CISC-365*
Test #2
October 20, 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>/25</th>
</tr>
</thead>
<tbody>
<tr>
<td>Question 2</td>
<td>/22</td>
</tr>
<tr>
<td>Question 3</td>
<td>/3</td>
</tr>
<tr>
<td>TOTAL</td>
<td>/50</td>
</tr>
</tbody>
</table>

“Education, therefore, is a process of living and not a preparation for future living.”

Happy Birthday to John Dewey
QUESTION 1 (25 Marks)

You have \( n \) bottles of maple syrup to sell to \( n \) customers. Let \( s_i \) be the number of litres of syrup in bottle \( i \). Let \( p_j \) be the price per litre that customer \( j \) will pay. You can sell one bottle to each customer. Your goal is to maximize your income.

Example: suppose you have two bottles containing 10 and 5 litres each, and two customers who will pay $9 per litre and $7 per litre.

If you sell the 10 litre bottle to Customer 2 and the 5 litre bottle to Customer 1, your income is \( 10 \times 7 + 5 \times 9 = 115 \)

But if you sell the 10 litre bottle to Customer 1 and the 5 litre bottle to Customer 2, your income is \( 10 \times 9 + 5 \times 7 = 125 \)

The second option gives you a larger income.
(a) [10 Marks] Create a Greedy Algorithm that will match bottles with customers so that your total income is maximized. Express your algorithm in clear pseudo-code.

A useful fact:
if \( s_1 \geq s_2 \geq 0 \) and \( p_1 \geq p_2 \geq 0 \), then

\[
(s_1 \cdot p_1) + (s_2 \cdot p_2) \geq (s_1 \cdot p_2) + (s_2 \cdot p_1)
\]

Solution:

Sort \( S \) into descending order (and renumber so largest value is \( S_1 \))
Sort \( P \) into descending order (and renumber so largest value is \( P_1 \))
for \( i = 1 \) to \( n \):
    sell bottle \( i \) to customer \( i \)

Note: sorting both lists into ascending order achieves the same pairing – both methods are correct

Marking:  
10/10 for anything equivalent to this
7/10 for a correct algorithm that has higher complexity
4/10 for an algorithm that doesn’t find the right solution
0/10 for not answering
(b) [5 Marks] Determine the complexity of your algorithm.

Solution: sorting the sets is in O(n* log n). The loop is in O(n). Therefore the algorithm is in O(n* log n)

Marking: 5/5 for correctly analyzing their algorithm, even if their algorithm is incorrect or not optimal
2/5 for incorrectly analyzing their algorithm
0/5 for not answering

(c) [10 Marks] Prove that there is an optimal solution that makes the same first choice as your greedy algorithm.

Solution: Let O be an optimal solution that does not pair s₁ with p₁ (which is what the greedy algorithm does). Suppose O contains the pairings (s₁, pᵢ) and (sⱼ, p₁). Then the value of O contains the products s₁ * pᵢ and sⱼ * p₁. But we know s₁ ≥ sⱼ and p₁ ≥ pᵯ so our useful fact above says that if we swap these two pairs to get the products s₁ * p₁ and sⱼ * pᵯ the value of the solution cannot decrease. Thus the solution given by making this swap is also optimal, and it matches the algorithm’s first choice.
Marking: 10/10 for any good proof, including forgiveness for trivial errors such as an incorrect subscript
7/10 for an incomplete or slightly incorrect proof
4/10 for a very incorrect proof
0/10 for not answering
QUESTION 2 (22 Marks)

Consider this variation on the 0/1-Knapsack Problem:

On your trip to the far-off land of Trapezium, you are told that your luggage cannot exceed k kilograms. You have a large number of items you wish to take with you – for each item i, you know the mass in kilograms: $m_i$. You also know the value of each item, but by a remarkable coincidence each item is worth exactly $1. The items are not divisible – for each item you either take it or leave it behind.

(a) [8 Marks] Design a Greedy algorithm to maximize the total value of the items you can pack in your luggage while staying within the k kilogram limit. State your algorithm in clear pseudo-code.

Solution:

Sort the items into increasing order by mass, and renumber them so that $m_1 \leq m_2 \leq \cdots \leq m_n$

for $i = 1$ to $n$:

if item $i$ fits in the luggage:
    add item $i$ to the luggage
else:
    stop

or

Sort the items into increasing order by mass, and renumber them so that $m_1 \leq m_2 \leq \cdots \leq m_n$

remaining_capacity = k

$i = 1$

while $i <= n$ and $m_i \leq$ remaining_capacity:
    add item $i$ to the luggage
    remaining_capacity -= $m_i$
    $i++$

or

any other clear expression of the idea of taking the objects in order of increasing mass.
Marking:

Some students may not have recognized that with each item valued at $1, the value of a solution = the number of items in the solution ... so the goal is simply to take as many items as possible. Even if they don’t state this (or misunderstand), they can get full marks by giving the correct algorithm.

If they misunderstand the question but give a reasonable algorithm for what they thought they were expected to do, give about 8/10

If they understand the question but give an algorithm that doesn’t work (such as taking the items in a different order), give about 5/10
(b) [14 marks] Prove the correctness of your algorithm.

Solution:
Let \( \text{Value}(X) \) be the total value of set \( X \). Note that \( \text{Value}(X) = |X| \)

PBI:

Base case: if \( n = 1 \), the only possible solutions are to take the only item or leave it out. The algorithm takes it if possible, so the algorithm finds the optimal solution when \( n = 1 \)

Inductive Assumption: Suppose the Greedy Algorithm always finds an optimal solution when the number of items is \( \leq t \) for some \( t \geq 1 \)

Suppose the number of items is \( t+1 \). Consider the algorithm’s first choice: it includes item 1 if \( m_1 \leq k \)

Let \( S \) be the Greedy Algorithm’s solution.

Claim: if there is any non-empty solution, there is an optimal solution that includes item 1

Pf: if \( m_1 > k \), there is no non-empty solution (and the algorithm’s first choice is therefore correct). If there are any non-empty solutions, let \( O \) be an optimal solution that does not include item 1. Let item \( i \) be the smallest item in \( O \). Let \( O^* = O - \{i\} + \{1\} \). \( O^* \) is a feasible solution with the same total value as \( O \), so \( O^* \) is an optimal solution that contains item 1.
Now observe that after selecting item 1, the Greedy Algorithm is solving a problem with \( \leq t \) items. Let \( S' \) be the algorithm’s solution to that problem. Note that \( O^*\{-1\} \) is a solution to the same problem, so by the IA, \( \text{Value}(S') \) is \( \geq \text{Value}(O^*\{-1\}) \)

Therefore \( \text{Value}(S) = \text{Value}(S') + 1 \geq \text{Value}(O^*\{-1\}) + 1 = \text{Value}(O^*) \)

Therefore \( S \) is an optimal solution.

**Marking:**

*If they use PBI, give 4 marks for the base case and 10 marks for the inductive part. The proof doesn’t have to exactly match mine but it should be properly structured to show that the algorithm’s solution is optimal.*

*If they use a different proof technique (such as the “remove the differences” method outlined in the notes) please use the following rough marking scheme:*

<table>
<thead>
<tr>
<th>Score</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>14/14</td>
<td>for a correct proof or one with only trivial errors</td>
</tr>
<tr>
<td>10/14</td>
<td>for a proof that is mostly complete or contains minor errors</td>
</tr>
<tr>
<td>7/14</td>
<td>for a proof that shows some understanding of what is required</td>
</tr>
<tr>
<td>4/14</td>
<td>for a proof with many errors</td>
</tr>
<tr>
<td>0/14</td>
<td>for not answering</td>
</tr>
</tbody>
</table>
QUESTION 3 (3 Marks)

Greedy algorithms are most often used in situations that involve finding the “best” solution to a problem.

TRUE  FALSE

Solution: TRUE

Greedy algorithms evaluate many different solutions and choose the best one.

TRUE  FALSE

Solution: FALSE

Greedy algorithms were named after the Star Wars bounty hunter character Greedo.

TRUE  FALSE

Solution: FALSE, but should be true.

Marking: 1 mark for each correct answer.