KAIST School of Computing 291 Daehak-ro, Yuseong-gu, Daejeon 305-701, Republic of Korea
Tel. +82-42-350-3502~3507 Fax. +82-42-350-3510

Prof. Dr. Martin Ziegler
+82-42-350-3568
ziegler@kaist.ac.kr
http://m.zie.de

CS500 Jul 7, 2017 PhD Qualifying Exam

You are allowed to bring and peruse a print copy of *Introduction to Algorithms* by Cormen, Leiserson, Rivest (Stein) and a two-sided A5-size paper with own handwritten notes.

30 points = 120%, 15 points=pass

Assignment 1 (3+ 3+ 3 points):

a) Explain in few sentences, and give examples demonstrating, the differences between (i) a program/code, (ii) an algorithm, and (iii) a heuristic.

b) Briefly explain the difference between the (i) cost of an algorithm and (ii) computational complexity of a problem. What does it mean for an algorithm to (iii) incur *optimal* cost?

c) Explain briefly the following notions of algorithmic cost analyses and their differences: i) worst-case runtime, ii) asymptotic worst-case runtime, iii) asymptotic worst-case memory consumption, iv) asymptotic average-case runtime, v) asymptotic amortized runtime, vi) asymptotic expected runtime.

Assignment 2 (4+ 1+ 1+ 4+ 1 points):

a) Specify the four (different?) problems solved by the following four algorithms, and report (without proofs) their asymptotic worst-case costs: (i) BubbleSort, (ii) QuickSort, (iii) RadixSort, (iv) CountingSort.

Consider the following algorithm **search**, searching for an integer M between 0 and N-1 such that subroutine call **verify(M)** returns **true**:

```
int search(int N, int verify(int))
for (int M=0; M<N; M++)
  if (verify(M)) return(1);
return(0);
```

Suppose that, for m the binary length of M, "**verify(M)**" uses time $t(m)$ and memory $s(m)$.

b) Analyze the worst-case runtime of **search(N)**, asymptotically in the bin. length n of N.

c) Analyze the worst-case memory use of **search(N)**, asymptotically as $n\to\infty$. Justify!

d) Recall that *Ackermann's Function* is defined recursively as

$$A_0(n) = n+2, \qquad A_{k+1}(0) = A_k(1), \qquad A_{k+1}(n+1) = A_k(A_{k+1}(n))$$

Prove by induction: (i) $A_1(n)=2n+3$ and (ii) $A_2(n)=2^{n+3}-3$

e) Determine $\log^*(n)$ for $n:=2^{80}$, the purported number of electrons in the universe.

Recall: $\log^*$ denotes the inverse of *tetration*, $\log^*(1)=0$, $\log^*(n) = 1 + \log^*(\log_2 n)$ for $n>1$.

KAIST School of Computing 291 Daehak-ro, Yuseong-gu, Daejeon 305-701, Republic of Korea
Tel. +82-42-350-3502~3507 Fax. +82-42-350-3510
Prof. Dr. Martin Ziegler

Problem 3 (2+2+2+3+1 points): Suppose $\mathcal{A}$ is a randomized algorithm solving the decision problem L with *one*-sided error 0.999: On inputs $\underline{x}\notin L$, $\mathcal{A}$ always correctly reports **`false`**; but on inputs $\underline{x}\in L$, $\mathcal{A}$ might also report **`false`** with probability as high as 0.999.

a) Design and analyze an algorithm $\mathcal{A}'$ that errs only with probability $\leq 2^{-N}$. Hint: $(1-{}^{1}/_{k})^{k}\leq ½$.

b) Analyze the (i) *worst* and (ii) *expected* cost of the following randomized 'algorithm':

Flip a fair coin. If it comes out ***heads***, stop; otherwise repeat.

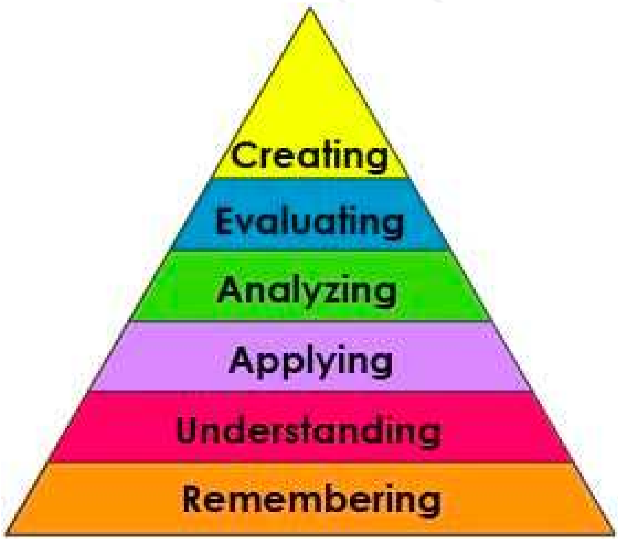
c) Analyze the *worst*-case cost of the following algorithm, asymptotically as $n\to\infty$:

Given an n-tuple $(b_1,\ldots,b_n)$ of bits, scan for the (index j of the) first non-zero bit. If all bits are zero, count to 2^n-1. Then stop.

d) Analyze the *average*-case cost of the algorithm from c), asymptotically as $n\to\infty$.

e) Prove $\sum_n n\cdot p^n = p/(1-p)^2$ for all $|p|<1$. **Hint:** Cancel one p and compare anti-derivatives.

This being an *open book* exam, your answers to problems 2b, 2c, 2d, 2e, and 3 must include full justification/proofs to earn full points: Mere claim/reference does not suffice!



Bloom's Taxonomy of Cognitive Learning