Exam in Algorithms & Data Structures 3 (1DL481)

Prepared by Pierre Flener

Friday 10 March 2017 from 14:00 to 17:00, at Bergsbrunnagatan 15

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. 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. Answer ***two*** of Questions 2 to 4. 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. Answer each question only on the indicated pages. Do ***not*** write anything into the following table:

Question	Max Points	Your Mark
1	8	
2	6	
3	6	
4	6	
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*** 4 points on Question 1, ***and*** you have earned a pass grade (p = pass) on your oral presentation and attendance to the oral presentations of the other students:

Grade	Condition
5	$18 \leq e \leq 20 \ \wedge \ p = \text{pass}$
4	$14 \leq e \leq 17 \ \wedge \ p = \text{pass}$
3	$10 \leq e \leq 13 \ \wedge \ p = \text{pass}$
U	$00 \leq e \leq 09 \ \vee \ p \neq \text{pass}$

We will grade your ***first two*** answers in case you address all of Questions 2 to 4.

Identity: Your anonymous exam code: | | | | | | |

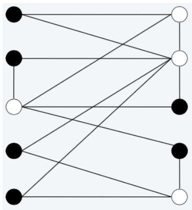
Answer to Question 1:

Question 1: NP-Completeness (*mandatory question*!) (8 points)

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

A. Exercise 34.5-7: The ***longest-simple-cycle problem*** is the problem of determining a simple cycle (no repeated vertices) of maximum length in a graph. Formulate a related decision problem, and show that the decision problem is NP-complete.

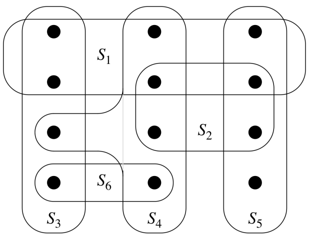
B. Problem 34-1a: An ***independent set*** of a graph $G = (V, E)$ is a subset $V' \subseteq V$ of vertices such that each edge in E is incident on at most one vertex in V'. The ***independent-set problem*** is to find a maximum-size independent set in G.



To the left is an independent-set instance (V, E), where the vertices of V are the 10 black and white points and the edges of E are depicted by lines. A maximum-size independent set consists of the 6 black vertices: each edge in E is incident on at most one black vertex.

Formulate a related decision problem for the independent-set problem, and prove that it is NP-complete, by reducing from the ***clique problem***, whose decision version asks if an undirected graph $G = (V, E)$ contains a clique of size k, that is a subset $V' \subseteq V$ of k vertices, each pair of which being connected by an edge in E.

C. Exercise 35.3-2: An instance $(X, \mathcal{F})$ of the ***set-covering problem*** consists of a finite set X and a family $\mathcal{F}$ of subsets of X, such that every element of X belongs to at least one subset in $\mathcal{F}$: $X = \bigcup_{S_i \in \mathcal{F}} S_i$. We say that a subset $S_i \in \mathcal{F}$ ***covers*** its elements. The problem is to find a minimum-size subset $\mathcal{C} \subseteq \mathcal{F}$ whose members cover all of X: $X = \bigcup_{S_i \in \mathcal{C}} S_i$.



To the left is a set-covering instance $(X, \mathcal{F})$, where X consists of the 12 black points and $\mathcal{F} = \{S_1, S_2, S_3, S_4, S_5, S_6\}$. A minimum-size set cover is $\mathcal{C} = \{S_3, S_4, S_5\}$, with size 3.

Show that the decision version of the set-covering problem is NP-complete by reducing it from the ***vertex-cover problem***, which asks to find a minimum-size vertex cover in a given undirected graph $G = (V, E)$, that is a minimum-size subset $V' \subseteq V$ such that if $(u, v) \in E$, then either $u \in V'$ or $v \in V'$ (or both).

Chosen exercise: ....

Continued answer (you should start on the previous side):

Answer to Question 2:

Question 2: Probabilistic Analysis and Randomised Algorithms $(3 + 3$ points)

Do ***both*** of the following short exercises from CLRS3:

A. Exercise 5.2-1: The procedure HIRE-ASSISTANT, given below, expresses a strategy for hiring the best applicant among candidates numbered 1 through n:

```
HIRE-ASSISTANT(n)
best = 0              // candidate 0 is a least-qualified dummy candidate
for i = 1 to n
    interview candidate i
    if candidate i is better than candidate best
        fire candidate best
        best = i
        hire candidate i
```

In HIRE-ASSISTANT, assuming that the n candidates are presented in a random order, what is the probability that you hire exactly one time? What is the probability that you hire exactly n times?

B. 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(ℓ, u) returns a random number from ℓ to u
    swap A[i] with A[RANDOM(i + 1, n)]
```

Does this code do what Professor Kelps intends?

Continued answer (you should start on the previous side):

Answer to Question 3:

Question 3: Amortised Analysis (6 points)

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

A. Exercise 17.1-3: 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 *aggregate analysis* to determine the amortised cost per operation.

B. Exercise 17.2-1: Suppose we perform a sequence of stack operations (PUSH or POP) on a stack whose size never exceeds k. After every k operations, a COPY operation is invoked automatically to make a copy of the entire stack for backup purposes. 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.

Chosen exercise: ....

Continued answer (you should start on the previous side):

Answer to Question 4:

Question 4: 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 Princeton lecture slides would call it a 1/2-approximation.]

B. Problem 35-3: ***Weighted set-covering problem***: Suppose that we generalise the set-covering problem (see Question 1C) so that each set S_i in the family $\mathcal{F}$ has an associated weight w_i and the ***weight*** of a cover $\mathcal{C}$ is $\sum_{S_i \in \mathcal{C}} w_i$. We wish to determine a minimum-weight cover. Section 35.3 handles the case in which $w_i = 1$ for all i, giving the following greedy set-covering heuristic:

GREEDY-SET-COVER$(X, \mathcal{F})$

$U = X$ // the set U maintains the set of remaining uncovered elements
$\mathcal{C} = \emptyset$ // the set $\mathcal{C}$ maintains the cover being constructed
while $U \neq \emptyset$
    select an $S_i \in \mathcal{F}$ that maximises $|S_i \cap U|$
    $U = U - S_i$
    $\mathcal{C} = \mathcal{C} \cup \{S_i\}$
return $\mathcal{C}$

On the instance of Question 1C, this heuristic produces a sub-optimal cover of size 4 by selecting either the sets S_1, S_4, S_5, and S_3 or the sets S_1, S_4, S_5, and S_6, in order. Show how to generalise this heuristic in a natural manner to provide an approximate solution for any instance of the weighted set-covering problem. Does the CLRS3 analysis for the unweighted case still carry through, establishing that the generalised heuristic also is a polynomial-time $(\ln|X|+1)$-approximation algorithm? [A yes/no answer with a high-level argument suffices for the last question.]

Chosen exercise: ....

Continued answer (you should start on the previous side):

Spare page for answers (or nice cartoons!)