1. What is the primary disadvantage of using a brute-force approach like an exhaustive search for solving problems, particularly as the input size grows?

A disadvantage is that the time complexity of the code will grow exponentially with the input size. This is an issue as an exponential time complexity will increase very fast with relation to increasing input size. Compared to other approaches the complexity increases much faster so for larger problems it may be optimal to use another approach.

1. In the context of greedy algorithms, what does it mean to make a "locally optimal choice"?

This is the option that the greedy algorithm makes while running, in the case it has multiple choices to make, it will always choose the best possible choice for the current moment or operation, without taking into consideration the overall problem.

1. Explain why finding a counterexample is sufficient to prove that an algorithm is incorrect.

By finding a counterexample you have found an input case where the algorithm will not provide the correct solution. Therefore, you have shown that the algorithm is not correct for all cases possible.

1. Why is understanding the growth of functions important when analyzing the time complexity of algorithms?

Growth of functions can impact the time complexity of solutions based on input size. For example, a linear function would only be O(n), meaning as input size increases it will become more complex but nothing compared to an exponential O(2^n), which will become many multiple times more complex than the linear function as input increases. By understanding growth of functions, we can better choose optimal solutions based of the problems needs.

1. Suppose that for inputs of size n on a particular computer, insertion sort runs in *8n2* steps, and merge sort runs in *64nlgn steps*. For which values of *n* does insertion sort beat merge sort?

*8n2*  < *64nlogn*

*8n < 64logn*

*n < 8logn*

*when n = 1, 1 < 0 , False*

*…*

*when n=6, 6 < 6.225 , true*

*then when n=7 , 7 < 6.7607, False*

*Therefore we see that insertion sort is better when n is in between 1 and 7.*