Student Number (Required) ______________________

SOLUTIONS

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>/18</th>
</tr>
</thead>
<tbody>
<tr>
<td>Question 2</td>
<td>/15</td>
</tr>
<tr>
<td>Question 3</td>
<td>/15</td>
</tr>
<tr>
<td>Question 4</td>
<td>/2</td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td>/50</td>
</tr>
</tbody>
</table>
General marking philosophy: a student who gives enough of an answer to show they understood what they were supposed to do, even if they couldn’t do it (or made lots of errors while doing it) should get at least 50% on that question.

Full marks should be given if a solution is sound and not missing anything important.

Feel free to give marks like 9.5/10 to a solution that is correct but contains a minor error.

Students may come up with solutions that are completely different from mine but still completely correct. Correct solutions should get full marks even if they don’t match mine.

Students should always get a few marks for trying a question. The only way to get a 0 is to leave the page blank or write something completely irrelevant.
QUESTION 1 (18 Marks)

Consider the following variation on the Subset Sum Problem:

25_Value_Subset_Sum: Given a set of exactly 25 integers S and a target integer k, does S contain a subset that sums to k?

a) [10 marks] Prove this problem is in \( \mathbf{P} \) by describing an algorithm to solve instances of the problem in polynomial time. You are not required to express your algorithm in a programming language – simply explain it in sufficient detail to demonstrate that it runs in polynomial time. You do not need to compute the exact order of your algorithm (no recurrence relations required!)

My Solution: Since the set has exactly 25 values, we can solve this problem in constant time by listing all the subsets (there are \( 2^{25} \) of them, which is a large number but still a constant), computing the sum of each of them (also constant time since the size of each subset is \( \leq 25 \)) and comparing each sum to k (also constant time).

Students should give a good description of their algorithm. It can be in text, as above, but must include comments about the polynomial complexity. It is not essential that they state that the complexity is constant time.

Students may include things in their algorithm such as sorting the set. This is not wrong – it is unnecessary but it doesn't affect the complexity.

The key thing for students to recognize in their solution is that since \( n \) is fixed at 25, the size of the set is not an issue in the complexity.

Students may come up with a solution and misanalyse the complexity of it - for example, they may claim the complexity is \( O(n^{25}) \) – in this case the mark should be something like 12/15 : 3 marks off for incorrect analysis.

Students who demonstrate that they know what a polynomial time algorithm is should get at least 8/15 even if they are unable to show an algorithm for this problem.
Students who cannot solve the problem, but who demonstrate an understanding that the general Subset_Sum problem is NP-Complete, should get at least 7/15. This doesn't answer the question but it shows that they understand why the question is interesting.

(Question 1 continues on the next page)
b) [4 marks] Suppose we prove that

\[ 25_{-}Value\_Subset\_Sum \ \alpha \ \subset \text{Subset\_Sum}. \]

(Remember, the \( \alpha \) means “reduces to”.) Does this give us any new information about the \( P = \text{NP} \) question? Explain your answer.

Solution: 25_Value_Subset_Sum is clearly in the set NP, and we know that Subset_Sum is an NP-Complete problem. Thus we already know that 25_Value_Subset_Sum reduces to Subset_Sum. The “new” result gives us no new information about \( P = \text{NP} \).

Students who get this wrong (usually by mis-remembering the direction in which reduction goes) should still get part marks – about 3/5 – if they demonstrate an understanding of the \( P = \text{NP} \) question.

c) [4 marks] Suppose we prove that

\[ \text{Subset\_Sum} \ \alpha \ 25_{-}Value\_Subset\_Sum. \]

Does this give us any new information about the \( P = \text{NP} \) question? Explain your answer.

Solution: 25_Value_Subset_Sum is clearly in the class NP, and in Part a) we saw that it is in the class P. We also know that Subset_Sum is NP-Complete. Showing that Subset_Sum \( \alpha \) 25_Value_Subset_Sum would prove \( P = \text{NP} \), since it would mean that Subset_Sum – and by transitivity, every problem in NP – can be solved in polynomial time.

Part marks should be given to students who demonstrate an understanding of the significance of \( P, \text{NP}, \text{NP-Complete} \), etc. even if they give an incorrect answer to this question. An answer that (falsely) claims no such proof can exist since we know \( P \neq \text{NP} \), should get about 2/5.
QUESTION 2 (15 marks)

Let X, Y and Z be lists (or 1-dimensional arrays) of integers, with

\[ |X| = |Y| = |Z| = n \]

Let k be any integer.

Consider the question: can you find an x in X, y in Y and z in Z such that x + y + z = k?

This can obviously be answered in \(O(n^3)\) time by examining all combinations.

a) [10 marks] Give an algorithm that runs in \(O(n^2)\) time. State your algorithm in clear pseudo-code or a procedural programming language such as Python, C or Java. You may assume that sort() is a built-in method that runs in \(O(n \times \log n)\) time.

Hint: choose a value in X. What are you now looking for in Y and Z?

```python
# I use 1-based addressing, so the first element of
# any set has subscript or index 1
sort Y ascending, Z ascending
for x in X:
    t = k - x
    a = 1, b = n
    while a <= n and b >= 1:
        c = Y[a] + Z[b]
        if c == t:
            Report "Yes", x, Y[a], Z[b]
            STOP
        elif c < t:
            a += 1
        else:
            b -= 1
    # loop did not find solution with this x, move
    # to next x
# solution not found for any x in X
report "No"
```
This algorithm uses the Horowitz-Sahni technique to look for \(y,z\) combinations that can go with each element \(x\) of \(X\) to total \(t-x\).

b) [5 marks] Explain why your algorithm runs in \(O(n^2)\) time.

Sorting the sets takes \(O(n \log n)\) time.

The “for \(x\) in \(X\)” loop executes \(n\) times

\[
\text{The nested while loop executes } \leq 2^n \text{ times}
\]

Therefore the loops combine to give \(O(n^2)\) time. This exceeds the sort complexity so the overall complexity is \(O(n^2)\).

MARKING: The students are most likely to run into difficulty with the technique for efficiently searching the \(Y\) and \(Z\) sets for a pair to go with any given \(x\) value. You may see \(O(n^3)\) algorithms. These (assuming they do solve the problem) should get marks between 50% for a completely naïve algorithm and 75% for one with a bit of insight (such as sorting the sets and trying to make use of that).

An interesting attempt is to sort only one of the sets (eg. \(Z\)). Then for each combination of \(x\) in \(X\) and \(y\) in \(Y\), use binary search to see if the value \(t-(x+y)\) is in \(Z\). This algorithm runs in \(O(n^2 \log n)\) so it does not meet the requirements of the question, but it does beat the naïve algorithm. I would give this solution at least 80%.
QUESTION 3 (15 marks)

Create an efficient algorithm to compute the value of \( x^n \) where \( x \geq 0 \) and \( n \) is a positive integer.

Note: the naïve method is simply to multiply \( x \) by itself \( n \) times. Your algorithm must do better than this. **Hint:** If \( n \) is even, \( x^n = n^{n/2} \times x^{n/2} \)

If \( n \) is odd, then \( x^n = x^{n-1} \times x \)

(Express your algorithm in clear pseudo-code or a procedural language like Python, C or Java).

**Solution:**

```python
def Power(x, n):
    if n == 1:
        return x
    elif n%2 == 0:
        m = Power(x, n/2)
        return m*m
    else:
        m = Power(x, (n-1)/2)
        return x*m*m
```

There are certainly other ways to arrive at the same result (including iterative rather than recursive methods). It is important that a recursive solution have a proper base case, and of course that the correct result is computed.

The key is to avoid duplicated effort – for example, do not compute \( \text{Power}(x,n/2) \times \text{Power}(x,n/2) \).

If the student gives an \( O(n) \) algorithm that does compute \( x^n \), they should get about 10/15

Give part marks to solutions that are incomplete or partially correct.
QUESTION 4 (2 marks)

True or false: Dijkstra’s Algorithm may fail if the graph contains any edges with negative weights.

TRUE

Solution: True

Marking: 2 for True, 0 for False