Exam in Algorithms & Data Structures 3 (1DL481)

Prepared by Pierre Flener

Monday 16 March 2020 from 14:00 to 17:00, at Bergsbrunnagatan 15, Sal 2

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. Differences from and additions to the CLRS3 version are typeset between square brackets: [...]. 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. Start each answer on a new sheet of paper, and indicate there the question (and exercise) 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	6	
4	4	
Total	20	

Help: Unfortunately, no teacher can attend this 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: Anonymous exam code (***no*** dashes!):

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

Do ***one*** of the following exercises from CLRS3 and earn ***at least half*** its points:

A. Exercise 34.5-2: Given an integer $m \times n$ matrix A and an integer m-vector b, the ***0-1 integer-programming problem*** asks whether there exists an integer n-vector x with elements in the set $\{0, 1\}$ such that $A\,x \leq b$. Prove that 0-1 integer programming is NP-complete. (*Hint*: Reduce from 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.)

B. Problem 35-1a: ***Bin packing.*** Suppose that we are given a set of n 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. (*Hint*: Reduce from the ***subset-sum problem***, which, given a finite set X of positive integers 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 $t = 14$.)

Start your answer on a new sheet of paper, and indicate there the chosen exercise (1A or 1B).

Question 2: Probabilistic Analysis, Randomised Algorithms (3 points)

Do ***one*** of the following exercises from CLRS3:

A. Exercise 5.2-4: Use indicator random variables to solve the following problem, which is known as the ***hat-check problem***. Each of n customers gives a hat to a hat-check person at a restaurant. The hat-check person gives the hats back to the customers in a random order. What is the expected number of customers who get back their own hat? Are your indicator random variables independent?

B. Exercise 5.3-4: Professor Armstrong suggests the following procedure for generating a uniform random permutation [of its input array]:

```
PERMUTE-BY-CYCLIC(A)
1  n = A.length
2  let B[1..n] be a new array
3  offset = RANDOM(1, n)        // RANDOM(ℓ, u) returns a random number from ℓ to u
4  for i = 1 to n
5       dest = i + offset
6       if dest > n
7            dest = dest − n
8       B[dest] = A[i]
9  return B
```

[What is the probability for] each element $A[i]$ of winding up in any particular position in B? [Is] the resulting permutation uniformly random?

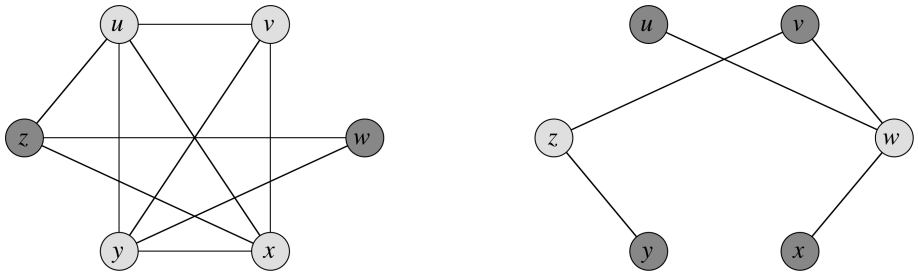
Start your answer on a new sheet of paper, and indicate there the chosen exercise (2A or 2B).

Question 3: Approximation Algorithms (6 points)

Do ***one*** of the following exercises from CLRS3:

A. Exercise 35.1-5: [First consider some reminders.] For an undirected graph $G = (V, E)$:

- A ***clique*** in G is a subset $V' \subseteq V$ of vertices, each pair of which is connected by an edge in E. The ***size*** of a clique is the number of vertices it contains. The ***clique problem*** is the optimisation problem of finding a clique of maximum size in a graph. [For example, a max-size clique of the graph in Figure (a) is $\{u, v, x, y\}$, of size 4.]
- A ***vertex cover*** of G is a subset $V' \subseteq V$ such that if $(u, v) \in E$, then $u \in V'$ or $v \in V'$ (or both). The ***size*** of a vertex cover is the number of vertices in it. The ***vertex-cover problem*** is to find a vertex cover of minimum size in a graph. [For example, a min-size vertex cover of the graph in Figure (b) is $\{w, z\}$, of size 2.] Note that there is a polynomial-time 2-approximation algorithm for the vertex-cover problem [and recall that we saw a polynomial-time 2-approximation algorithm (based on LP rounding) for the *weighted* vertex-cover problem, both analyses being tight].
- The ***complement*** of G is $\overline{G} = (V, \overline{E})$, where $\overline{E} = \{(u, v) : u, v \in V, u \neq v, (u, v) \notin E\}$. [For example, the graphs in the following figure are each other's complements:]



(a) (b)

[Now comes the exam question itself:] From the proof [by reduction from a decision version of the clique problem] of Theorem 34.12 [which states that a decision version of the vertex-cover problem is NP-complete], we know that the vertex-cover problem and the clique problem are complementary in the sense that [a minimum-size] vertex cover is the complement of a maximum-size clique in the complement graph. [For example, the minimum-size vertex cover $\{w, z\}$ of the graph in Figure (b) is the complement $V \setminus V'$ of the maximum-size clique $V' = \{u, v, x, y\}$ in the complement graph in Figure (a).] Does this relationship imply that there is a polynomial-time approximation algorithm with a constant approximation ratio for the clique problem? Justify your answer [and state whether you use the CLRS3 or Princeton convention for maximisation problems].

B. 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 MAX-3-CNF satisfiability, except that it does not restrict each clause to have exactly 3 literals. Give a randomised 2-approximation algorithm for MAX-CNF satisfiability. [A high-level argument for polynomial time suffices. The Princeton slides would call it a 1/2-approximation.]

Start your answer on a new sheet of paper, and indicate there the chosen exercise (3A or 3B).

Question 4: Amortised Analysis (4 points)

Do ***one*** of the following exercises from CLRS3:

A. Exercise 17.2-1: Suppose we perform a sequence of stack operations (PUSH or POP) on a stack whose size [somehow] never exceeds k. After every k operations, [a COPY operation is invoked automatically to] make a copy of the entire stack for backup purposes[, but without emptying the stack]. Use an *accounting method of analysis* to show that the cost of n stack operations, including copying the stack, is $\mathcal{O}(n)$ by assigning suitable amortised costs to the various stack operations.

B. 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* to determine the amortised cost per operation.

Start your answer on a new sheet of paper, and indicate there the chosen exercise (4A or 4B).