CE2001/CZ2001: Algorithms

Introduction to Algorithms

Dr. Loke Yuan Ren

Learning Objectives

At the end of this lecture, students should be able to:

- Explain what an algorithm is
- Explain the different designs of algorithms using examples
  - Summation function
  - Fibonacci Sequence
  - Sine function
- State the criteria to choose algorithms

What is an Algorithm?

- An algorithm is a sequence of unambiguous instructions for solving a problem, i.e., for obtaining a required output for any legitimate input in a finite amount of time.

  Introduction to The Design & Analysis of Algorithms

  -Anany Levitin

- An algorithm is any well-defined computational procedure that takes some value, or set of values, as input and produces some value, or set of values, as output.

  Introduction to Algorithms

  -T. H. Cormen et. al.

What is an Algorithm?

- A clearly specified set of simple instructions to be followed to solve a problem or compute a function.

How to make a cake?

Step 1:..............................
....
Step 2: ....
Step 3: ....
....
....
Step 4:...............
Step 5:.........
Step 6: ....
Step 7: ....
Step 8:................

Reference: Reference Stock Vectors, Clipart and Illustrations." *123RF Stock Photos.* N.p., n.d. Web. 16 May 2016.

What is an Algorithm?

- A clearly specified set of simple instructions to be followed to solve a problem or compute a function.

How to make a cake?

- ✓ **Step 1:**.........................
  ....
- ✓ **Step 2:** ....
- ✓ **Step 3:** ....
  ....
  ....
- ✓ **Step 4:**...............
- ✓ **Step 5:**.........
- ✓ **Step 6:** ....
- ✓ **Step 7:** ....
- ✓ **Step 8:**...............

Reference: Reference Stock Vectors, Clipart and Illustrations." *123RF Stock Photos.* N.p., n.d. Web. 16 May 2016.

What is an Algorithm?

A **good algorithm** possesses the following properties:

1. **It must be correct.**
   - ✓ **Returns the correct results.**

How to make a cake?

- ✓ **Step 1:**.....................
- ✓ **Step 2:** ....
- ✓ **Step 3:** ....
  ....
- ✓ **Step 4:**...............
- ✓ **Step 5:**.........
- ✓ **Step 6:** ....
- ✓ **Step 7:** ....
- ✓ **Step 8:**...............

Reference: Reference Stock Vectors, Clipart and Illustrations." *123RF Stock Photos.* N.p., n.d. Web. 16 May 2016.

What is an Algorithm?

A **good algorithm** possesses the following properties:

2. **It must be composed of a series of concrete steps.**
   - ✓ **There can be no ambiguity as to which step will be performed next.**

How to make a cake?

- ✓ **Step 1:**.........................
  ....
- ✓ **Step 2:** ....
- ✓ **Step 3:** ....
  ....
  ....
  Step 4:...............
- ✓ **Step 5:**.........
- ✓ **Step 6:** ....
- ✓ **Step 7:** ....
- ✓ **Step 8:**...............

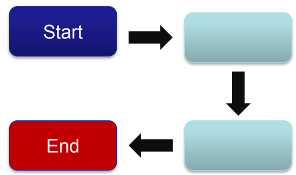
What is an Algorithm?

A **good algorithm** possesses the following properties:

3. **It must be composed of a finite number of steps.**
   - ✓ **It must terminate.**

How to make a cake?

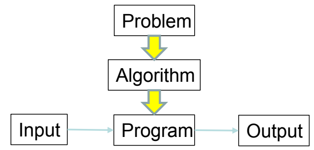
- ✓ **Step1:**.........................
  ....
- ✓ **Step 2:** ....
- ✓ **Step 3:** ....
  ....
  ....
  Step 4:...............
- ✓ **Step 5:**.........
- ✓ **Step 6:** ....
- ✓ **Step 7:** ....
- ✓ **Step 8:**...............



Computer Programs
Vs.
Algorithms

Computer Programs

- A **computer program** is an instance, or concrete representation of an algorithm in some programming language.
- **Programming constructs**
  - A composite collection of basic steps.
  - Commonly used ones:
    - ✓ for-loop, while-loop
    - ✓ if-else-statement, switch-case statement
    - ✓ functions (recursive and non-recursive)
- **Implementation** is the task of turning an algorithm into a computer program.



Design of Algorithms

There are often many ways (algorithms) to solve a problem.

Example 1: Summing up 1 to n

1+2+3+4+…….+n

Design of Algorithms

There are often many ways (algorithms) to solve a problem.

Example 1: Summing up 1 to n

1
```
Initialize sum to 0;
for (i = 1 to n)
add i to sum;
```

2
```
int sum;
sum = (1+n)*n/2;
```

3
```
int sum (int n);
{
      if (n is 1) return 1
      else return n + sum(n-1);
}
```

NANYANG TECHNOLOGICAL UNIVERSITY

Design of Algorithms

There are often many ways (algorithms) to solve a problem.

Example 1: Summing up 1 to n

Algorithm 1 Summing Arithmetic Sequence

function Method_One(n)
begin
$sum \leftarrow 0$
for $i = 1$ **to** n **do**
    $sum \leftarrow sum + i$
end

CE2001/ CZ2001: ALGORITHMS

NANYANG TECHNOLOGICAL UNIVERSITY

Design of Algorithms

There are often many ways (algorithms) to solve a problem.

Example 1: Summing up 1 to n

1

```
Initialize sum to 0;
for (i = 1 to n)
add i to sum;
```

2

```
int sum;
sum = (1+n)*n/2;
```

3

```
int sum (int n);
{
      if (n is 1) return 1
      else return n + sum(n-1);
}
```


CE2001/ CZ2001: ALGORITHMS

NANYANG TECHNOLOGICAL UNIVERSITY

Design of Algorithms

There are often many ways (algorithms) to solve a problem.

Example 1: Summing up 1 to n

Algorithm 2 Summing Arithmetic Sequence

function Method_Two(n)
begin
$sum \leftarrow n * (1 + n)/2$
end

CE2001/ CZ2001: ALGORITHMS

NANYANG TECHNOLOGICAL UNIVERSITY

Design of Algorithms

There are often many ways (algorithms) to solve a problem.

Example 1: Summing up 1 to n

1

```
Initialize sum to 0;
for (i = 1 to n)
add i to sum;
```

2

```
int sum;
sum = (1+n)*n/2;
```

3

```
int sum (int n);
{
      if (n is 1) return 1
      else return n + sum(n-1);
}
```

Design of Algorithms

There are often many ways (algorithms) to solve a problem.

Example 1: Summing up 1 to n

Algorithm 3 Summing Arithmetic Sequence

```
function Method_Three(n)
begin
if n=1 then
    return 1
else
    return  n+Method_Three(n − 1)
end
```


Design of Algorithms

There are often many ways (algorithms) to solve a problem.

Example 2: Fibonacci Sequence

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, …

Algorithm 4 Fibonacci Sequence: A Simple Recursive Function

```
function Fibonacci_Recursive(n)
begin
if n<1 then
    return 0
if n==1 OR n==2 then
    return 1
return  Fibonacci_Recursive(n − 1)+Fibonacci_Recursive(n − 2)
end
```

Algorithm 5 Fibonacci Sequence: A Simple Iterative Function

```
function Fibonacci_Iterative(n)
begin
if n<1 then
    return 0
if n==1 OR n==2 then
    return 1
F_2 ← 1
F_1 ← 1
for i = 3 to n do
    begin
    F_i ← F_{i−2} + F_{i−1}
    F_{i−2} ← F_{i−1}
    F_{i−1} ← F_i
    end
return F_n
end
```


Design of Algorithms

Example 3: How to compute the Sine function?

1. By storing and referring to trigonometry table.

Trigonometric Functions

	sin	cos	tan	cot	sec	csc	
0°	0.0000	1.0000	0.0000	···	1.000	···	90°
1°	0.0175	0.9998	0.0175	57.29	1.000	57.30	89°
2°	0.0349	0.9994	0.0349	28.64	1.001	28.65	88°
3°	0.0523	0.9986	0.0524	19.08	1.001	19.11	87°
4°	0.0698	0.9976	0.0699	14.30	1.002	14.34	86°
5°	0.0872	0.9962	0.0875	11.43	1.004	11.47	85°
6°	0.1045	0.9945	0.1051	9.514	1.006	9.567	84°
7°	0.1219	0.9925	0.1228	8.144	1.008	8.206	83°
8°	0.1392	0.9903	0.1405	7.115	1.010	7.185	82°
9°	0.1564	0.9877	0.1584	6.314	1.012	6.392	81°
10°	0.1736	0.9848	0.1763	5.671	1.015	5.759	80°
11°	0.1908	0.9816	0.1944	5.145	1.019	5.241	79°
12°	0.2079	0.9781	0.2126	4.705	1.022	4.810	78°
13°	0.2250	0.9744	0.2309	4.331	1.026	4.445	77°
14°	0.2419	0.9703	0.2493	4.011	1.031	4.134	76°
15°	0.2588	0.9659	0.2679	3.732	1.035	3.864	75°
16°	0.2756	0.9613	0.2867	3.487	1.040	3.628	74°
17°	0.2924	0.9563	0.3057	3.271	1.046	3.420	73°
18°	0.3090	0.9511	0.3249	3.078	1.051	3.236	72°
19°	0.3256	0.9455	0.3443	2.904	1.058	3.072	71°
20°	0.3420	0.9397	0.3640	2.747	1.064	2.924	70°
21°	0.3584	0.9336	0.3839	2.605	1.071	2.790	69°
22°	0.3746	0.9272	0.4040	2.475	1.079	2.669	68°
23°	0.3907	0.9205	0.4245	2.356	1.086	2.559	67°
24°	0.4067	0.9135	0.4452	2.246	1.095	2.459	66°
25°	0.4226	0.9063	0.4663	2.145	1.103	2.366	65°

Reference: Abelsson. (2006). Illustration of CORDIC operation. Retrieved from https://commons.wikimedia.org/wiki/File:CORDIC-illustration.png.

Design of Algorithms

Example 3: How to compute the Sine function?

1. By storing and referring to trigonometry table.
2. By expanding Maclaurin series.

$$\sin x = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)!} x^{2n+1} = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \cdots \textit{ for all } x$$

which is a special case of Taylor series with **a=0**.

$$f(x) = f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \frac{f^{(3)}(a)}{3!}(x-a)^3 \ldots$$

NANYANG TECHNOLOGICAL UNIVERSITY

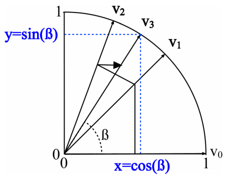
Design of Algorithms

Example 3: How to compute the Sine function?

3 For computers, easier to use **CORDIC**.

CORDIC (for **CO**ordinate **R**otation **DI**gital **C**omputer)

- To calculate hyperbolic and trigonometric functions.



Reference: Abelsson. (2006). Illustration of CORDIC operation. Retrieved from https://commons.wikimedia.org/wiki/File:CORDIC-illustration.png.

NANYANG TECHNOLOGICAL UNIVERSITY

Design of Algorithms

Example 3: How to compute the Sine function?

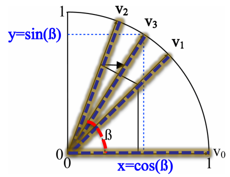
3 For computers, easier to use **CORDIC**.

$$v_i = Ki \begin{bmatrix} 1 & -\sigma_i 2^{-i} \\ \sigma_i 2^{-i} & 1 \end{bmatrix} \begin{bmatrix} x_{i-1} \\ y_{i-1} \end{bmatrix}$$

where

$$v_0 = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad K_i = \frac{1}{\sqrt{1 + 2^{-2i}}}$$

and σ_i = -1 or 1 depending on direction of rotation.



Reference: Abelsson. (2006). Illustration of CORDIC operation. Retrieved from https://commons.wikimedia.org/wiki/File:CORDIC-illustration.png.

NANYANG TECHNOLOGICAL UNIVERSITY

How do we choose among the different algorithms?

NANYANG TECHNOLOGICAL UNIVERSITY

Choosing the Right Algorithm

At the heart of computer program design, there are two (sometimes conflicting) goals:

1) To design an algorithm that is easy to understand, code and debug.

❖ **Software Engineering**

2) To design an algorithm that makes efficient use of the computer's resources.

❖ **Data structures and algorithm analysis**

Choosing the Right Algorithm

In this course, we favour the most efficient algorithms.

❖ **One that uses the least amount of resources**

Summary

- What an Algorithm is?
- Computer programs vs. Algorithms
- How do we choose among the different algorithms?