Algorithms and Data Structures

Makeup Exam

October 7, 2025

1. Consider the function `my_function(L, tol)` where `L` is assumed to be a Python list containing floating point numbers, and `tol` is assumed to be a single floating point number.

```
1 def my_function(L, tol):
2     for j in range(len(L)):
3         for k in range(j+1, len(L)):
4             if abs(L[j] - L[k]) <= tol:
5                 return True
6     return False
```

   (a) What is this function doing?

   (b) What is its worst case asymptotic running time as a function of `n = len(L)`?

   (c) Give an asymptotically faster equivalent implementation of `my_function`.

   (5+5+5)

2. You are given that the *pre-order* traversal of a binary tree yields the sequence A–B–D–E–C–F, while the *in-order* traversal of the same tree yields the sequence D–B–E–A–F–C.

   (a) Reconstruct the binary tree given this information.

   (b) State the *post-order* traversal sequence of the same tree.

   (c) Can you always reconstruct a binary tree if you know the pre-order and the in-order traversal sequence? Explain why or give a counter example.

   (5+5+5)

3. Insert the following sequence of elements into an initially empty AVL tree. Indicate all subtree heights and show the changes introduced by the insertions and the subsequent rebalancing step-by-step. (You may consult the summary of tree rotations on the front page.)

$$10, 5, 2, 7, 6, 8, 9$$

   (10)

4. State one distinct advantage that would favor each of the following sorting algorithms for a specific use case.

(a) Insertion sort

(b) Quick sort

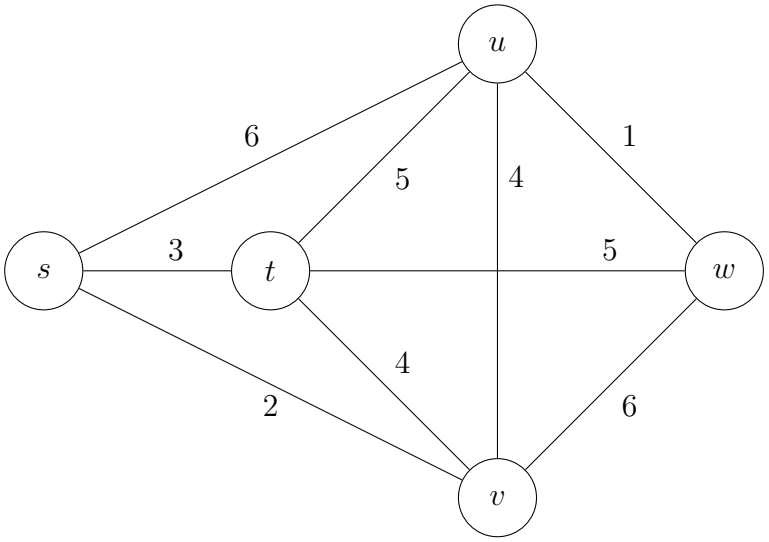
(c) Merge sort

(d) Heap sort

(e) Splay sort

(5)

5. Consider the following weighted graph.



(a) Draw the minimum spanning tree that results from running the Prim–Jarník algorithm on this graph, starting at vertex s. List the order in which edges are added to the tree.

(b) Draw the minimum spanning tree that results from running Kruskal's algorithm on this graph. List the order in which edges are added to the tree.

(5+5)

6. (a) State Dijkstra's algorithm for computing the shortest path in a weighted graph G with vertex set V, edge set E, and non-negative weights w from a starting vertex $s \in V$ to each vertex $v \in V$ using Python or pseudo-code. You may assume that you have access to an implementation of the Adaptable Priority Queue ADT.

(b) Adaptable priority queues are often implemented using the heap data structure. State the asymptotic running times for each of the core methods of the Adaptable Priority Queue ADT. Give a brief justification in each case.

(c) For Dijkstra's algorithm, there are situations where a simpler implementation of the Adaptable Priority Queue ADT is actually asymptotically faster. Explain!

(5+5+5)