A^n \overset{8}{\text{"}\prec\text{"}} B^n \overset{9}{\text{"}\prec\text{"}} n! \overset{10}{\text{"}\prec\text{"}} n^n$$

$$\lim_{n\to\infty} \frac{n^{\log_2\log_2 n}}{(\log_2 n)^{\log_2 n}}$$

$$\lim_{n\to\infty} \frac{2^{\log(n)^{\log_2\log_2 n}}}{2^{\log(\log_2 n)^{\log_2 n}}}$$

$$\lim_{n\to\infty} \frac{2^{\log_2\log_2 n \cdot \log_2(n)}}{2^{\log_2\log_2 n \cdot \log_2(n)}} = 1$$

$$\Rightarrow n^{\log_2\log_2 n} \in \Theta(\log_2 n^{\log_2 n})$$

$$\lim_{n\to\infty} \frac{3^{\sqrt{\log_2 n}}}{n^{1-\epsilon}}$$

$$\lim_{n\to\infty} \frac{3^{\sqrt{\log_2 n}}}{3^{\log_3(n)\cdot(1-\epsilon)}}$$

$$\left(\sqrt{\log_2 n} \in o(\log_3(n)\cdot(1-\epsilon))\right)$$

$$= 0 \Rightarrow 3^{\sqrt{\log_2 n}} \in o(n^{1-\epsilon})$$

$$\lim_{n\to\infty} \frac{n^{1-\epsilon}}{n/(\log_2\log_2 n)} = \lim_{n\to\infty} \frac{\cancel{n^{1-\epsilon}} \cdot \log_2\log_2 n}{n}$$

$$\lim_{n\to\infty} \frac{\log_2\log_2 n}{n^{1-\epsilon}}$$

should be n^ϵ

$$\left(n^{1-\epsilon} \in O(\log_2\log_2 n)\right)$$

$$= 0 \Rightarrow n^{1-\epsilon} \in o(n/\log_2\log_2 n)$$

$$\lim_{n\to\infty} \frac{n\epsilon}{n/\log_2\log_2 n}$$

$$\lim_{n\to\infty} \frac{\cancel{n}\epsilon \cdot \log_2\log_2 n}{\cancel{n}}$$

$$\lim_{n\to\infty} \epsilon \cdot \log_2\log_2 n$$

$$= \infty \Rightarrow n\epsilon \in O(n/\log_2\log_2 n)$$

$$\lim_{n\to\infty} \frac{\cancel{n}\epsilon}{n^{\cancel{5/4}}\log_2 n}$$

$$\lim_{n\to\infty} \frac{\epsilon}{n^{1/4}\log_2 n}$$

$$= 0$$

$$\Rightarrow n\epsilon \in o(n^{5/4}\log_2 n)$$

$$\lim_{n\to\infty} \frac{n^{\cancel{5/4}}\log_2 n}{n^{\cancel{7/5}}}$$

$$\lim_{n\to\infty} \frac{n^{\cancel{5/20}}\log_2 n}{n^{\cancel{8/20}}}$$

$$\lim_{n\to\infty} \frac{\log_2 n}{n^{3/20}} = 0$$

$$\Rightarrow n^{5/4}\log_2 n \in o(n^{7/5})$$

$$\lim_{n\to\infty} \frac{n^3 + 2n}{n^{7/5}}$$

$$\lim_{n\to\infty} \frac{n^{15/5} + 2n^{5/5}}{n^{7/5}}$$

$$\lim_{n\to\infty} n^{8/5} + \frac{2}{n^{2/5}} = \infty$$

$$\Rightarrow n^3 + 2n \in O(n^{7/5})$$

$$\lim_{n\to\infty} \frac{n^3 + 2n}{n^{\log_2\log_2 n}}$$

$$\underbrace{\lim_{n\to\infty} \frac{n^3}{n^{\log_2\log_2 n}}} + \underbrace{\lim_{n\to\infty} \frac{2n}{n^{\log_2\log_2 n}}}$$

$$\left(\lim_{n\to\infty} 3 < \log_2\log_2 n\right) \Rightarrow \left(n^3 \in o(n^{\log_2\log_2 n})\right)$$

$$\left(2n \in o(n^{\log_2\log_2 n})\right)$$

$$0 \quad + \quad 0$$

$$= 0 \Rightarrow n^3 + 2n \in o(n^{\log_2\log_2 n})$$

$$\lim_{n\to\infty} \frac{\sqrt{n}^{\log_2 n}}{n^{\log_2\log_2 n}}$$

$$\lim_{n\to\infty} \frac{2^{\log_2(\sqrt{n})\cdot\cancel{\log_2 n}}}{2^{\cancel{\log_2 n}\cdot\log_2\log_2 n}}$$

$$\lim_{n\to\infty} \frac{2^{\log_2(\sqrt{n})}}{2^{\log_2\log_2 n}} = \lim_{n\to\infty} \frac{\sqrt{n}}{\log_2 n} =$$

$$\left(\sqrt{n} \in O(\log_2 n)\right)$$

$$= \infty$$

$$\Rightarrow \sqrt{n}^{\log_2 n} \in O(n^{\log_2\log_2 n})$$

$$\lim_{n\to\infty} \frac{n^{\sqrt{n}}}{\sqrt{n}^{\log n}}$$

$$\lim_{n\to\infty} \frac{2^{\sqrt{n}\cdot\log_2(n)}}{2^{\log_2(\sqrt{n})\cdot\log_2 n}}$$

$$\lim_{n\to\infty} \frac{2^{\sqrt{n}}}{2^{\log_2(\sqrt{n})}}$$

$$= \infty \Rightarrow (n^{\sqrt{n}}) \in O(\sqrt{n}^{\log n})$$

$$\lim_{n\to\infty} \frac{(1+\epsilon)^{n}}{n^{\sqrt{n}}}$$

$$\lim_{n\to\infty} \frac{2^{n\cdot\log_2(1+\epsilon)}}{2^{\sqrt{n}\cdot\log(n)}}$$

$$\lim_{n\to\infty} \frac{2^{\sqrt{n}\cdot\log_2(1+\epsilon)}}{2^{\log(n)}} = \frac{(1+\epsilon)^{\sqrt{n}}}{n} = \infty$$

$$\Rightarrow (1+\epsilon)^{n} \in O(n^{\sqrt{n}})$$

4 P.

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 $(1 - \frac{1}{n^2})$?

$$\overbrace{1}^{\text{Ein Durchgang}} = \overbrace{\frac{1}{\ln n}}^{\text{wahrscheinlichkeit}} \quad / \cdot \ln n$$

$$\ln n = 1 \quad / - \frac{1}{n^2}$$

$$\underbrace{\ln n - \frac{1}{n^2}}_{\text{Durchgänge}} = \underbrace{1 - \frac{1}{n^2}}_{\text{wahrscheinlichkeit}}$$

$\left\lceil \ln n - \frac{1}{n^2} \right\rceil$ round up because there cant be "half" runs

Why would probability of success be additive?

For example: $Pr(\text{coin} = \text{tail}) = \frac{1}{2}$

Doesn't mean that 2 coinflips must give a tail

0.5P.

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.

$5! = \binom{5}{3} 3! \cdot 2!$

(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.

$N! = \binom{N}{k} \cdot k \cdot (n-k)!$

(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?

a)

```
def pow(a,n):

    temp = pow(a, n div 2)        O(Log(n))

    if (n==0):                    O(1)
        return 1

    elif (n mod 2 == 0):          O(1)
        return temp*temp

    else:                         O(1)
        return x*temp*temp
```

✓ 2P

b) $(A+1)^N = \sum_{k=0}^{N} A^k \cdot \binom{N}{k}$ choose $A = 2^n$ $\Rightarrow (2^n+1)^n = \sum_{k=0}^{N} 2^{nk} \cdot \binom{n}{k}$

$(2^n+1)^n$ can be computed in $O(\log(n))$. Store the result in one cell.

To find $\binom{N}{k}$ use binsearch $(O(\log(n)))$ and search in the cell for position 2^{nk} to $2^{nk+1}-1$. Read all the bits between this position. (Exeption $\binom{n}{0}$ are not defined)

Why need binary search? just read out using div and mod.

E.g. $(2^3+1)^3 = 729$

$\binom{3}{3}=1$ $\binom{3}{2}=3$ $\binom{3}{1}=3$ $\left(\binom{3}{0}=1\right)$

Binary: 0 0 0 1 0 1 1 0 1 1 0 0 1

$2^{3\cdot 4}$ $2^{3\cdot 3}$ $2^{3\cdot 2}$ $2^{3\cdot 1}$ $2^{3\cdot 0}$

E.g. $(2^4+1)^4 = 83521$

$\binom{4}{4}=1$ $\binom{4}{3}=4$ $\binom{4}{2}=6$ $\binom{4}{1}=4$

Binary: 0 0 0 1 0 1 0 0 0 1 1 0 0 1 0 0 0 0 1

$2^{4\cdot 5}$ $2^{4\cdot 4}$ $2^{4\cdot 3}$ $2^{4\cdot 2}$ $2^{4\cdot 1}$ $2^{4\cdot 0}$

1.5P

c) def fac (n):

$N! = \binom{N}{\frac{N}{2}} \cdot \frac{N}{2}! \cdot (n-\frac{N}{2})! = \binom{N}{\lceil\frac{N}{2}\rceil} \cdot \lfloor\frac{N}{2}\rfloor! \cdot \lceil\frac{N}{2}\rceil!$

```
if ( n== 1 ):
    return 1
```

$O(1)$

```
else:
    return (n [n/2]) · fac([n/2]) · fac(n-[n/2])
```

$O(\log_2 n)$

$\Rightarrow O(\log_2 n)$

1.5 ~~P~~

Algorithm is OK but why $O(\lg n)$?

Even computing $\binom{n}{\frac{n}{2}}$ takes $O(\lg n)$.

Should be $O(\lg^2 n)$ overall.