

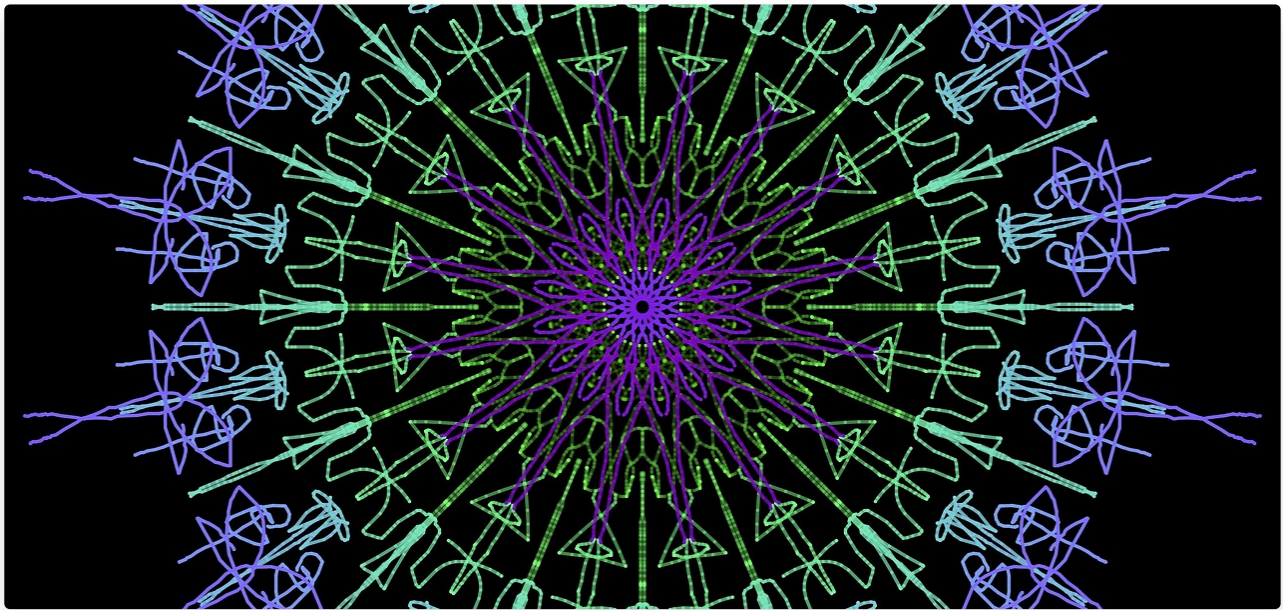
# Choosing the Best Algorithm

---

## Textbook

---

## Choosing the Best Algorithm



### Problem Solving in Action

Have you ever had to pick the best way to solve a problem? Maybe you were figuring out how to organize a group project, plan a party, or even rearrange your room. You probably thought through different steps, compared your options, and chose the method that made the most sense. That process is a lot like selecting an **algorithm**.

### What Is an Algorithm?

An **algorithm** is a set of steps that tells a computer—or a person—how to solve a problem or complete a task. Algorithms need to follow **logical rules** and work within **limitations**, just like we do when solving real-world problems.

For example, if you're planning to seat 24 students in a small classroom for a guest speaker, you need to:

- Make sure your method adds up to 24 desks (mathematical rule)
- Think about how much space you have (limitation)
- Choose a design that lets everyone see the speaker (accessibility)

This is where **choosing the right algorithm** becomes important.

### Understanding Limitations

When we create or choose an algorithm, we must think about its **limitations**—these are the things that might prevent it from working perfectly. Some examples include:

- Not enough time to set it up
- Not enough space for materials
- Too many complicated steps
- Not following mathematical rules (like forgetting to add up to the right total)

In computer science, ignoring limitations can cause a program to crash, work too slowly, or give the wrong results. The same is true when solving real-world problems!

## Selecting the Most Efficient Algorithm

Not all algorithms are created equal. Some are **more efficient** than others—they work faster, use fewer resources, or solve the problem better. When comparing different algorithms, we look at:

- **Time:** How long will it take to complete the steps?
- **Resources:** What do we need to make this work? (space, materials, people)
- **Accessibility:** Will it work well for everyone involved?

Just like programmers do, we make smart choices by comparing our options and picking the one that meets the needs of the situation best.

Here are some activity instructions designed to help students create efficient, reliable, and valid algorithms for decomposed problems, using contexts like video games, robot obstacle courses, and making dinner.

---

## Activity Instructions: Designing Algorithms

**Standard:** Create an algorithm to solve one or more parts of a decomposed problem.

Introduction:

When a big problem seems too hard to solve all at once, computer scientists (and good problem-solvers!) break it down into smaller, easier pieces. This is called decomposition. Once you have these smaller pieces, you can create a step-by-step plan, or algorithm, for each one. Your goal is to make your algorithms efficient (done in the best way, maybe fastest or simplest), reliable (always works the same way), and valid (actually solves the problem!).

### Activity 1: Video Game Mini-Quest Algorithm

**Problem Context:** Imagine you're playing a simple adventure video game. Your character needs to complete a mini-quest.

**The Big Problem:** Your character needs to find a hidden key, open a locked door, and then talk to a Non-Player Character (NPC) to get a reward.

**Instructions:**

1. **Decompose the Problem:** Break down the main quest into at least three smaller, simpler steps your character needs to complete. List them out.
  - *Example:*
    - Find the key.
    - Open the door.

- Talk to the NPC.
2. **Choose One Part:** Pick *one* of the smaller steps you just listed. This is the part you will create an algorithm for.
  3. **Create Your Algorithm (Step-by-Step Plan):**
    - Write down clear, precise, and unambiguous steps for your chosen part. Think about every small action the character would need to take.
    - Consider making it **efficient**: Is there a simpler or faster way for the character to achieve this goal?
    - Consider making it **reliable**: Will these steps *always* work, no matter what? For instance, if the key isn't exactly where expected, what should happen? (Keep it simple, but think about it!)
    - Consider making it **valid**: Does your algorithm actually solve *this specific* part of the problem?
    - *Example (for "Find the key"):*
      1. Move forward 5 steps.
      2. Turn right 90 degrees.
      3. Move forward until a sparkling object is detected.
      4. Pick up the sparkling object.
      5. Confirm object is "key." If not, go back to step 1 and try another direction.
  4. **Share and Discuss:** Share your chosen part and its algorithm with a partner or the class. Explain why you think your algorithm is efficient, reliable, and valid.

## Activity 2: Robot Obstacle Course Navigation Algorithm

**Problem Context:** You have a simple robot that can move forward, backward, turn left, turn right, and detect objects in front of it. It needs to navigate a small obstacle course.

**The Big Problem:** The robot starts at one end of a course, needs to pass a block, turn a corner, and stop at a finish line.

### Instructions:

1. **Decompose the Problem:** Break down the robot's journey into at least three smaller, manageable segments. List them out.
  - *Example:*
    - Move past the first block.
    - Turn the corner.
    - Reach the finish line.

2. **Choose One Part:** Select *one* specific segment of the robot's journey to create an algorithm for.
3. **Create Your Algorithm (Step-by-Step Plan):**
  - Write down each action the robot must take, step-by-step. Be very precise with movements and turns (e.g., "turn left 90 degrees").
  - **Efficiency:** Can the robot achieve this part of the course in the fewest steps possible?
  - **Reliability:** Will these steps work every time the robot encounters this type of obstacle or turn? Think about what would make it fail.
  - **Validity:** Does your algorithm specifically solve *this* segment of the obstacle course?
  - *Example (for "Move past the first block"):*
    1. Move forward 10 cm.
    2. Check for object directly in front.
    3. If object detected, turn right 45 degrees.
    4. Move forward 5 cm.
    5. Turn left 45 degrees.
    6. Continue moving forward until object is no longer detected.
    7. Move forward an additional 5 cm.
4. **Visualize and Refine:** Draw a simple diagram of your chosen course segment and trace your robot's path using your algorithm. See if you can spot any improvements.

## Activity 3: Making Dinner Algorithm (Simple Meal)

**Problem Context:** You are going to prepare a simple dinner that involves multiple steps, like making a sandwich and cutting some fruit.

**The Big Problem:** Prepare a peanut butter and jelly sandwich and slice an apple.

### Instructions:

1. **Decompose the Problem:** Break down the overall dinner preparation into at least three smaller, distinct tasks. List them out.
  - *Example:*
    - Gather ingredients for sandwich.
    - Assemble sandwich.
    - Prepare the apple.
2. **Choose One Part:** Select *one* specific task from your list to create an algorithm for.

### 3. Create Your Algorithm (Step-by-Step Plan):

- Write detailed, sequential steps. Imagine you are writing instructions for someone who has never made this before.
- **Efficiency:** Are there any unnecessary steps? Can you combine actions?
- **Reliability:** Will these steps consistently lead to the correct outcome every time they are followed? Are there any missing steps that could cause a problem?
- **Validity:** Does your algorithm truly accomplish *this particular* part of making dinner?
- *Example (for "Assemble sandwich"):*
  1. Take two slices of bread from the bag.
  2. Place bread on a clean plate.
  3. Open the peanut butter jar.
  4. Take a knife and scoop a spoonful of peanut butter.
  5. Spread peanut butter evenly on one slice of bread.
  6. Close the peanut butter jar.
  7. Open the jelly jar.
  8. Take a knife and scoop a spoonful of jelly.
  9. Spread jelly evenly on the other slice of bread.
  10. Close the jelly jar.
  11. Place the peanut butter side of bread onto the jelly side, forming a sandwich.

4. **Critique and Improve:** Have a classmate review your algorithm. Could they follow it perfectly? Did they find any missing steps or areas for improvement?

## Critical Thinking Questions

1. Imagine your school is planning a field trip. What algorithm could you use to organize how students board the buses? What limitations might you face?
  2. Why is it important for an algorithm to follow mathematical rules?
  3. You're given three algorithms to complete a task. One is fast but hard to understand, another is slow but simple, and the third requires special tools. Which would you choose and why?
-

## Questions (5)

### 1. What is an algorithm?

MULTIPLE CHOICE

Choose the correct answer:

- A. A type of computer
- B. A detailed set of steps to solve a problem
- C. A machine that organizes data
- D. A password for accessing programs

### 2. Which of the following is an example of a limitation when choosing an algorithm?

MULTIPLE CHOICE

Choose the correct answer:

- A. Having a fun idea
- B. Using a colorful design
- C. Not enough time to complete the steps
- D. Finishing early

### 3. Why is it important for an algorithm to follow mathematical rules?

MULTIPLE CHOICE

Choose the correct answer:

- A. So it looks nice
- B. So it is easy to read
- C. So it can be done by only one person
- D. So it gives correct and logical results

### 4. What is one factor to consider when selecting the most efficient algorithm?

MULTIPLE CHOICE

Choose the correct answer:

- A. How many colors it uses
- B. Whether your friends like it
- C. How fast and resourceful it is
- D. If it can be turned into a song

**5. You are given three algorithms to complete a task. What should you do first?****Choose the correct answer:**

- A. Pick the one that sounds the coolest
- B. Choose the one with the most steps
- C. Compare each one based on time, resources, and accessibility
- D. Ask someone else to decide for you

---

## Answer Keys & Solutions

---

### Questions

---

#### 1. What is an algorithm?

MULTIPLE CHOICE

**Correct Answer:**

- A. A type of computer ✗ Incorrect
- B. A detailed set of steps to solve a problem ✓ Correct
- C. A machine that organizes data ✗ Incorrect
- D. A password for accessing programs ✗ Incorrect

#### 2. Which of the following is an example of a limitation when choosing an algorithm?

MULTIPLE CHOICE

**Correct Answer:**

- A. Having a fun idea ✗ Incorrect
- B. Using a colorful design ✗ Incorrect
- C. Not enough time to complete the steps ✓ Correct
- D. Finishing early ✗ Incorrect

#### 3. Why is it important for an algorithm to follow mathematical rules?

MULTIPLE CHOICE

**Correct Answer:**

- A. So it looks nice ✗ Incorrect
- B. So it is easy to read ✗ Incorrect
- C. So it can be done by only one person ✗ Incorrect



D. So it gives correct and logical results

✓ Correct

#### 4. What is one factor to consider when selecting the most efficient algorithm?

MULTIPLE CHOICE

**Correct Answer:**

A. How many colors it uses

✗ Incorrect

B. Whether your friends like it

✗ Incorrect

C. How fast and resourceful it is

✓ Correct

D. If it can be turned into a song

✗ Incorrect

#### 5. You are given three algorithms to complete a task. What should you do first?

MULTIPLE CHOICE

**Correct Answer:**

A. Pick the one that sounds the coolest

✗ Incorrect

B. Choose the one with the most steps

✗ Incorrect

C. Compare each one based on time, resources, and accessibility

✓ Correct

D. Ask someone else to decide for you

✗ Incorrect