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>/12</th>
</tr>
</thead>
<tbody>
<tr>
<td>Question 2</td>
<td>/12</td>
</tr>
<tr>
<td>Question 3</td>
<td>/16</td>
</tr>
<tr>
<td>Question 4</td>
<td>/10</td>
</tr>
</tbody>
</table>

TOTAL /50

“People will forget what you said, people will forget what you did, but people will never forget how you made them feel.”

Happy Birthday to Maya Angelou
Question 1 (12 marks)

For each of the following operations on a graph, state its complexity if the graph is stored in (a) an adjacency matrix or (b) a set of adjacency lists. You may assume that the graph has n vertices and the vertices are numbered from 1 to n.

1. [4 marks] add an edge to the graph

(a) adjacency matrix

Solution: $O(1)$

Marking:

Correct Answer: 2
Incorrect but plausible answer (eg $O(n)$): 1
Very wrong answer (eg $O(n^2)$): 1/2
No answer: 0

(b) adjacency lists

Solution: $O(1)$

Marking: as above

2. [4 marks] delete an edge from the graph

(a) adjacency matrix

Solution: $O(1)$

Marking: as above

(b) adjacency lists

Solution: $O(n)$

Marking: as above

(Question 1 continues on the next page)
3. [4 marks] determine the number of edges in the graph

(a) adjacency matrix

Solution: \( O(n^2) \)
Marking: as above

(b) adjacency lists

Solution: \( O(m) \) where \( m \) is the number of edges

\( O(n^2) \) is also correct but not as precise – give 1.5 marks

Marking: as above, with 1.5 for \( O(n^2) \) as noted
Question 2 (12 marks)

Consider the “impartial judge” problem: we are given a group of people and a pairwise “friendship” relationship. Within the group, person X and person Y have a dispute. We need to find somebody in the group who is not a friend of either of the two persons (ie. not a friend of X and also not a friend of Y) to impartially resolve the dispute.

(a) [4 marks] Give a pseudo-code algorithm for this problem. You may assume the people are numbered 1 to n. Make sure your algorithm handles the case where there is no suitable person.

Solution: There is a small ambiguity in the question: as the question is stated, if x and y are not friends then x is qualified to be an impartial judge. This is not what I intended – I should have added the requirement that neither x nor y can be the judge, and some students may have made that assumption. I’ll accept solutions for either form of the problem:

Solution 1:

```plaintext
found = False
for person = 1 to n {
    if (person,x) is not an edge &&
        (person,y) is not an edge :
        print(person, " can be the judge")
        found = True
        break
}
if not found:
    print("No impartial judges are available")
```
Solution 2:

```python
found = False
for person = 1 to n {
    if (person != x) && (person != y) &&
       (( person,x) is not an edge) &&
       ((person,y) is not an edge):
        print(person, " can be the judge")
        found = True
        break
}
if not found:
    print("No impartial judges are available")
```

Marking: Obviously the students' solutions do not have to be identical to mine. The key idea is to iterate through the vertices of the graph looking for one that meets the criteria for being an impartial judge.

Here is another approach that effectively does the same thing:

Create an array of length n, with one element for each vertex. Call the array Eligible, and initialize it to True for all vertices.
for each neighbour v of x:
    Eligible[v] = False
for each neighbour v of y:
    Eligible[v] = False
found = False
for v = 1 to n:
    if Eligible[v]:
        found = True
        print v," can be the judge"
        break
if not found:
    print "no impartial judges are available"

The only difference here is that eligibility is computed for all vertices, rather than computing it one vertex at a time until a judge is found.
Marking:
   for an algorithm that correctly finds a judge and correctly deals with the case where there is no judge:  4

   for an algorithm that is slightly incorrect but is on the right track:  3

   for an algorithm that is mostly correct but fails to deal with the “no judge” case:  2

   for an algorithm with major problems:  1

   for no answer  0

(Question 2 continues on the next page)
(Question 2 continued)

(b) [4 marks] Choose a data structure to represent the group of people and the relationship.

Solution: This depends on the details of the algorithm. For algorithms similar to the first two solutions shown above, an adjacency matrix makes sense because we are looking for the existence of very specific edges for each vertex. For these solutions, adjacency lists would not be a good choice because we would need to search each list for x and for y.

For algorithms similar to the third solution, either an adjacency matrix or adjacency lists are a good choice – both give efficient solutions.

Marking:

- for a data structure that allows the necessary operations to be executed efficiently 4
- for a data structure that allows the necessary operations to be executed, but not efficiently 2
- for a data structure that does not allow the necessary operations to be executed, but still shows understanding of what was needed (I can’t think of such a structure either) 1
- for no answer 0
(c) [4 marks] Determine the complexity of your algorithm, using your chosen data structure.

Solution:

All of the solutions shown above, coupled with the recommended data structures, run in $O(n)$ time.

Marking:

- for correctly determining the complexity, even if it is worse than $O(n)$: 4 marks
- for incorrectly determining the complexity, but getting it almost right: 3 marks
- for incorrectly determining the complexity, but showing some understanding of how complexity is determined: 2 marks
- for trying: 1 mark
- for no answer: 0 marks
Question 3 (16 marks)

Alice and Orville work with very large graphs from two different sources. All of the graphs have positive weights on the edges.

Alice’s graphs start with all possible edges present, and then $n$ edges are removed (where $n$ is the number of vertices). The deleted edges are chosen so that the graph is still connected.

Orville’s graphs start with just a tree connecting $n$ vertices, and then $n$ more edges are added.

All the graphs are stored using Adjacency Lists.

Alice and Orville both want to use Prim’s algorithm to find minimum-weight spanning trees of their graphs.

For each of Alice and Orville, recommend a data structure to use in the implementation of Prim’s algorithm, and give your reasoning.

(Question 3 continues on the next page)
(Question 3 continued)

For Alice [8 marks]:

\textbf{Solution:}

Alice's graphs have \( \frac{n \times (n - 1)}{2} \) edges, which simplifies to

\( \frac{n \times (n - 3)}{2} \), which means \( m \in \Omega(n^2) \). For dense graphs I would recommend using either a min-heap of the vertices, rebuilding the heap after each edge is chosen (Version 3), or a simple array containing vertices, closest neighbour, and cost (Version 4). Both of these solutions give \( O(n^2) \) complexity for the algorithm.
Marking:

The 8 marks are divided between choosing a data structure (4) and justifying the choice (4). In my solution above, I have given the reasoning first. Students can state the structure first and then justify it.

Students do not need to compute the precise number of edges, nor do they have to name the version of Prim’s algorithm they would use.

For recognizing that these are dense graphs, and that
is important in choosing the data structures

4/4

For recognizing that density is relevant, but not realizing that these are dense graphs

2/4

For answers that base their choice on some other criterion

1/4

For choosing an appropriate data structure for dense graphs

4/4

For choosing an inappropriate data structure for dense graphs, but which still works

2/4

For choosing a data structure that doesn’t really work at all

1/4
For Orville [8 marks]:

Solution:

Orville’s graphs have \((n - 1) + n\) edges, so \(m \leq 2n\), which means \(m \in O(n)\). For sparse graphs I would recommend either a min-heap of the edges or a min-heap of the vertices (that is, use Version 2 or Version 3 of the algorithm). Both of these give \(O(n \times \log n)\) complexity for the algorithm.

Marking:

Marks are allocated in exactly the same way as for Alice’s graphs. Once again students are not required to state the exact number of edges in Orville’s graphs, but they should recognize that these are sparse graphs.

- For recognizing that these are sparse graphs, and that is important in choosing the data structure 4/4

- For recognizing that density is relevant, but not realizing that these are sparse graphs 2/4

- For answers that base their choice on some other criterion 1/4

- For choosing an appropriate data structure for sparse graphs 4/4

- For choosing an inappropriate data structure for sparse graphs, but which still works 2/4

- For choosing a data structure that doesn’t really work at all 1/4
Question 4 (10 marks)

A proper colouring of a graph is an assignment of colours to the vertices so that no adjacent vertices have the same colour.

Suppose we have a graph on n vertices and m edges, and each vertex has been assigned a colour. What data structure would be most appropriate for use in an algorithm to determine if the colouring is proper? Why?

Solution:

To determine if the colouring is proper, we have to compare the colour of each vertex to the colours of all its neighbours. We know that using adjacency lists is more efficient than using an adjacency matrix for any algorithm that involves “look at all the neighbours of a vertex” operations.

Therefore the most appropriate data structure for this problem is a set of adjacency lists representing the graph.
Marking:

Adjacency Lists, with a good explanation
(“looking at all neighbours of each vertex” is acceptable) 10

Adjacency Lists, with a weak explanation, such as “they take up less space” 7

Adjacency Lists, with no explanation 5

Adjacency Matrix, with an explanation 5

Adjacency Matrix, with no explanation 2

No answer 0