CISC-365*
Test #1
October 4, 2017

Student Number (Required) ______________________

Name (Optional)________________________________

This is a closed book test. You may not refer to any resources.

This is a 50 minute test.

Please write your answers in ink. Pencil answers will be marked, but will not be re-marked under any circumstances.

The test will be marked out of 50.

<table>
<thead>
<tr>
<th>Question 1</th>
<th>/16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Question 2</td>
<td>/16</td>
</tr>
<tr>
<td>Question 3</td>
<td>/15</td>
</tr>
<tr>
<td>Question 4</td>
<td>/3</td>
</tr>
<tr>
<td>TOTAL</td>
<td>/50</td>
</tr>
</tbody>
</table>

“Change is the process by which the future invades our lives.”

Alvin Toffler, author of “Future Shock”
QUESTION 1 (16 Marks)

It is known that the 3-Colouring problem is NP-Complete.

(It is not important to this question, but the 3-Colouring problem is defined as: Given a graph G on n vertices, can we colour the vertices of G using no more than 3 colours in such a way that no vertices that are joined by an edge have the same colour?)

(a) (5 marks) Professor Parsley claims he has discovered an algorithm for solving the 3-Colouring problem. He has proved that his algorithm always runs in \(O(n^{16})\) time. What is your expectation regarding the correctness of Prof. Parsley’s algorithm? Why?

SOLUTION: I am 99.999999999% sure his algorithm is not correct. If it were correct it would constitute a proof that P = NP, which I don’t believe is true.

Marking:

5/5 for saying “not correct because that would imply P = NP”
3/5 for saying “not correct” but giving a poor reason such as “\(n^{16}\) is too high”
2/5 for saying “not correct” and giving no reason
2/5 for saying “can’t tell if it is correct” and mentioning P=NP in some way
1/5 for saying “correct” with or without reason
(b) (5 marks) Professor Pomegranate has also created an algorithm for the 3-Colouring problem. She has proved that her algorithm always solves the problem correctly. What is your expectation regarding the complexity of the running time of this algorithm? Why?

SOLUTION: I expect this algorithm does not run in polynomial time. If it did, it would prove P = NP, which I don’t believe is true.

Marking:
5/5 for answer as above
5/5 for “must run in exponential time, because otherwise P=NP”
3/5 for either “non-polynomial” or “exponential”, with poor reason such as “all NP-Complete problems are solvable in exponential time”
2/5 for either “non-polynomial” or “exponential”, without reason
2/5 for “can’t determine the complexity” and mentioning P=NP
1/5 for “polynomial time”
(c) (6 marks) It is known that the **2-Colouring problem** is in the class $P$. Professor Papaya has proved she can solve any $n$-vertex instance of the 3-Colouring problem by transforming it into a $2^n$-vertex instance of the 2-Colouring Problem. Does this prove $P = NP$? Why or why not?

**SOLUTION:** No, it does not. In order to prove $P = NP$ in this way we would need to transform an instance of 3-Colouring into an instance of 2-Colouring in polynomial time. Professor Papaya's transformation must take exponential time to construct the instance of 2-Colouring (since the constructed instance has $2^n$ vertices), so it is not a polynomial time transformation.

**Marking:**
6/6 for “no, because the transformation takes $2^n$ time”
4/6 for “no, because we can’t prove $P=NP$”
4/6 for “no, because we can’t transform an NP-Complete problem to a problem in P”
3/6 for “no” with no reason given
3/6 for “yes, because she has reduced an NP-Complete problem to a problem in P”
1/6 for “yes” with no reason given
QUESTION 2 (16 Marks)

Consider the following restricted form of the Subset Sum Problem:

**Multiples-of-5-Subset-Sum (M5SS):** Let \( S \) be a set of numbers all of which are multiples of 5, and let \( k \) be an integer that is a multiple of 5. Does \( S \) contain a subset that sums to \( k \)?

Example: \( S = \{15, 10, 1375, 840, 200\} \quad k = 225 \quad \text{… in this case the answer is “Yes”} \)
(a) (8 marks) Show that \( M5SS \in NP \) (ie. show that \( M5SS \) has the properties required to be a member of the class \( NP \))

SOLUTION:

\( M5SS \) is a decision problem for which the answer is completely determined by the instance – there is no randomness or nondeterminacy.

If the answer to an instance of \( M5SS \) is Yes and we know the details of the solution, we can verify it in \( O(n) \) time by adding the selected elements of \( S \) and confirming that they sum to \( k \).

Therefore \( M5SS \in NP \)

ALTERNATIVE SOLUTION: \( M5SS \) is a special case of Subset-Sum, which we know is \( \in NP \). Therefore \( M5SS \in NP \).

Marking

8/8 for the alternative solution – it is short but complete!

6/8 if they mention the “verify Yes in polynomial time” but leave out the “decision problem” requirement

3/8 if they mention the “decision problem” requirement but leave out the “verify Yes in polynomial time” requirement

4/8 if they show they know the requirements but cannot show that this problem meets them

1/8 to 3/8 for showing partial knowledge of the requirements with no connection to this problem
(b) (8 marks) Prove that M5SS is \(NP\)-Complete by reducing Subset Sum to M5SS. (Hint: how can you transform a set of integers into a set in which each element is a multiple of 5?)

**SOLUTION:** Let \( S, k \) be an instance of Subset Sum, and let \(|S| = n\).

Construct an instance \( S', k' \) of M5SS by multiplying each element of \( S \), and \( k \), by 5. This transformation takes \( O(n) \) time.

**Proof that the transformation is answer-preserving:**

Suppose the answer to Subset Sum on \( S, k \) is Yes. Then the corresponding subset in \( S' \) will sum to \( 5k \), which is the value of \( k' \) ... so the answer to M5SS on \( S', k' \) is Yes.

** Suppose the answer to M5SS on \( S', k' \) is Yes. Then the corresponding subset in \( S \) will sum to \( k'/5 \), which is the value of \( k \) ... so the answer to Subset Sum on \( S, k \) is also Yes.

**ALTERNATIVE ARGUMENT for paragraph **:** Suppose the answer to Subset Sum on \( S, k \) is No. That means every subset of \( S \) sums to something other than \( k \). That means every subset of \( S' \) sums to something other than \( k' \) since the sum of a subset in \( S' \) is just 5 times its sum in \( S \). So the answer to M5SS in \( S', k' \) is No.

Marking: for the transformation: 4 marks
- other valid transformations are possible. For example we can multiply all values by 10
- give part marks for partial answers
- give 2/4 for a transformation that doesn’t work

for demonstrating that it is answer-preserving: 4 marks
- part marks for incomplete or partially correct answers

4/8 for describing what needs to be done, but not being able to do it for this problem
QUESTION 3 (15 marks)

Let A and B be two sets, each containing n integers. Each of the sets is stored in an n-element array.

Create an algorithm to compute $A \cap B$ (that’s “$A$ intersect $B$”). Your algorithm should run in $O(n \times \log n)$ time.

(A note on data structures: many people are tempted to solve problems like this using hash-tables which give $O(1)$ expected case search time. Unfortunately the worst case search time for a hash-table is $O(n)$.)

Express your algorithm in clear pseudo-code or a standard procedural language. You may assume that \texttt{sort()} is a built-in function that runs in $O(n \times \log n)$ time.

\textbf{SOLUTION:}

$$A\_\text{INT}\_B = \emptyset$$

$A = \text{sort}(A)$

\texttt{for each element $x$ of $B$:}

\hspace{1em} \texttt{use binary search to determine in $x \in A$}

\hspace{2em} \texttt{if $x \in A$:}

\hspace{3em} \texttt{append $x$ to $A\_\text{INT}\_B$}

\hspace{1em} \texttt{... or ...}
ALTERNATIVE SOLUTION:

\[ A_{\text{INT}}B = \emptyset \]
A = sort(A)
B = sort(B)
# assume A and B are indexed from 1 to n
A_counter = 1
B_counter = 1
while (A_counter \leq n) and (B_counter \leq n):
    if A[A_counter] == B[B_counter]:
        append A[A_counter] to A_{INT}B
    elsif A[A_counter] < B[B_counter]:
        A_counter++
    else:
        B_counter++

Other O(n*\log n) solutions are possible.

Marking: 15/15 for any correct solution that runs in O(n*\log n) time

If a student wrote out the Binary Search algorithm and got it wrong, don’t penalize them

9/15 for algorithms that solve the problem but are not O(n*\log n) solutions

up to 7/15 for algorithms that have O(n*\log n) complexity but which don’t correctly solve the problem – actual mark should depend on “how close” the solution is to working
QUESTION 4 (3 Marks)

(a) The Cook-Levin Theorem was first proved in 2015.

TRUE   FALSE

SOLUTION: FALSE

(b) If A and B are both NP-Complete problems, then A \(\sim\) B and B \(\sim\) A

TRUE   FALSE

SOLUTION: TRUE

(c) If A \(\sim\) B and B \(\sim\) A, then A and B must both be NP-Complete problems

TRUE   FALSE

SOLUTION: FALSE

Marking: 1 mark for each correct answer. No penalty for incorrect answers.
What is the meaning of the figure above?