Coláiste na Tríonóide, Baile Átha Cliath
Trinity College Dublin
Ollscoil Átha Cliath | The University of Dublin

Faculty of Engineering, Mathematics and Science
School of Computer Science and Statistics

ALGORITHMS AND DATA STRUCTURES II 2023/24 **Semester 2** **2024**

ALGORITHMS AND DATA STRUCTURES II

26 April 2024 **SC** **14.00 – 16.00**

Dr Anthony Ventresque

Instructions to Candidates:
Answer ***ALL*** of the five questions. All questions carry equal marks. The paper is marked out of 100. No books or notes are permitted. No calculator permitted.

Answer all the questions ***in the answer book*** (not the exam paper).

All the multiple choice questions require ***only 1 answer*** – the question will receive a 0 if multiple answers are chosen.

Instructions for invigilators:
No books or notes are permitted. No calculator permitted.

Question 1 **(20 Marks)**

a) What problem does Dijkstra's algorithm solve? **(1 Mark)**
   a. Finding the minimum spanning tree in a graph.
   b. Finding the shortest path from a single source to all other vertices in a weighted graph.
   c. Sorting a list of numbers.
   d. Finding the maximum flow in a network.

b) Kruskal's algorithm builds the minimum spanning tree by: **(1 Mark)**
   a. Selecting edges in decreasing weight order without forming a cycle.
   b. Selecting edges in increasing weight order without forming a cycle.
   c. Starting from the highest-degree vertex.
   d. Starting from the lowest-degree vertex.

c) Prim's algorithm starts with: **(1 Mark)**
   a. An empty graph and adds edges.
   b. A single vertex and adds edges and vertices until the tree spans all vertices in the graph.
   c. The shortest edge in the graph.
   d. The longest edge in the graph.

d) Prim's and Kruskal's algorithms always yield the same minimum spanning tree for a given graph. **(1 Mark)**
   a. True
   b. False

e) Assume that `Vertex` is the vertex type for a connected, directed, acyclic graph in which each vertex has a maximum out-degree of 3 – and assume there exists a Java method `find` that looks for a particular value `x` in a graph starting at `Vertex v`

```
boolean find(Vertex v, int x) {
      for (Vertex s : v.successors()) {
            if (s.getLabel() == x || find(s, x)) {
                  return true;
            }
      }
      return false;
}
```

Let N be the number of edges in the graph. For `Vertex w`, estimate the worst-case running time of `find(w, 42);` as a function of N. **(4 Marks)**

f) Draw the graph corresponding to the following adjacency matrix (vertex labels A–D for the rows and columns are above and to the left): **(2 Marks)**

$$\begin{array}{c|cccc|} & A & B & C & D \\ \hline A & 0 & 1 & 0 & 0 \\ B & 0 & 1 & 0 & 1 \\ C & 1 & 0 & 0 & 1 \\ D & 0 & 0 & 0 & 0 \\ \hline \end{array}$$

g) Give the adjacency-list representation of the above graph (question f). **(2 Marks)**

h) Using an adjacency-list representation of a graph, what is the running time in the worst case to check for the existence of an edge between two vertices? **(3 Marks)**

i) What is the running time of the same operation using an adjacency matrix, in the worst case? **(2 Marks)**

j) Why does Dijkstra's algorithm not work for graphs with negative edges? **(3 Marks)**

Question 2 **(20 Marks)**

a) What is the defining characteristic of a greedy best-first search algorithm? **(1 Mark)**

   a. It always selects the path that appears best at that moment.
   b. It evaluates paths based on their historical cost from the start.
   c. It uses a random selection strategy to determine the next step.
   d. It considers the cost to the goal and the cost from the start equally.

b) What is the primary difference between the greedy best-first search and A* algorithms? **(1 Mark)**

   a. A* is a type of greedy best-first search.
   b. Greedy best-first search considers the cost from the start node to the current node.
   c. A* considers both the cost from the start to the current node and an estimate of the cost from the current node to the goal.
   d. Greedy best-first search is optimal, whereas A* is not.

c) Which of the following is true about the heuristic function used in A* search? **(1 Mark)**

   a. It should overestimate the cost to the nearest goal.
   b. It should underestimate or exactly estimate the cost to the nearest goal.
   c. It has no impact on the performance of the algorithm.
   d. It only considers the cost from the start node to the current node.

d) Describe a situation where the greedy best-first search might fail to find the shortest path. **(3 Marks)**

e) What problem does the Bellman-Ford algorithm solve? **(1 Mark)**

   a. Finding the minimum spanning tree in a graph
   b. Searching for a target value in a sorted array
   c. Finding the shortest paths from a single source to all other vertices in a weighted graph
   d. Sorting a list of numbers in ascending order

f) Which of the following is a feature of the Bellman-Ford algorithm? **(1 Mark)**

   a. It cannot handle graphs with negative weight edges.
   b. It has a faster runtime than Dijkstra's algorithm for all types of graphs.
   c. It can detect negative weight cycles in a graph.
   d. It requires a graph to be acyclic.

g) What is the time complexity of the Bellman-Ford algorithm? **(1 Mark)**

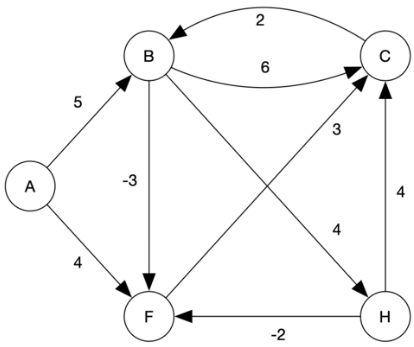
   a. $O(V + E)$
   b. $O(V^2)$
   c. $O(VE)$
   d. $O(E \log V)$

h) What is the primary reason the Bellman-Ford algorithm can handle negative weight edges? **(1 Mark)**
   a. It calculates distances only once.
   b. It uses a greedy strategy to pick the next vertex to process.
   c. It relaxes edges in a specific order based on their weights.
   d. It relaxes all edges a certain number of times equal to the number of vertices minus one.

i) Under what condition does the Bellman-Ford algorithm report that no solution exists? **(1 Mark)**
   a. When it finds an edge with a positive weight
   b. When it detects a negative weight cycle
   c. When there are disconnected components in the graph
   d. When the graph is directed

j) What does the relaxation process in the Bellman-Ford algorithm involve? **(1 Mark)**
   a. Removing all negative weight edges from the graph
   b. Updating the distance to a vertex if a shorter path is found
   c. Increasing the weight of all edges to remove negative cycles
   d. Swapping vertices until all distances are minimized

k) Fill-in the following matrix with the different values at different steps of the Bellman-Ford algorithm: **(4 Marks)**



	S	A	B	C	D	E
0	0	∞	∞	∞	∞	∞
1						
2						
…						

l) How does the Bellman-Ford algorithm ensure that it relaxes all edges in the correct order? **(4 Marks)**

Question 3 **(20 Marks)**

a) What problem does the Floyd-Warshall algorithm solve? **(1 Mark)**
   a. Finding the minimum spanning tree
   b. Finding the shortest path between all pairs of vertices
   c. Searching for a target value in a binary tree
   d. Detecting cycles in a directed graph

b) Which of the following best describes the Floyd-Warshall algorithm? **(1 Mark)**
   a. A greedy algorithm
   b. A divide and conquer algorithm
   c. A dynamic programming algorithm
   d. A backtracking algorithm

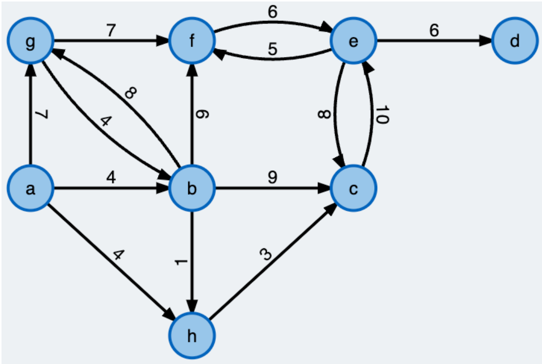
c) What is the time complexity of the Floyd-Warshall algorithm? **(1 Mark)**
   a. $O(V^2)$
   b. $O(V^2 \log V)$
   c. $O(V^3)$
   d. $O(VE)$

d) What initial values are used in the Floyd-Warshall algorithm for the distance matrix? **(1 Mark)**
   a. 0 for all pairs
   b. Infinity for all pairs except the diagonals, which are 0
   c. The weights of the edges for direct connections, and infinity otherwise
   d. 1 for connected vertices and 0 for non-connected vertices

e) Which of the following is a characteristic of the Floyd-Warshall algorithm? **(1 Mark)**
   a. It updates the shortest paths in a greedy manner.
   b. It recalculates paths to include each vertex as an intermediate vertex.
   c. It uses depth-first search to find the shortest path.
   d. It only works with directed acyclic graphs.

f) Explain how the Floyd-Warshall algorithm updates the distance between two vertices (give the formula). **(3 Marks)**

Given the following Graph, answer the next 3 questions:



g) Initialise the following matrix (step 0 of Floyd-Warshall) **(4 Marks)**

	A	B	C	D	E	F	G	H
A								
B								
C								
D								
E								
F								
G								
H								

h) Give the value of the matrix at step 2 of Floyd-Warshall **(4 Marks)**

i) Give the value of the matrix at the last step of Floyd-Warshall **(4 Marks)**

Question 4 **(20 Marks)**

a) What does a Trie primarily store? **(1 Mark)**
   a. Numbers
   b. Strings
   c. Graphs
   d. Arrays

b) Which of the following best describes a key characteristic of a Trie? **(1 Mark)**
   a. Each node represents the entire key.
   b. Nodes store the common prefix of keys.
   c. Each node represents a single character of a string.
   d. Nodes are connected in a linear fashion.

c) Tries are most commonly used for which application? **(1 Mark)**
   a. Balancing binary trees
   b. Storing relational database entries
   c. Performing arithmetic operations on large numbers
   d. Implementing dictionaries for spell-checking

d) What is the space complexity of creating a Trie that stores N keys, each of length M? **(1 Mark)**
   a. O(N)
   b. O(M)
   c. O(N + M)
   d. O(N*M)

e) Assuming all strings are of length N, what is the time complexity for searching and inserting in a Trie? **(1 Mark)**
   a. O(log N)
   b. O(N)
   c. O(N log N)
   d. O(1)

f) Why are Tries efficient for prefix searching? **(1 Mark)**
   a. Because they can store keys with common prefixes in shared nodes
   b. Because they automatically sort the keys alphabetically
   c. Because they use hash tables for indexing prefixes
   d. Because they reduce the search space by half at each step

g) What does each node in a Trie typically contain? (Assuming an alphabet of size A) **(1 Mark)**
   a. A list of A pointers, each to a child node
   b. A single character from the alphabet
   c. The complete string stored up to that point
   d. A numeric value representing the position of the node in the trie

h) Construct a Trie with the following key-value pairs. **(4 Marks)**

Key	Value
cat	1
cap	2
can	3
bat	4
ball	5

i) What is a Ternary Search Trie (TST)? **(1 Mark)**

a. A data structure that stores characters at tree nodes with each node having three children.
b. A type of binary search tree optimized for searching numeric data.
c. A trie where each node can have up to three parents.
d. A data structure where each node has two or three children, randomly assigned.

j) In a TST, what does the middle child of a node represent? **(1 Mark)**

a. The next character in a string if the current character matches.
b. A lesser value in the character set.
c. A greater value in the character set.
d. The end of a string.

k) Construct a TST with the following key-value pairs. **(4 Marks)**

Key	Value
she	1
sells	2
sea	3
shells	4
by	5
the	6
seashore	7

l) How does searching in a TST differ from searching in a traditional Trie, and what benefits does this bring? Detail how search is performed with each data structure. **(3 Marks)**

Question 5 **(20 Marks)**

a) What type of sorting algorithm is Radix Sort? **(1 Mark)**
   a. Comparison-based
   b. Divide and conquer
   c. Non-comparison based
   d. Recursive

b) What is a critical step in Radix Sort? **(1 Mark)**
   a. Choosing a suitable pivot
   b. Splitting the array into two halves
   c. Distributing elements into buckets based on their digits
   d. Swapping elements to their correct position

c) How does Radix Sort achieve sorting of integers? **(1 Mark)**
   a. By comparing pairs of elements
   b. By creating a binary search tree
   c. By distributing elements into buckets based on their radix
   d. By partitioning the dataset into smaller subsets

d) What is the time complexity of Radix Sort in the best case? **(1 Mark)**
   a. O(n log n)
   b. O(n)
   c. O(n+k)
   d. O(log n)

e) Most Significant Digit (MSD) Radix Sort is particularly effective for sorting what type of data? **(1 Mark)**
   a. Integers
   b. Floating-point numbers
   c. Strings
   d. All of the above

f) Which of the following best describes the LSD Radix Sort algorithm? **(1 Mark)**
   a. Starts sorting from the most significant digit
   b. Starts sorting from the least significant digit
   c. Uses a pivot element to sort
   d. Divides the array into two halves and sorts each half

g) What role does the choice of radix (base) play in the efficiency of Radix Sort? **(3 Marks)**

h) Describe a scenario where Radix Sort would be preferred over a comparison-based sorting algorithm. **(3 Marks)**

i) Provide the trace of sorting the array of strings given in the table below using LSD sort, providing an equivalent table for each of the 3 passes of the algorithm.

(4 Marks)

B	A	T
C	A	R
C	A	T
A	R	C
T	A	B
R	A	T
B	A	R

j) Provide the trace of sorting the array of strings given in the table below using MSD sort, providing an equivalent table for each of the 3 passes of the algorithm.

(4 Marks)

B	A	T
C	A	R
C	A	T
A	R	C
T	A	B
R	A	T
B	A	R

[oOo]