Introduction to Programming

Week – *2*, Lecture – *1*

Algorithms and Procedures

SAURABH SRIVASTAVA

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

IIT KANPUR

Let’s revise some Mathematics !!

Assume that you are given a Quadratic Equation in one variable:

Let’s revise some Mathematics !!

Assume that you are given a Quadratic Equation in one variable:

- Say something like $x^2 + 2x = 15$

Let’s revise some Mathematics !!

Assume that you are given a Quadratic Equation in one variable:

- Say something like $x^2 + 2x = 15$

How can you solve this?

Let's revise some Mathematics !!

Assume that you are given a Quadratic Equation in one variable:

- Say something like $x^2 + 2x = 15$

How can you solve this?

Probably the first thing you may do with it, is to bring it in the "standard form"

- … which is $ax^2 + bx + c = 0$

Let’s revise some Mathematics !!

Assume that you are given a Quadratic Equation in one variable:

- Say something like $x^2 + 2x = 15$

How can you solve this?

Probably the first thing you may do with it, is to bring it in the “standard form”

- … which is $ax^2 + bx + c = 0$
- So, you rewrite it as $x^2 + 2x - 15 = 0$

Let’s revise some Mathematics !!

Assume that you are given a Quadratic Equation in one variable:

- Say something like $x^2 + 2x = 15$

How can you solve this?

Probably the first thing you may do with it, is to bring it in the “standard form”

- … which is $ax^2 + bx + c = 0$
- So, you rewrite it as $x^2 + 2x - 15 = 0$

Then, the next step is to see if there are any real roots

Let’s revise some Mathematics !!

Assume that you are given a Quadratic Equation in one variable:

- Say something like $x^2 + 2x = 15$

How can you solve this?

Probably the first thing you may do with it, is to bring it in the “standard form”

- … which is $ax^2 + bx + c = 0$
- So, you rewrite it as $x^2 + 2x - 15 = 0$

Then, the next step is to see if there are any real roots

- So, you calculate the discriminant, i.e. $D = b^2 - 4ac$

Let’s revise some Mathematics !!

Assume that you are given a Quadratic Equation in one variable:

- Say something like $x^2 + 2x = 15$

How can you solve this?

Probably the first thing you may do with it, is to bring it in the “standard form”

- … which is $ax^2 + bx + c = 0$
- So, you rewrite it as $x^2 + 2x - 15 = 0$

Then, the next step is to see if there are any real roots

- So, you calculate the discriminant, i.e. $D = b^2 - 4ac$
- … and, if $D >= 0$, we say that the equation does have real roots

Let’s revise some Mathematics !!

Assume that you are given a Quadratic Equation in one variable:

- Say something like $x^2 + 2x = 15$

How can you solve this?

Probably the first thing you may do with it, is to bring it in the “standard form”

- … which is $ax^2 + bx + c = 0$
- So, you rewrite it as $x^2 + 2x - 15 = 0$

Then, the next step is to see if there are any real roots

- So, you calculate the discriminant, i.e. $D = b^2 - 4ac$
- … and, if D >= 0, we say that the equation does have real roots

>= is how we usually write “greater than or equal to” while programming

Let’s revise some Mathematics !!

Assume that you are given a Quadratic Equation in one variable:

- Say something like $x^2 + 2x = 15$

How can you solve this?

Probably the first thing you may do with it, is to bring it in the “standard form”

- … which is $ax^2 + bx + c = 0$
- So, you rewrite it as $x^2 + 2x - 15 = 0$

Then, the next step is to see if there are any real roots

- So, you calculate the discriminant, i.e. $D = b^2 - 4ac$
- … and, if $D >= 0$, we say that the equation does have real roots
- … and if $D = 0$, we say that the equation has repeated real roots

>= is how we usually write “greater than or equal to” while programming

Let’s revise some Mathematics !!

Assume that you are given a Quadratic Equation in one variable:

- Say something like $x^2 + 2x = 15$

How can you solve this?

Probably the first thing you may do with it, is to bring it in the “standard form”

- … which is $ax^2 + bx + c = 0$
- So, you rewrite it as $x^2 + 2x - 15 = 0$

Then, the next step is to see if there are any real roots

- So, you calculate the discriminant, i.e. $D = b^2 - 4ac$
- … and, if $D >= 0$, we say that the equation does have real roots
- … and if $D = 0$, we say that the equation has repeated real roots

>= is how we usually write “greater than or equal to” while programming

Then, the two roots for x can be calculated as $\frac{-b \pm \sqrt{D}}{2a}$

How will you explain it to a younger sibling?

1. Check if the equation is in the standard form, i.e. $ax^2 + bx + c = 0$

How will you explain it to a younger sibling?

1. Check if the equation is in the standard form, i.e. $ax^2 + bx + c = 0$
2. If it is not, you can also do that by bringing all the terms on LHS, making the RHS 0

How will you explain it to a younger sibling?

1. Check if the equation is in the standard form, i.e. $ax^2 + bx + c = 0$
2. If it is not, you can also do that by bringing all the terms on LHS, making the RHS 0
3. Calculate the discriminant, D as $b^2 - 4ac$

How will you explain it to a younger sibling?

1. Check if the equation is in the standard form, i.e. $ax^2 + bx + c = 0$
2. If it is not, you can also do that by bringing all the terms on LHS, making the RHS 0
3. Calculate the discriminant, D as $b^2 - 4ac$
4. If $D = 0$
   - Find $\frac{-b}{2a}$; this is the only root of this equation

How will you explain it to a younger sibling?

1. Check if the equation is in the standard form, i.e. $ax^2 + bx + c = 0$
2. If it is not, you can also do that by bringing all the terms on LHS, making the RHS 0
3. Calculate the discriminant, D as $b^2 - 4ac$
4. If $D = 0$
   - Find $\frac{-b}{2a}$; this is the only root of this equation
5. Else, if $D > 0$
   - Find $\frac{-b + \sqrt{D}}{2a}$ and $\frac{-b - \sqrt{D}}{2a}$; these are the two roots of this equation

How will you explain it to a younger sibling?

1. Check if the equation is in the standard form, i.e. $ax^2 + bx + c = 0$
2. If it is not, you can also do that by bringing all the terms on LHS, making the RHS 0
3. Calculate the discriminant, D as $b^2 - 4ac$
4. If $D = 0$
   - Find $\frac{-b}{2a}$; this is the only root of this equation
5. Else, if $D > 0$
   - Find $\frac{-b + \sqrt{D}}{2a}$ and $\frac{-b - \sqrt{D}}{2a}$; these are the two roots of this equation
6. Else
   - Wait till you study complex numbers… till then, enjoy your life !!

How will you explain it to a younger sibling?

1. Check if the equation is in the standard form, i.e. $ax^2 + bx + c = 0$
2. If it is not, you can also do that by bringing all the terms on LHS, making the RHS 0
3. Calculate the discriminant, D as $b^2 - 4ac$
4. If $D = 0$
   - Find $\frac{-b}{2a}$; this is the only root of this equation
5. Else, if $D > 0$
   - Find $\frac{-b + \sqrt{D}}{2a}$ and $\frac{-b - \sqrt{D}}{2a}$; these are the two roots of this equation
6. Else
   - Wait till you study complex numbers… till then, enjoy your life !!

What we just saw, is the **Algorithm** to solve a quadratic equation

Algorithms – you know about them already

An algorithm is just a *sequence* of *well defined* steps towards *solving a problem*

Algorithms – you know about them already

An algorithm is just a *sequence* of *well defined* steps towards *solving a problem*

The problem could be as simple as solving a quadratic equation

- … or, as complex as launching a rocket to moon !!

Algorithms – you know about them already

An algorithm is just a *sequence* of *well defined* steps towards *solving a problem*

The problem could be as simple as solving a quadratic equation

- … or, as complex as launching a rocket to moon !!

Every problem, if it can be solved by human brains, has a corresponding algorithm

Algorithms – you know about them already

An algorithm is just a *sequence* of *well defined* steps towards *solving a problem*

The problem could be as simple as solving a quadratic equation

- … or, as complex as launching a rocket to moon !!

Every problem, if it can be solved by human brains, has a corresponding algorithm

Once we have an algorithm ready, all that is left, is to "make the computer understand it"

Algorithms – you know about them already

An algorithm is just a *sequence* of *well defined* steps towards *solving a problem*

The problem could be as simple as solving a quadratic equation

- … or, as complex as launching a rocket to moon !!

Every problem, if it can be solved by human brains, has a corresponding algorithm

Once we have an algorithm ready, all that is left, is to “make the computer understand it”

But computers (read *hardware*), only understands the language of 0s and 1s

Algorithms – you know about them already

An algorithm is just a *sequence* of *well defined* steps towards *solving a problem*

The problem could be as simple as solving a quadratic equation

- … or, as complex as launching a rocket to moon !!

Every problem, if it can be solved by human brains, has a corresponding algorithm

Once we have an algorithm ready, all that is left, is to “make the computer understand it”

But computers (read *hardware*), only understands the language of 0s and 1s

So we need a “translator” which can interpret our language to the language of 0s and 1s

Algorithms – you know about them already

An algorithm is just a *sequence* of *well defined* steps towards *solving a problem*

The problem could be as simple as solving a quadratic equation

- … or, as complex as launching a rocket to moon !!

Every problem, if it can be solved by human brains, has a corresponding algorithm

Once we have an algorithm ready, all that is left, is to "make the computer understand it"

But computers (read *hardware*), only understands the language of 0s and 1s

So we need a "translator" which can interpret our language to the language of 0s and 1s

This "translator" is called a *compiler*

- We will discuss more about compilers later… for now, just assume they can do this translation somehow !!

Designing Algorithms – Planning Inputs

Let us revisit the algorithm for solving quadratic equations

Designing Algorithms – Planning Inputs

Let us revisit the algorithm for solving quadratic equations

Assume that you have a compiler, that can change your Algorithm to a set of instructions

- Remember the instructions we saw in Lecture 0.2? Something like that…

Designing Algorithms – Planning Inputs

Let us revisit the algorithm for solving quadratic equations

Assume that you have a compiler, that can change your Algorithm to a set of instructions

- Remember the instructions we saw in Lecture 0.2? Something like that…

So, you have decided to make your computer solve these equations for you

- … and show off your skills in front of your younger siblings

Designing Algorithms – Planning Inputs

Let us revisit the algorithm for solving quadratic equations

Assume that you have a compiler, that can change your Algorithm to a set of instructions

- Remember the instructions we saw in Lecture 0.2? Something like that…

So, you have decided to make your computer solve these equations for you

- … and show off your skills in front of your younger siblings

To start with, you need an equation – something like $x^2 + 2x = 15$

Designing Algorithms – Planning Inputs

Let us revisit the algorithm for solving quadratic equations

Assume that you have a compiler, that can change your Algorithm to a set of instructions

- Remember the instructions we saw in Lecture 0.2? Something like that…

So, you have decided to make your computer solve these equations for you

- … and show off your skills in front of your younger siblings

To start with, you need an equation – something like $x^2 + 2x = 15$

But do you really need your sibling to type the whole equation like this?

- How about asking them to do some work first – like changing the equation to the standard form !!
- … meaning changing $x^2 + 2x = 15$ to $x^2 + 2x - 15 = 0$

Designing Algorithms – Planning Inputs

Let us revisit the algorithm for solving quadratic equations

Assume that you have a compiler, that can change your Algorithm to a set of instructions

- Remember the instructions we saw in Lecture 0.2? Something like that…

So, you have decided to make your computer solve these equations for you

- … and show off your skills in front of your younger siblings

To start with, you need an equation – something like $x^2 + 2x = 15$

But do you really need your sibling to type the whole equation like this?

- How about asking them to do some work first – like changing the equation to the standard form !!
- … meaning changing $x^2 + 2x = 15$ to $x^2 + 2x - 15 = 0$

Now, all you need from them *as inputs*, are the values 1, 2 and -15

- … corresponding to coefficients a, b and c in the standard form !!

Designing Algorithms – Identifying Outputs

What does your algorithm produce?

Designing Algorithms – Identifying Outputs

What does your algorithm produce?

It produces roots of the quadratic equations

- There can be *at max* two roots

Designing Algorithms – Identifying Outputs

What does your algorithm produce?

It produces roots of the quadratic equations

- There can be *at max* two roots

Let us call them x_1 and x_2

Designing Algorithms – Identifying Outputs

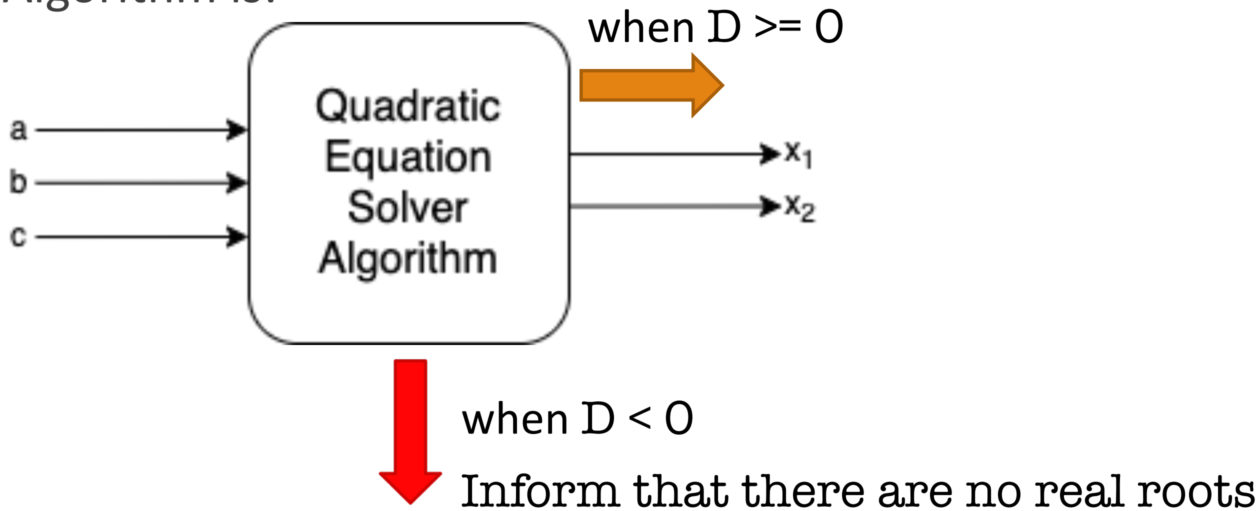
What does your algorithm produce?

It produces roots of the quadratic equations

- There can be *at max* two roots

Let us call them x_1 and x_2

So one way to represent the Algorithm is:



Designing Algorithms – Identifying Outputs

What does your algorithm produce?

It produces roots of the quadratic equations

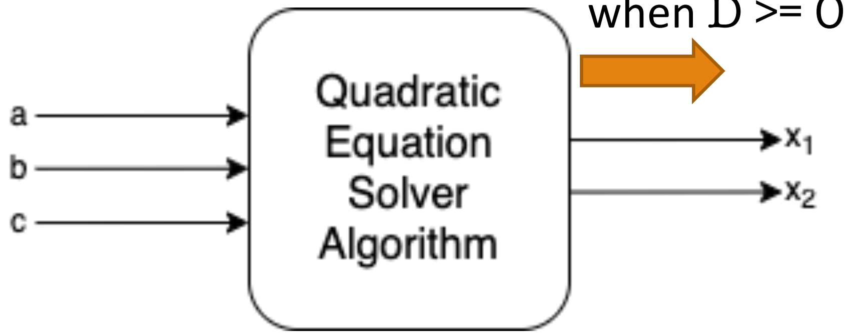
- There can be *at max* two roots

Let us call them x_1 and x_2

So one way to represent the Algorithm is:

when D >= 0

The Algorithm outputs x_1 and x_2, when they are real



when D < 0

Inform that there are no real roots

Designing Algorithms – Identifying Outputs

What does your algorithm produce?

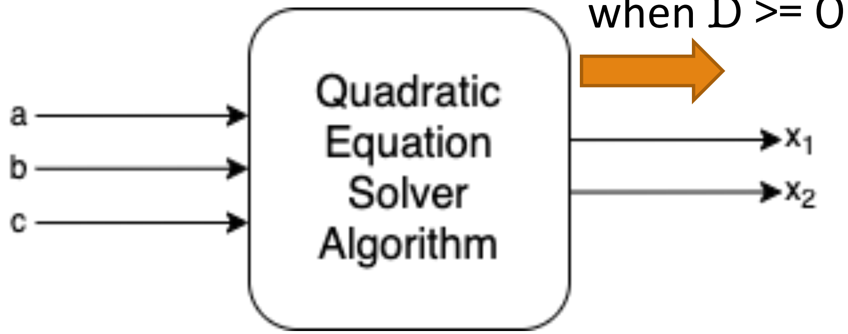
It produces roots of the quadratic equations

- There can be *at max* two roots

Let us call them x_1 and x_2

So one way to represent the Algorithm is:

when D >= 0



The Algorithm outputs "a message", if x_1 and x_2 are not real

when D < 0

Inform that there are no real roots

Writing Pseudocode

The actual set of instructions that a computer follows, is also called “code”

- What we saw in Lecture 0.2, was something similar to what is known as “assembly language code”

Writing Pseudocode

The actual set of instructions that a computer follows, is also called “code”

- What we saw in Lecture 0.2, was something similar to what is known as “assembly language code”

Once we have an algorithm to solve a problem, we may try to represent it more formally

Writing Pseudocode

The actual set of instructions that a computer follows, is also called “code”

- What we saw in Lecture 0.2, was something similar to what is known as “assembly language code”

Once we have an algorithm to solve a problem, we may try to represent it more formally

Instead of writing sentences in English, we can start using a more “constrained language”

- This constrained language, is often “inspired” from a programming language
- … i.e. it “closely resembles” instructions written in a particular programming language

Writing Pseudocode

The actual set of instructions that a computer follows, is also called "code"

- What we saw in Lecture 0.2, was something similar to what is known as "assembly language code"

Once we have an algorithm to solve a problem, we may try to represent it more formally

Instead of writing sentences in English, we can start using a more "constrained language"

- This constrained language, is often "inspired" from a programming language
- … i.e. it "closely resembles" instructions written in a particular programming language

We call this constrained language *pseudocode* – code that isn't really code !!

Writing Pseudocode

The actual set of instructions that a computer follows, is also called “code”

- What we saw in Lecture 0.2, was something similar to what is known as “assembly language code”

Once we have an algorithm to solve a problem, we may try to represent it more formally

Instead of writing sentences in English, we can start using a more “constrained language”

- This constrained language, is often “inspired” from a programming language
- … i.e. it “closely resembles” instructions written in a particular programming language

We call this constrained language *pseudocode* – code that isn’t really code !!

Example:

Algorithm	Pseudocode

If $D = 0$
Find $\frac{-b}{-2a}$; this is the only root of this equation
	`if (D = 0)`
	`{`
	`    x1 = x2 = -b / (2 * a)`
	`}`

Pseudocode for solving Quadratic Equations

```
Procedure QuadraticEquationSolver
    Inputs: a, b, c

    D = b * b - 4 * a * c;
    if (D = 0)
    {
        x1 = x2 = -b / (2 * a)
    }
    else if (D > 0)
    {
        x1 = (-b + √D ) / (2 * a)
        x2 = (-b - √D ) / (2 * a)
        return as Output : x1, x2
    }
    else
    {
        return as Output : "No real roots"
    }
```

Procedures

A Procedure is a collection of instructions, that perform a specific job

Procedures

A Procedure is a collection of instructions, that perform a specific job

These are instructions, which are to be to be followed in the exact sequence, every time !

- … and, they are usually followed from first to last, or not followed at all

Procedures

A Procedure is a collection of instructions, that perform a specific job

These are instructions, which are to be to be followed in the exact sequence, every time !

- … and, they are usually followed from first to last, or not followed at all

Think about the Quadratic Equation problem

- You follow the steps in an order, like calculating discriminant, and then, finding roots (not the other way)
- It makes sense only if you complete all the steps (for example, do not stop at calculating discriminant)

Procedures

A Procedure is a collection of instructions, that perform a specific job

These are instructions, which are to be to be followed in the exact sequence, every time !

- … and, they are usually followed from first to last, or not followed at all

Think about the Quadratic Equation problem

- You follow the steps in an order, like calculating discriminant, and then, finding roots (not the other way)
- It makes sense only if you complete all the steps (for example, do not stop at calculating discriminant)

Procedures are a way to represent one specific task in the real world

Procedures

A Procedure is a collection of instructions, that perform a specific job

These are instructions, which are to be to be followed in the exact sequence, every time !

- … and, they are usually followed from first to last, or not followed at all

Think about the Quadratic Equation problem

- You follow the steps in an order, like calculating discriminant, and then, finding roots (not the other way)
- It makes sense only if you complete all the steps (for example, do not stop at calculating discriminant)

Procedures are a way to represent one specific task in the real world

This task may be a part of a larger task

- For example, we may have a separate procedure to calculate Discriminants

Example of Procedures

Procedure QuadraticEquationSolver

```
Inputs: a, b, c

D = DiscriminantCalculator(a, b, c)
if (D = 0)
{
    x1 = x2 = -b / (2 * a)
}
else if (D > 0)
{
    x1 = (-b + √D ) / (2 * a)
    x2 = (-b - √D ) / (2 * a)
    return as Output : x1, x2
}
else
{
    return as Output : "No real roots"
}
```

Example of Procedures

Procedure QuadraticEquationSolver

```
    Inputs: a, b, c

    D = DiscriminantCalculator(a, b, c)
    if (D = 0)
    {
        x1 = x2 = -b / (2 * a)
    }
    else if (D > 0)
    {
        x1 = (-b + √D) / (2 * a)
        x2 = (-b - √D) / (2 * a)
        return as Output : x1, x2
    }
    else
    {
        return as Output : "No real roots"
    }
```

`DiscriminantCalculator` is a separate procedure here, which takes `a`, `b` and `c` as inputs

Example of Procedures

Procedure QuadraticEquationSolver

```
    Inputs: a, b, c

    D = DiscriminantCalculator(a, b, c)
    if (D = 0)
    {
        x1 = x2 = -b / (2 * a)
    }
    else if (D > 0)
    {
        x1 = (-b + √D) / (2 * a)
        x2 = (-b - √D) / (2 * a)
        return as Output : x1, x2
    }
    else
    {
        return as Output : "No real roots"
    }
```

Writing it on the RHS of = means that this procedure will ”return” a value, that should become the value of D

Example of Procedures

Procedure QuadraticEquationSolver

```
    Inputs: a, b, c

    D = DiscriminantCalculator(a, b, c)
    if (D = 0)
    {
        x1 = x2 = -b / (2 * a)
    }
    else if (D > 0)
    {
        x1 = (-b + √D ) / (2 * a)
        x2 = (-b - √D ) / (2 * a)
        return as Output : x1, x2
    }
    else
    {
        return as Output : "No real roots"
    }
```

Procedure DiscriminantCalculater

```
    Inputs: a, b, c

    D = b * b - 4 * a * c
    return as Output : D
```

Example of Procedures

Procedure QuadraticEquationSolver

```
    Inputs: a, b, c

    D = DiscriminantCalculator(a, b, c)
    if (D = 0)
    {
        x1 = x2 = -b / (2 * a)
    }
    else if (D > 0)
    {
        x1 = (-b + √D ) / (2 * a)
        x2 = (-b - √D ) / (2 * a)
        return as Output : x1, x2
    }
    else
    {
        return as Output : "No real roots"
    }
```

Procedure DiscriminantCalculater

```
    Inputs: a, b, c

    D = b * b - 4 * a * c
    return as Output : D
```

… and this is what the `DiscriminantCalculator` procedure may look like !!

Homework !!

Write the algorithm to solve a system of Linear Equation in three variables

Clearly differentiate between the cases when they have

- A unique solution
- Infinitely many solutions
- No solution

Write pseudocode for the above algorithm