**Example Algorithms**

Linear Search (Iterative):

Explanation: Imagine you have a bunch of numbers written on a piece of paper, and you want to find the biggest one. The simplest way is to go through each number, one by one, and keep track of the biggest one you've seen so far. If you see a number that's bigger than the one you're currently remembering as the biggest, you change your memory to that bigger number. This way, when you've looked at all the numbers, the one you're remembering is the biggest.

Divide and Conquer (Recursive):

Explanation: Think of having a lot of friends each holding a list of numbers. Instead of checking all the numbers yourself, you ask your friends to help you. Each friend checks their list and tells you the biggest number they found. Then you compare all the biggest numbers your friends found, and the biggest one out of all of them is your answer.

Sorting:

Explanation: Suppose you have a stack of cards with numbers written on them, and you want to find the biggest number. One way to do it is to arrange the cards in order from smallest to largest. Once you've done that, you just look at the last card in the stack, and that's the biggest number.

Priority Queue (Heap):

Explanation: Imagine a line of students, each holding a card with a number written on it. A priority queue is like a line where the student with the highest number goes to the front. So, you ask the student at the front of the line for their card, and that's the biggest number.

**Question:**

Write an algorithm that finds the **maximum elements in an array** full of integers. Explain why did you choose this approach.

For example if the array is [48 ,15 ,34 ,21 ,8] then your algorithm gives the output 48.

You algorithm:

1.

2.

3.

4.

5.

6.

**Example Algorithms**

Iterative Algorithm:

Explanation: Think of factorial like making a big pyramid of blocks. To find the factorial of a number, you start with 1 and keep multiplying by each number up to the given number. For example, if you want to find the factorial of 5, you start with 1 and multiply by 2 to get 2, then multiply by 3 to get 6, then multiply by 4 to get 24, and finally multiply by 5 to get 120. It's like stacking blocks one by one to build a big pyramid.

Recursive Algorithm:

Explanation: Imagine you have a magical rule book that tells you how to find the factorial of any number. To find the factorial of a number, you look it up in the rule book. If the number is 1, the rule book says the factorial is 1. Otherwise, you look up the factorial of the number one less than the one you're given, and you multiply that by the number you were given. For example, to find the factorial of 5, you look up the factorial of 4 (which is 24) and then multiply by 5 to get 120. It's like following a set of instructions until you reach the answer.

**Question:**

Write an algorithm that finds the factorial of a given integer. Explain why did you choose this approach.

For example if the number given is 5, then your algorithm gives the output 120.

You algorithm:

1.

2.

3.

4.

5.

6.

7.

8.

**Example Algorithms**

Iterative Algorithm:

Explanation: Imagine you have a string written on a piece of paper, and you want to reverse it. One way to do it is to start from the end of the string and take each letter one by one, moving towards the beginning. You write down each letter in reverse order as you go. For example, if you have the word "hello", you start with 'o', then 'l', then 'l' again, 'e', and finally 'h'. This way, you end up with the reversed string "olleh".

Recursive Algorithm:

Explanation: Think of the string as a bunch of letters arranged in a row. To reverse the string, you can take the last letter and put it in front, then reverse the remaining letters. For example, if you have the word "hello", you take 'o' and put it in front, then reverse the rest of the string "hell". Now you have "ohell". Then you take 'l' and put it in front, and reverse the rest of the string "hel". Now you have "lohell". You keep doing this until you reverse the entire string. This approach is like breaking down the problem into smaller pieces and solving each piece recursively.

**Question:**

Write an algorithm that reverses a string. Explain why did you choose this approach.

For example if the string given is “hello”, then your algorithm gives the output “olleh”.

You algorithm:

1.

2.

3.

4.

5.

6.

7.

8.

**Example Algorithms**

Iterative Algorithm:

Explanation: Imagine you have a sequence of numbers written down, and you want to find their sum. One way to do it is to start with 0 as the sum, then go through each number in the sequence one by one. For each number, you add it to the sum. So, if you have a sequence like [1, 2, 3, 4, 5], you start with 0, then add 1 to get 1, then add 2 to get 3, then add 3 to get 6, and so on until you've added all the numbers. This way, you end up with the total sum of all the numbers in the sequence.

Recursive Algorithm:

Explanation: Think of the sequence as a big pile of numbers. To find the sum of the sequence, you can break it down into smaller parts. For example, if you have a sequence like [1, 2, 3, 4, 5], you can split it into [1] and [2, 3, 4, 5]. Then, you find the sum of each part separately and add them together. This process continues recursively until you're left with individual numbers, and then you add them up. It's like breaking down a big problem into smaller pieces and solving each piece separately before putting everything back together to find the total sum.

Using Formula (Optimization Algorithm):

Explanation: Instead of adding up each number individually, you can use a mathematical formula to find the sum of a sequence. The formula for finding the sum of a sequence of numbers is (n \* (n + 1)) / 2, where n is the last number in the sequence. So, if you have a sequence like [1, 2, 3, 4, 5], you can plug in 5 for n and calculate (5 \* (5 + 1)) / 2 to get the sum, which is 15. This approach is much faster and more efficient than adding up each number individually, especially for large sequences.

**Question:**

Write an algorithm that calculates the sum of a sequence.

For example if the sequence given is “1+2+3+4+….+10”, then your algorithm gives the output 55.

You algorithm:

1.

2.

3.

4.

5.

6.

7.

8.

9.