Exam in Algorithms & Data Structures 3 (1DL481)

Prepared by Pierre Flener

Wednesday 16 March 2022 from 14:00 to 17:00 in the Polacksbacken exam hall

Materials: This is a ***closed***-book exam, drawing from the book *Introduction to Algorithms* by T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein, published in 3rd edition by the MIT Press in 2009, and denoted by CLRS3 here. The usage of electronic devices is ***not*** allowed.

Instructions: Question 1 is ***mandatory***: you must earn ***at least half*** its points in order to pass the exam (see **Grading** below). Start each answer on a new sheet, and indicate there the question number. Your answers must be written in English. Unreadable, unintelligible, and irrelevant answers will not be considered. Provide only the requested information and nothing else, but always show ***all*** the details of your reasoning, unless explicitly not requested, and make explicit ***all*** your assumptions. Do ***not*** write anything into the following table:

Question	Max Points	Your Mark
1	7	
2	3	
3	4	
4	6	
Total	20	

Help: Unfortunately, no teacher can attend the exam.

Grading: Your grade is as follows when your exam mark is e points, including ***at least*** 3.5 points on Question 1:

Grade	Condition
5	$18 \leq e \leq 20$
4	$14 \leq e \leq 17$
3	$10 \leq e \leq 13$
U	$00 \leq e \leq 09$

Identity: Your anonymous exam code:

					–					–			

Question 1: NP-Completeness (*mandatory question*!) (7 points)

Problem 35-1a: ***Bin packing.*** Suppose that we are given a set of m objects, where the size s_i of the ith object satisfies $0 < s_i < 1$. We wish to pack all the objects into the minimum number of unit-size bins. Each bin can hold any subset of the objects whose total size does not exceed 1. Prove that the problem of determining the minimum number of bins required is NP-hard, by using a *single* reduction, *directly* from the ***subset-sum problem***, which, given a finite set X of n positive integers x_i and an integer target $t > 0$, asks whether there exists a subset $X' \subseteq X$ whose elements sum to t; for example, if $X = \{7, 9, 8\}$ and $t = 15$, then the subset $X' = \{8, 7\}$ is a solution, but there is no solution for $t = 14$.

You must earn ***at least half*** the points. Start your answer on a new sheet. You ***must*** use the identifiers m, s_i, X, n, x_i, t, X' of the question.

Question 2: Randomised Algorithms (3 points)

Exercise 5.3-3: Many randomised algorithms randomise the input by permuting an input array. Suppose that instead of swapping element $A[i]$ with a random element from the subarray $A[i\,..\,n]$, we swapped it with a random element from anywhere in the array:

PERMUTE-WITH-ALL(A)

```
1  n = A.length
2  for i = 1 to n            // RANDOM(ℓ, u) returns a random number from ℓ to u inclusive
3      swap A[i] with A[RANDOM(1, n)]
```

Does this code produce a uniform random permutation? Why or why not?

Start your answer on a new sheet.

Question 3: Amortised Analysis (4 points)

Exercise 17.3-2: Suppose we perform a sequence of n operations on a data structure in which the ith operation costs i if i is an exact power of 2, and 1 otherwise. Use a *potential method of analysis* (also known as the physicist's method) to determine the amortised cost per operation.

Start your answer on a new sheet.

Question 4: Approximation Algorithms (6 points)

Exercise 35.4-2: The input to the ***MAX-3-CNF satisfiability problem*** is the same as for 3-CNF satisfiability (which asks whether a conjunction of clauses, each of exactly three distinct literals, is satisfiable; a *literal* is an occurrence of a Boolean variable or its negation), and the goal is to return an assignment of the variables that maximises the number of satisfied clauses. The ***MAX-CNF satisfiability problem*** is like the MAX-3-CNF satisfiability problem, except that it does not restrict each of the m clauses C_i to have exactly 3 literals. Give a randomised 2-approximation algorithm for the MAX-CNF satisfiability problem, upon denoting its produced number of satisfied clauses by M and the number of satisfiable clauses by M^*. (The Princeton slides would call it a 1/2-approximation.) A high-level argument for polynomial time suffices.

Start your answer on a new sheet. You ***must*** use the identifiers m, C_i, M, M^* of the question and state whether you use the CLRS3 or Princeton convention for maximisation problems.