Exam in Algorithms & Data Structures 3 (1DL481)

Prepared by Pierre Flener

Tuesday 15 March 2016 from 08:00 to 13:00, in Polacksbacken


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 on the indicated pages. Do not write anything into the following table:

<table>
<thead>
<tr>
<th>Question</th>
<th>Max Points</th>
<th>Your Mark</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>20</td>
<td></td>
</tr>
</tbody>
</table>

Help: Normally, a teacher will attend this exam from 09:00 to 10:00.

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:

<table>
<thead>
<tr>
<th>Grade</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>( 18 \leq e \leq 20 \land p = \text{pass} )</td>
</tr>
<tr>
<td>4</td>
<td>( 14 \leq e \leq 17 \land p = \text{pass} )</td>
</tr>
<tr>
<td>3</td>
<td>( 10 \leq e \leq 13 \land p = \text{pass} )</td>
</tr>
<tr>
<td>U</td>
<td>( 00 \leq e \leq 09 \lor p \neq \text{pass} )</td>
</tr>
</tbody>
</table>

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 34.5-2: 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 \( Ax \leq b \). Prove that 0-1 integer programming is NP-complete. (*Hint:* Reduce from 3-CNF-SAT, which asks whether a conjunction of disjunctions, each of exactly three distinct literals, is satisfiable; a **literal** is an occurrence of a Boolean variable or its negation.)

B. Exercise 34.5-5: The **set-partition problem** takes as input a set \( S \) of numbers. 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.

C. Exercise 35.3-2: An instance \((X, F)\) of the **set-covering problem** consists of a finite set \( X \) and a family \( F \) of subsets of \( X \), such that every element of \( X \) belongs to at least one subset in \( F \): \( X = \bigcup_{S \in F} S \). We say that a subset \( S \in F \) **covers** its elements. The problem is to find a minimum-size subset \( C \subseteq F \) whose members cover all of \( X \): \( X = \bigcup_{S \in C} S \). 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, Randomised Algorithms, and Universal Hashing (6 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?

B. Exercise 5.3-3: Many randomised algorithms randomise the input by permuting the given 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 between ℓ and u
3    swap $A[i]$ with $A[RANDOM(1, n)]$
```

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

Chosen exercise: ........

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, 17.2-2, or 17.3-2: Suppose we perform a sequence of \( n \) operations on a data structure in which the \( i \)th operation costs \( i \) if \( i \) is an exact power of 2, and 1 otherwise. Use either aggregate analysis, or an accounting method of analysis, or a potential method of analysis to determine the amortised cost per operation.

Chosen method: .................

Continued answer (you should start on the previous side):
Answer to Question 4:
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 (see Question 1A), 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.

B. Problem 35-1: Bin packing: Suppose that we are given a set of \( n \) objects, where the size \( s_i \) of the \( i \)th 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 is at least \( \lceil S \rceil \).

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

e. Prove an approximation ratio of 2 for the first-fit heuristic. (AD3 teacher’s hint: Use your results of tasks b and c.)

C. 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 \( F \) has an associated weight \( w_i \) and the weight of a cover \( C \) is \( \sum_{S_i \in C} w_i \). 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, where the set \( U \) contains, at each stage, the set of remaining uncovered elements and the set \( C \) contains the cover being constructed:

\[
\text{Greedy-Set-Cover}(X, F)
\]

\begin{enumerate}
\item \( U = X \)
\item \( C = \emptyset \)
\item \textbf{while} \( U \neq \emptyset \)
\item \hspace{1em} select an \( S_i \in F \) that maximises \( |S_i \cap U| \)
\item \hspace{1em} \( U = U - S_i \)
\item \hspace{1em} \( C = C \cup \{S_i\} \)
\item \textbf{return} \( C \)
\end{enumerate}

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 \( (\ln d) \)-approximation algorithm, where \( d \) is the maximum size of any set \( S_i \)? [A yes/no answer with a high-level argument suffices for the last task.]

Chosen exercise: 

Continued answer (you should start on the previous side):
Spare page for answers