Exam in Algorithms & Data Structures 3 (1DL481)

Prepared by Pierre Flener

Monday 15 March 2021 from 08:00 to 11:00; plus 30 minutes for uploading if home exam

Materials: This exam draws 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. You are ***not allowed*** to use any help (that is: ***no*** books, ***no*** slides, ***no*** notes, ***no*** electronic devices, ***no*** interaction with other people, etc) during the exam. If not in the exam hall, then you are ***only allowed*** to be on-line for downloading these questions and uploading your answers.

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, 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.

Help: Unfortunately, no teacher can attend the hall exam.

Uploading, by those not in the exam hall: Upload your answers via Studium: collate trimmed and aligned photos of your handwritten answer sheets, in the ***question order***, into a ***single*** file in ***PDF*** format, called ***vw-nnnn-xyz.pdf*** after your anonymous exam code of length 11. Files with other names or formats will be ignored.

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:

		–					–			

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

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

A. ***Container packing.*** Suppose that we are given a set of p parcels, where the volume v_i of the ith parcel satisfies $0 < v_i < 1$. We wish to pack all the parcels into the minimum number of unit-volume containers. Each container can hold any subset of the parcels whose total volume does not exceed 1. Prove that the problem of determining the minimum number of containers required is NP-hard, by reducing *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 is a subset $X' \subseteq X$ whose elements sum to t; for example, if $X = \{7, 9, 8\}$ and $t = 15$, then $X' = \{8, 7\}$ is a solution, but there is no solution for $t = 14$.

B. The ***weightlifting problem*** takes as input a finite set W of m weights w_i. The question is whether the weights can be partitioned into two sets L and $\overline{L} = W - L$ such that $\sum_{w \in L} w = \sum_{w \in \overline{L}} w$. Show that the weightlifting problem is NP-complete.

Start your answer on a new sheet and indicate there the chosen exercise (1A or 1B). You ***must*** use the terminology (container, parcel, volume, weightlifting, and weight) and mathematical object names (p, v_i, X, n, x_i, t, X', W, m, w_i, and L) of the question.

Question 2: Randomised Algorithms (3 points)

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

A. Professor Armstrong decides to write a procedure that produces at random any permutation of the vector $V[1\,..\,n]$ besides the identity permutation. He proposes the following:

```
PERMUTE(V)
  n = V.length
  for i = 1 to n − 1          // RANDOM(ℓ, u) returns a random number from ℓ to u
      swap V[i] with V[RANDOM(i + 1, n)]
```

Does this code do what Professor Armstrong intends? Why or why not?

B. Many randomised algorithms randomise the input by permuting the given input vector. Suppose that instead of swapping element $V[i]$ with a random element from the subvector $V[i\,..\,n]$, we swapped it with a random element from anywhere in the vector:

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

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

Start your answer on a new sheet and indicate there the chosen exercise (2A or 2B). You ***must*** use the terminology (vector) and mathematical object names (V, n, RANDOM, and PERMUTE) of the question.

Question 3: Amortised Analysis (4 points)

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

A. Suppose we perform a sequence of m functions on a data structure in which the jth function costs j if j is an exact power of 2, and 1 otherwise. Use an *accounting method of analysis* to determine the amortised cost per function.

B. Suppose we perform a sequence of m functions on a data structure in which the jth function costs j if j is an exact power of 2, and 1 otherwise. Use a *potential method of analysis* to determine the amortised cost per function.

Start your answer on a new sheet and indicate there the chosen exercise (3A or 3B). You ***must*** use the terminology (function) and mathematical object names (m and j) of the question.

Question 4: Approximation Algorithms (6 points)

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

A. The input to the ***MAX-3-SAT problem*** is the same as for 3-SAT (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-SAT problem*** is like the MAX-3-SAT 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-SAT problem, upon denoting its returned number of satisfied clauses by M, and the maximum number of satisfied clauses by M^*. (The Princeton slides would call it a 1/2-approximation.) State whether you use the Princeton convention or the textbook convention for defining the approximation ratio of a maximisation problem. A high-level argument for polynomial time complexity suffices.

B. ***Container packing***: Suppose that we are given a set of p parcels, where the volume v_i of the ith parcel satisfies $0 < v_i < 1$. We wish to pack all the parcels into the minimum number of unit-volume containers. Each container can hold any subset of the parcels whose total volume does not exceed 1. The ***first-fit*** heuristic takes each parcel in turn and places it into the first container that can accommodate it. Let $V = \sum_{i=1}^{p} v_i$.

   b. Argue that the optimal number of containers required, which is to be denoted by C^*, is at least $\lceil V \rceil$.

   c. Argue that the first-fit heuristic leaves at most one container less than half full.

   d. Prove that the number of containers used by the first-fit heuristic, which is to be denoted by C, is never more than $\lceil 2\,V \rceil$.

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

Start your answer on a new sheet and indicate there the chosen exercise (4A or 4B). You ***must*** use the terminology (SAT, container, parcel, and volume) and mathematical object names (m, C_i, M, M^*, p, v_i, V, C, and C^*) of the question.