Problem Solving and Computational Thinking

Tushar B. Kute,
http://tusharkute.com

Computational Problem Solving

- ‘Computational problem solving’ is the iterative process of developing computational solutions to problems.
- Computational solutions are expressed as logical sequences of steps (i.e. algorithms), where each step is precisely defined so that it can be expressed in a form that can be executed by a computer.
- Much of the process of computational problem solving is thus oriented towards finding ways to use the power of computers to design new solutions or execute existing solutions more efficiently.

Computational Problem Solving

- Using computation to solve problems requires the ability to think in a certain way, which is often referred to as ‘computational thinking’.
- The term originally referred to the capacity to formulate problems as a defined set of inputs (or rules) producing a defined set of outputs.
- Today, computational thinking has been expanded to include thinking with many levels of abstractions (e.g. reducing complexity by removing unnecessary information), simplifying problems by decomposing them into parts and identifying repeated patterns, and examining how well a solution scales across problems.

Computational Problem Solving

- Computers and the technologies they enable play an increasingly central role in jobs and everyday life.
- Being able to use computers to solve problems is thus an important competence for students to develop in order to thrive in today's digital world.
- Even people who do not plan a career in computing can benefit from developing computational problem solving skills because these skills enhance how people understand and solve a wide range of problems beyond computer science.

Computational Problem Solving

- This skillset can be connected to multiple domains of education, and particularly to subjects like science, technology, engineering or mathematics (STEM) and the social sciences.
- Computing has revolutionised the practices of science, and the ability to use computational tools to carry out scientific inquiry is quickly becoming a required skillset in the modern scientific landscape.

Need a programming language?

- A programming language is an artificial language used to write instructions (i.e. code) that can be executed by a computer.
- However, writing computer code requires many skills beyond knowing the syntax of a specific programming language.
- Effective programmers must be able to apply the general practices and concepts involved in computational thinking and problem solving.

Computational problem solving skills

- Decompose problems
- Recognize and address patterns
- Generalize solutions
- Systematically test and debug

Programming Concepts Needed

- Sequences
- Conditionals
- Loops
- Functions

Define a problem

- To define a problem in computational problem solving, we need to clearly identify the inputs, outputs, and constraints of the problem.
- Inputs: The inputs to a computational problem are the data that is used to solve the problem. For example, the inputs to a sorting problem might be a list of unsorted numbers.
- Outputs: The outputs of a computational problem are the results that are produced by the problem-solving algorithm. For example, the output of a sorting problem might be a list of sorted numbers.

Define a problem

- Constraints:
  - The constraints of a computational problem are any restrictions that must be considered when solving the problem.
  - For example, a sorting problem might have a constraint that the algorithm must be efficient, meaning that it must be able to sort the list of numbers in a reasonable amount of time.

Define a problem

- Here are some tips for defining problems in computational problem solving:
  - Be clear and concise.
  - Use unambiguous language.
  - Identify all of the inputs, outputs, and constraints of the problem.
  - Write the definition in a way that is easy to understand for both humans and computers.

Identify a problem

- To identify a problem in computational problem solving, you can:
  - Look for opportunities to improve efficiency or accuracy. Are there any tasks that are currently being done manually that could be automated? Are there any calculations that could be made more accurate by using a computer?
  - Look for patterns or trends in data. Can you identify any patterns or trends in your data that could be used to make better predictions or decisions?

Identify a problem

- Look for ways to optimize existing systems.
  - Can you find ways to improve the performance or scalability of your existing systems?
- Look for new ways to use data.
  - Is there any data that you are not currently using that could be used to solve problems in new ways?
- Talk to stakeholders.
  - What are the biggest challenges that your stakeholders are facing? Can you think of any computational solutions to these challenges?

Identify a problem

- Here are some tips for identifying computational problems:
- Be open-minded.
  - Don't be afraid to think outside the box. Computational solutions can be used to solve a wide range of problems, so don't limit yourself to traditional problem-solving approaches.
- Be creative.
  - There is no one right way to solve a computational problem. Be creative and try different approaches.
- Be persistent.
  - Don't give up if you don't find a solution right away. Computational problem solving can be challenging, but it is also rewarding.

Problem Solving

- Computational problem solving can be used to solve a wide range of problems, including:
- Scientific problems: Computational problem solving is used in many scientific disciplines, such as physics, chemistry, and biology, to simulate complex systems and make predictions.
- Engineering problems: Computational problem solving is used in engineering to design and test products and systems.
- Business problems: Computational problem solving is used in business to optimize operations and make better decisions.
- Social problems: Computational problem solving is increasingly being used to address social problems, such as poverty, crime, and disease.

Problem Solving

- Computational problem solving can be broken down into the following steps:
  - Problem formulation: The first step is to clearly define the problem that needs to be solved. This involves understanding the inputs and outputs of the problem, as well as the constraints.
  - Algorithmic design: The next step is to design an algorithm to solve the problem. An algorithm is a step-by-step procedure for solving a problem.

Problem Solving

- Implementation:
  - The next step is to implement the algorithm using a programming language.
- Testing:
  - The next step is to test the implementation to ensure that it is correct and efficient.
- Deployment:
  - Once the implementation has been tested, it can be deployed to production.

Problem Solving

- Here are some tips for effective computational problem solving:
  - Choose the right tools.
    - There are many different computational tools and techniques available. It is important to choose the right tools for the problem that you are trying to solve.
  - Break the problem down into smaller pieces.
    - Complex problems can be more easily solved by breaking them down into smaller, more manageable pieces.

Problem Solving

- Use abstraction.
  - Abstraction is the process of hiding unnecessary details. When solving a problem, it is important to focus on the essential aspects of the problem and abstract away the unnecessary details.
- Test your solutions thoroughly.
  - It is important to test your computational solutions thoroughly to ensure that they are correct and efficient.
- Document your work.
  - It is important to document your work so that you can understand and maintain your solutions in the future.

Problem-1

- Write the steps to find minimum number from a collection of elements.

Problem-2

- Write the steps to find the number is odd or even.

Problem-3

- Write the steps to find the number is prime or not?

Problem-4

- Write the steps to swap the contents of two numbers.

Thank you

@mitu_skillologies @mITuSkillologies @mitu_group @mitu-skillologies @MITUSkillologies

kaggle

@mituskillologies

Web Resources
https://mitu.co.in
http://tusharkute.com

@mituskillologies

contact@mitu.co.in

tushar@tusharkute.com