Part 1.

1. what are the three primary criteria we seek when designing an algorithm

ans: correctness, effectiveness in terms of speed, memory, and optimality, and ease of implementation

2. List the five basic data structure types and their avg time and space complexities

ans:

	Space	Access	Search	Insert	Delete
Array	O(n)	O(1)	O(n)	*	*
Linked List	O(n)	O(n)	O(n)	*	*
Stack	O(n)	-	-	O(1)	O(1)
Queue	O(n)	-	-	O(1)	O(1)
Tree	O(n)	-	O(logn)	O(logn)	O(logn)

Array

insert/delete (append/pop) = O(1) amortized

" at beginning/middle = O(n)

Linked List

insert/delete at head/known node = O(1)

" tail = O(1)

3. What is the specific objective of the Interval Scheduling Problem?

ans: The objective is to find the maximum number of non overlapping events from a given set of intervals

4. List three different types of greedy strategies that could be used for interval scheduling

ans: Earliest Start Time - events in asc order of start time

Earliest Finish Time - " finish time

Shortest Distance/Interval - " interval length

5. What is the fundamental difference between an algorithm and a heuristic?

ans: algorithms are a set list of steps that results in a guaranteed correct solution whereas, a heuristic is an approach that quickly finds a good solution without a guarantee for its correctness and optimality

Part 2:

1. Describe the EFT greedy algorithm steps

ans: step 1 - sort events by their finish time in asc order
step 2 - choose the first item i.e. the event with the earliest finish time
step 3 - set the finish time of chosen interval as "current finish time"
step 4 - Loop through the remaining sorted events where
if current finish time ≤ start time
then set new finish time of the new event as "current finish time"
step 5 - Keep doing step 4 until all events are checked
step 6 - Print all selected intervals

2. Why is the nearest neighbour approach for the TSP considered a heuristic rather than a correct algorithm?

ans: Because this particular approach results in a valid tour but not necessarily the optimal/shortest tour

3. Explain the concept of a loop invariant and its relation to mathematical induction

ans: Invariant means that something is always true. A loop invariant is a condition where something is always true at the beginning and end of every iteration of a loop; Used to prove that an algorithm is correct.

As a form of proof, a loop invariant contains three main steps:
1. Initialization where the invariant is true before the first iteration
2. loop maintainance " , and still is after that iteration
3. Termination where the invariant + loop stopping condition implies that the algo's result is correct

On the other hand, mathematical induction is a logical structure that is used to prove a statement is true for all all integers. Therefore both loop invariant and mathemical induction are the same in different contexts.

4. What does it mean for a problem like the TSP to be NP-hard?

ans: NP-hard means that no efficient algorithm is known to solve all instances of a problem quickly as the number of points/nodes increases

5. How does modularity contribute to an algorithm's ease of implementation?

ans: modularisation is the practice of building programs using independent modules/components to simplify implementations. In addition, these components, or building blocks can be reused and tested independently thus, simplifying the overall implementation of an algorithm.

Part 3 :

1. Proof by Contradiction: Summarize the arguement used to prove that the greedy algorithm for interval scheduling cannot select fewer jobs than an optimal schedule

ans:

1. assumption : greedy algorithm for interval scheduling can select fewer jobs than an optimal schedule
2. by using EFT as our greedy approach,
   let g_1 be the first job chosen by greedy
   o_1 " optimal schedule
3. Since greedy will definitely choose earliest finish time,
   $\text{finish}(g_1) \leq \text{finish}(o_1)$
4. ∴ we can replace o_1 with g_1 without breaking feasability since g_1 finishes no later thus, leaving as much room for remaining jobs
5. As a result, we can construct a new optimal schedule that starts with g_1 where there would still be the same number of jobs as the optimal algorithm
6. By repeating the o_1 to g_1 replacement, we can see that greedy can be converted to an optimal schedule but still would not reduce the number of jobs
7. ∴ the assumption where greedy can select fewer jobs fails

∴ Greedy cannot select fewer jobs than optimal schedule

2. Counterexample Challenge: Draw or describe a set of intervals where the shortest Distance/Interval greedy strategy fails to provide a maximum size subset of compatible jobs

ans: Given the following intervals

$\{(0,1),(0,5),(1,5),(5,10)\}$

optimum $= \{(0,1),(1,5),(5,10)\}$

shortest distance $= \{(0,1),(1,5)\}$

3. Exhaustive Search Complexity: Why exhaustive (trying n! permutation) impractical for a Robot Tour Optimization w 20 points? Provide the approximate number of permutations involved

ans: Because exhaustive search would try all n! permutation which would be very slow as the number of points increases

when $n = 20$

$n! = 20!$

$\approx 2.4 \times 10^{18}$

4. Algorithm Correctness: The slides state that "Failure to find a counterexample... does not mean the algorithm is correct." Explain why mathematical induction is preferred over trial-and-error for proving correctness

ans: mathematical induction is an approach that is used to prove the truthfulness of an algorithm or statement for every possible input. Thus, a reliable way to prove correctness. Trial-and-error, on the other hand, is a way of testing algorithms on a case to case basis meaning that there could exist cases (i.e. inputs) where the algorithm fails.

5. Complexity Tradeoffs: Compare Quicksort and Merge sort based on their worst-case time complexity and space complexity

ans:	Worst case time complexity	" space
Quicks.	$O(n^2)$	$O(\log n)$
merges.	$O(n\log n)$	$O(n)$

comparison: merge sort results in a more favourable complexity of no more than $O(n\log n)$ in comparison to Quicksort. However, it can potentially use a larger amount of a maximum space of $O(n)$ in comparison to Quicksort.