Exam in Algorithms & Data Structures 3 (1DL481)

Prepared by Pierre Flener

Thursday 21 March 2019 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 below. Differences or additions to the CLRS3 formulation 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. 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	6	
3	3	
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: Your anonymous exam code:

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

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

A. 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 $X = \{1, 2, 7, 32, 56, 35, 234, 12332\}$ and $t = 299$, then $X' = \{2, 7, 56, 234\}$ is a solution, but there is no solution for $t = 11$.)

B. Exercise 34.5-5: The ***set-partition problem*** [which we used to show that the load-balancing problem is NP-hard] takes as input a [finite] set S of [positive integers]. The question is whether the numbers can be partitioned into two sets A and $\overline{A} = S - A$ such that $\sum_{x \in A} x = \sum_{x \in \overline{A}} x$. Show that the set-partition problem is NP-complete.

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

Question 2: Approximation Algorithms (6 points)

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

A. Exercise 35.4-2: The input to the ***MAX-3-CNF satisfiability problem*** is the same as for 3-CNF-SAT (which asks whether a conjunction of disjunctions, each of exactly 3 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 disjunctions. The ***MAX-CNF satisfiability problem*** is like the MAX-3-CNF satisfiability problem, except that it does not restrict each disjunction to have exactly 3 literals. Give a randomised 2-approximation algorithm for the MAX-CNF satisfiability problem. A high-level argument suffices for the proof of 2-approximation. (The used lecture slides from Princeton would call it a 1/2-approximation.)

B. Problem 35-1bce: ***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. The ***first-fit*** heuristic takes each object in turn and places it into the first bin that can accommodate it. Let $S = \sum_{i=1}^{n} s_i$.

   b. Argue that the optimal number of bins required, to be denoted by B^*, is at least $\lceil S \rceil$.

   c. Argue that the first-fit heuristic leaves at most one bin less than half full. Denote by B the number of bins used by the first-fit heuristic.

   e. Prove an approximation ratio of 2 for the first-fit heuristic.

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

Question 3: Randomised Algorithms (3 points)

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

A. Exercise 5.3-2: Professor Kelps decides to write a procedure that produces at random any permutation of the array $A[i\,..\,n]$ besides the identity permutation. He proposes the following procedure:

```
PERMUTE-WITHOUT-IDENTITY($A$)
$n = A.length$
for $i = 1$ to $n - 1$          // RANDOM($\ell, u$) returns a random number from $\ell$ to $u$
    swap $A[i]$ with $A[\text{RANDOM}(i + 1, n)]$
```

Does this code do what Professor Kelps intends?

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

```
PERMUTE-BY-CYCLIC($A$)
$n = A.length$
let $B[1\,..\,n]$ be a new array
$offset = \text{RANDOM}(1, n)$          // RANDOM($\ell, u$) returns a random number from $\ell$ to $u$
for $i = 1$ to $n$
    $dest = i + offset$
    if $dest > n$
        $dest = dest - n$
    $B[dest] = A[i]$
return $B$
```

[What is the probability for] each element $A[i]$ of winding up in any particular position in B? Show that Professor Armstrong is mistaken by showing that the resulting permutation is not uniformly random.

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 17.1-1: If the [seen] set $\{\text{PUSH}(S, x),\ \text{POP}(S),\ \text{MULTIPOP}(S, \ell)\}$ of stack operations included a $\text{MULTIPUSH}(S, [x_1, \ldots, x_k])$ operation, which pushes k items x_i onto the stack S, would the [seen] $O(n)/n = O(1)$ bound on the amortised cost [per operation of a sequence] of n stack operations continue to hold?

B. Exercise 17.2-2: 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 an *accounting 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).