

Evaluating Algorithms

Textbook

Evaluating Algorithms



When we create an algorithm, we evaluate how well it works based on correctness, clarity, and efficiency.

Correctness

Correctness means the algorithm consistently produces the accurate output for every valid input and finishes its task. If an algorithm isn't correct, its other qualities don't matter.

How to check for correctness:

- **Testing with Multiple Inputs:** Try your algorithm with various types of data:
 - **Typical Cases:** Normal, expected inputs.
 - **Edge Cases:** Extreme inputs (e.g., empty lists, single-item lists, lists with all identical items).
 - **Invalid Inputs:** What happens if the input is unexpected? A robust algorithm handles these gracefully (e.g., with an error message).
- **Debugging:** This is the process of finding and fixing mistakes ("bugs") when your algorithm doesn't work correctly.
 - **Step-by-step check:** Manually trace the algorithm's execution with a specific input.

- **Print values:** In programming, temporarily display variable values at different points to see what's happening.

Example: Finding the Smallest Number If your algorithm finds the smallest number in `[5, 2, 8, 1, 9]`, it should output `1`. For an empty list `[]`, it should handle it gracefully (e.g., return an error).

Clarity: Is it easy to understand?

Clarity (or readability) means how easy it is for a human to understand your algorithm. This is important for debugging, teamwork, and future updates.

How to make an algorithm clear:

- **Meaningful Names:** Use descriptive names for variables and functions (e.g., `student_score` instead of `s`).
- **Comments:** Add notes to explain complex parts or design choices.
- **Organized Steps:** Keep the algorithm's flow logical and easy to follow.

Calculating an Average

Let's take a well known algorithm for finding the average from a set of numbers and analyze it.

1. Is it correct? Does it accomplish what it was supposed to? Try this algorithm with multiple inputs and see if it gives the correct answer. Does it have any bugs?
2. Is it clear? Does it make sense to the user?

Less Clear Algorithm:

```
1 function calc(data):
2     total = 0
3     i = 0
4     while i < length(data):
5         total = total + data[i]
6         i = i + 1
7     if length(data) > 0:
8         return total / length(data)
9     else:
10        return 0
11
12
```

More Clear Algorithm:

```
1 function calculate_average(numbers_list):
2     if length(numbers_list) == 0:
3         return 0 # Handle empty list
4
5     sum_of_numbers = 0
6     for number in numbers_list:
7         sum_of_numbers = sum_of_numbers + number
8
9     average_result = sum_of_numbers / length(numbers_list)
```

```
10     return average_result
11
12
```

The second version is clearer due to better naming and concise logic.

Invitation

Take the above algorithm and improve on it! What improvements can you make? How could you write a new program that accomplishes the same thing or make it better?

Critical Thinking Questions

1. Imagine you're developing an algorithm for a system that controls traffic lights at a busy intersection. Why would thoroughly testing this algorithm with "edge cases" (like unusually high traffic at one turn, or a complete power outage) be even more critical than just testing with typical traffic flow?
2. You've been given two different algorithms that both correctly sort a list of names alphabetically. One algorithm is much longer and uses very short, unclear variable names, while the other is shorter and uses descriptive names and comments. If you had to choose one to maintain and update over the next five years, which would you pick and why?
3. An algorithm designed to calculate student grades works perfectly for students who complete all assignments. However, when tested with a student who has missing assignments, it crashes. Which aspect of algorithm evaluation (correctness, clarity, or efficiency) does this problem primarily relate to, and why is addressing this particular issue essential before the algorithm can be considered reliable?

Questions (5)

1. You create an algorithm that is supposed to find the largest number in a list. When you give it the list [3, 7, 2, 9], it correctly outputs 9. What aspect of algorithm evaluation does this demonstrate?

MULTIPLE CHOICE

Choose the correct answer:

- A. Clarity
- B. Efficiency
- C. Correctness
- D. Debugging

2. Your algorithm is designed to calculate an average. When you give it an empty list [], it produces an error message instead of crashing. This shows the algorithm handles what well?

MULTIPLE CHOICE

Choose the correct answer:

- A. Typical Cases
- B. Clarity
- C. Edge Cases
- D. Efficiency

3. You are looking at two algorithms. One uses variable names like x, y, z, while the other uses names like student_name, grade_average. Which aspect of algorithm evaluation is better in the second example?

MULTIPLE CHOICE

Choose the correct answer:

- A. Correctness
- B. Efficiency
- C. Clarity
- D. Debugging

4. Why are "comments" important in an algorithm, even if the algorithm works perfectly without them?

MULTIPLE CHOICE

Choose the correct answer:

- A. They make the algorithm run faster.
- B. They hide errors from the computer.
- C. They help humans understand complex parts or design choices.
- D. They are only for very simple algorithms.

5. When you are trying to find and fix mistakes in your algorithm, what is this process called?

MULTIPLE CHOICE

Choose the correct answer:

- A. Compiling
- B. Debugging
- C. Evaluating
- D. Commenting

Answer Keys & Solutions

Questions

1. You create an algorithm that is supposed to find the largest number in a list. When you give it the list [3, 7, 2, 9], it correctly outputs 9. What aspect of algorithm evaluation does this demonstrate?

MULTIPLE CHOICE

Correct Answer:

- | | |
|----------------|-------------|
| A. Clarity | ✗ Incorrect |
| B. Efficiency | ✗ Incorrect |
| C. Correctness | ✓ Correct |
| D. Debugging | ✗ Incorrect |

Explanation:

Think about whether the algorithm gives the right answer for a normal input.

2. Your algorithm is designed to calculate an average. When you give it an empty list [], it produces an error message instead of crashing. This shows the algorithm handles what well?

MULTIPLE CHOICE

Correct Answer:

- | | |
|------------------|-------------|
| A. Typical Cases | ✗ Incorrect |
| B. Clarity | ✗ Incorrect |
| C. Edge Cases | ✓ Correct |
| D. Efficiency | ✗ Incorrect |

Explanation:

Consider how a robust algorithm deals with unusual or extreme inputs.

3. You are looking at two algorithms. One uses variable names like `x`, `y`, `z`, while the other uses names like `student_name`, `grade_average`. Which aspect of algorithm evaluation is better in the second example?

MULTIPLE CHOICE

Correct Answer:

- A. Correctness ✗ Incorrect
- B. Efficiency ✗ Incorrect
- C. Clarity ✓ Correct
- D. Debugging ✗ Incorrect

Explanation:

Think about what makes code easy for a human to understand.

4. Why are "comments" important in an algorithm, even if the algorithm works perfectly without them?

MULTIPLE CHOICE

Correct Answer:

- A. They make the algorithm run faster. ✗ Incorrect
- B. They hide errors from the computer. ✗ Incorrect
- C. They help humans understand complex parts or design choices. ✓ Correct
- D. They are only for very simple algorithms. ✗ Incorrect

Explanation:

Think about who benefits from notes within the code.

5. When you are trying to find and fix mistakes in your algorithm, what is this process called?

MULTIPLE CHOICE

Correct Answer:

- A. Compiling ✗ Incorrect
- B. Debugging ✓ Correct

C. Evaluating

✗ Incorrect

D. Commenting

✗ Incorrect

Explanation:

The word literally means removing "bugs."