មហាវិទ្យាល័យវិស្វកម្ម
FACULTY OF ENGINEERING

Data Structure & Algorithm II

Lecture 8
Time Complexity

Chhoeum Vantha, Ph.D.
Telecom & Electronic Engineering

Content

- The Software Development Process
- Performance Analysis: the Big Oh.
- Abstract Data Types
- Introduction to Data Structures

Software Development

- Requirement analysis, leading to a specification of the problem
- Design of a solution
- Implementation of the solution (coding)
- Analysis of the solution
- Testing, debugging and integration
- Maintenance and evolution of the system.

Software Development

Time complexity

- **Time complexity** is a type of computational complexity that describes the time required to execute an algorithm.
- The time complexity of an algorithm is the amount of time it takes for each statement to complete.
- It is highly dependent on the size of the processed data

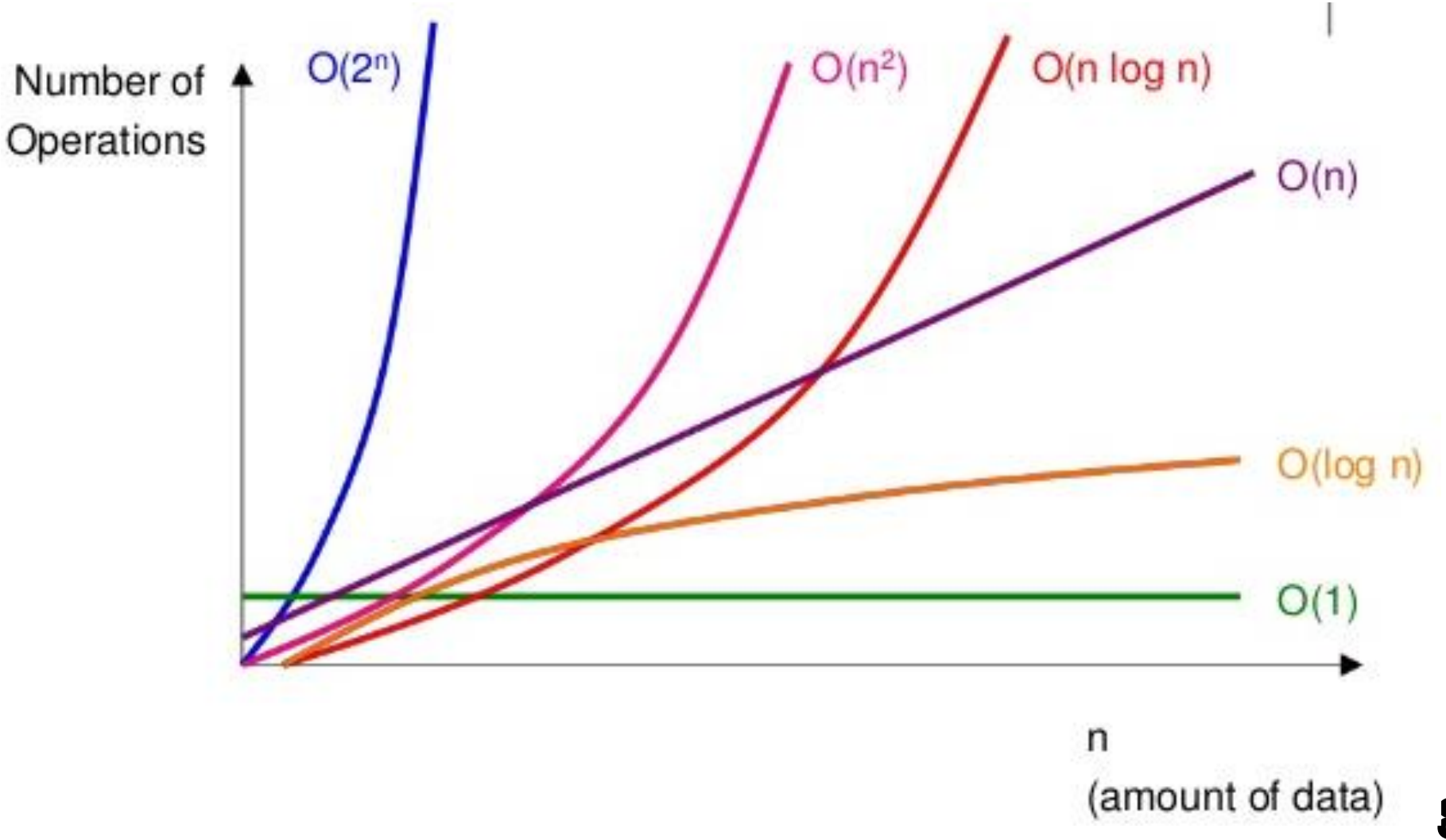
Big-O Notation

- $T(n)=O(1)$ // constant time
- $T(n)=O(\log n)$ // logarithmic
- $T(n)=O(n)$ // linear
- $T(n)=O(n^2)$ //quadratic
- $T(n)=O(n^3)$ //cubic
- $T(n)=O(n^c)$, $c \geq 1$ // polynomial
- $T(n)=O(\log^c n)$, $c \geq 1$ // polylogarithmic
- $T(n)=O(n\log n)$

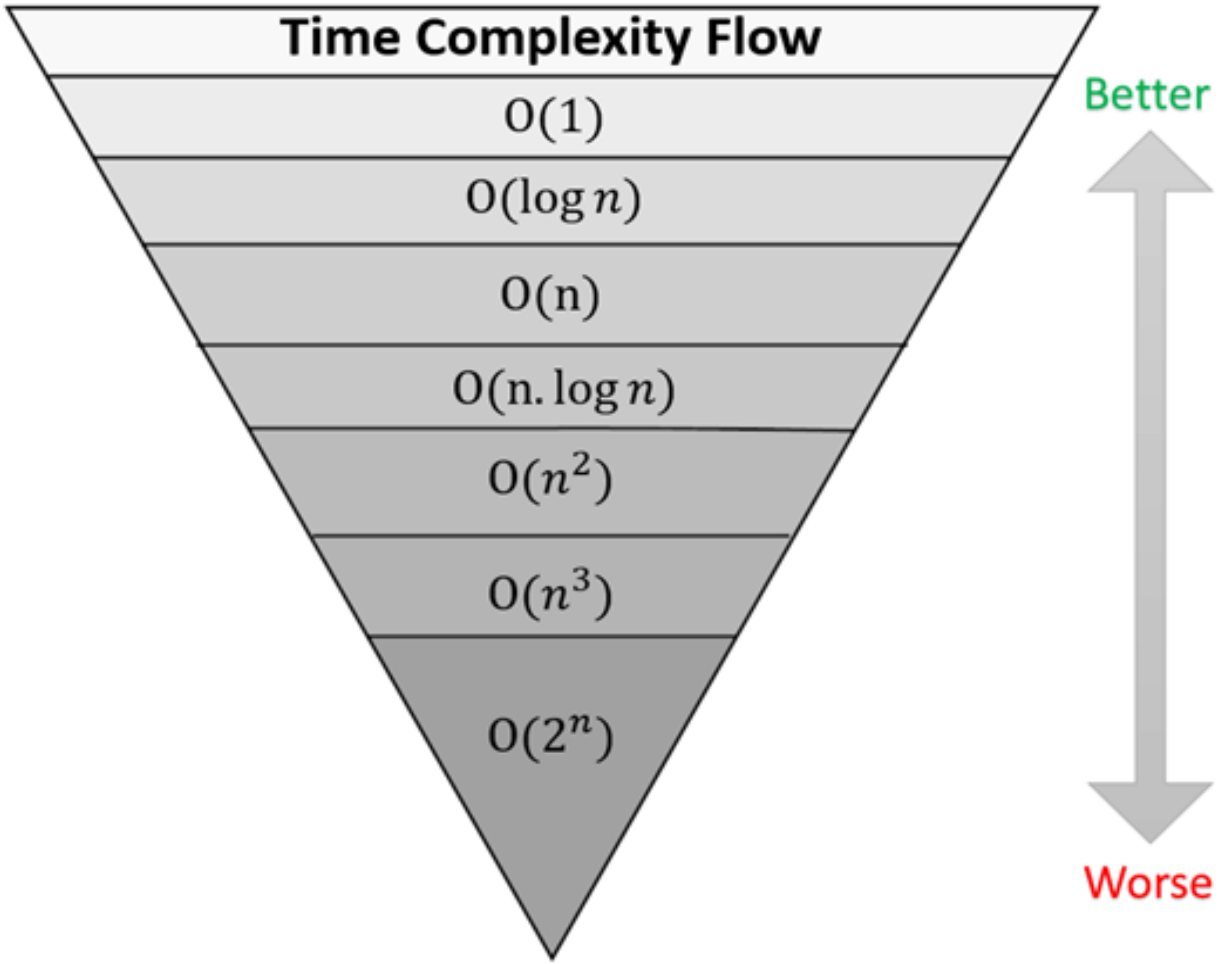
Big-O Notation

- O(1) – **constant** time, the time is independent of n, e.g. array look-up
- O(log n) – **logarithmic** time, usually the log is base 2, e.g. binary search
- O(n) – **linear** time, e.g. linear search
- O(n*log n) – e.g. efficient sorting algorithms
- O(n^2) – **quadratic** time, e.g. selection sort
- O(n^k) – **polynomial** (where k is some constant)
- O(2^n) – **exponential** time, very slow!

Comparing Big O Functions



Comparing Big O Functions



Time complexity

Data Structure	Worst Case Time Complexity			
	Access	Search	Insertions	Delete
Array	O(1)	O(n)	O(n)	O(n)
Stack	O(n)	O(n)	O(1)	O(1)
Queue	O(n)	O(n)	O(1)	O(1)
Singly Linked List	O(n)	O(n)	Begin: O(1), End: O(n)	Begin: O(1), End: O(n)
Doubly Linked List	O(n)	O(n)	Begin: O(1), End: O(n)	Begin: O(1), End: O(n)
Binary Search Tree	O(n)	O(n)	O(n)	O(n)
B-Tree	O(log(n))	O(log(n))	O(log(n))	O(log(n))
AVL Tree	O(log(n))	O(log(n))	O(log(n))	O(log(n))

Data Structures

- A data structure is a user-defined abstract data type
- Examples:
  - **Complex numbers**: with operations +, -, /, *, *magnitude*, *angle*, etc.
  - **Stack**: with operations *push, pop, peek, isempty*
  - **Queue**: *enqueue, dequeue, isempty …*
  - **Binary Search Tree**: *insert, delete, search.*
  - **Heap**: *insert, min, delete-min*.

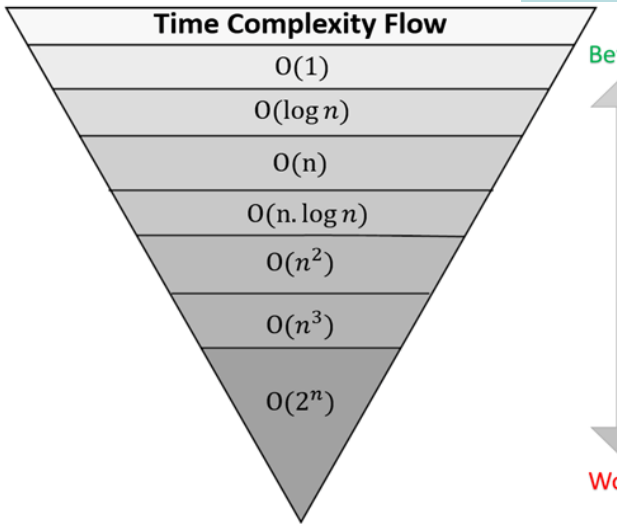
Data Structures

- Specification
  - A set of data
  - Specifications for a number of operations to be performed on the data
- Design
  - A lay-out organization of the data
  - Algorithms for the operations
- Goals of Design: **fast** operations

Time Complexity with Simple Examples

- How is the time complexity of program below:

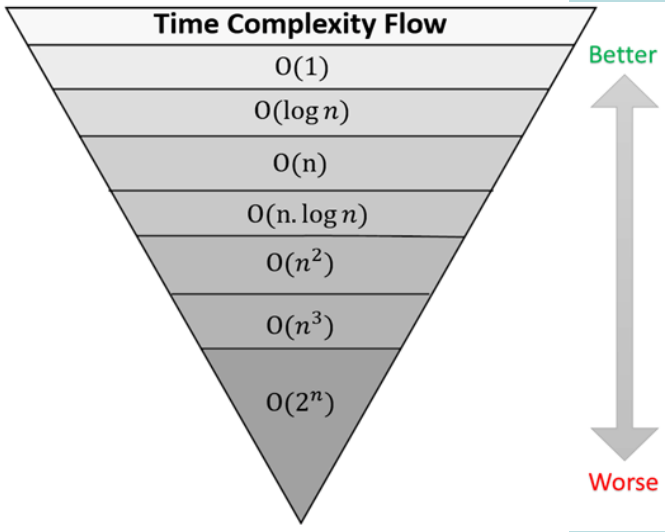
```
1   #include <iostream>
2   using namespace std;
3   int main()
4   {
5       cout << "Hello World";
6       return 0;
7   }
```



Time Complexity with Simple Examples

- How is the time complexity of program below:

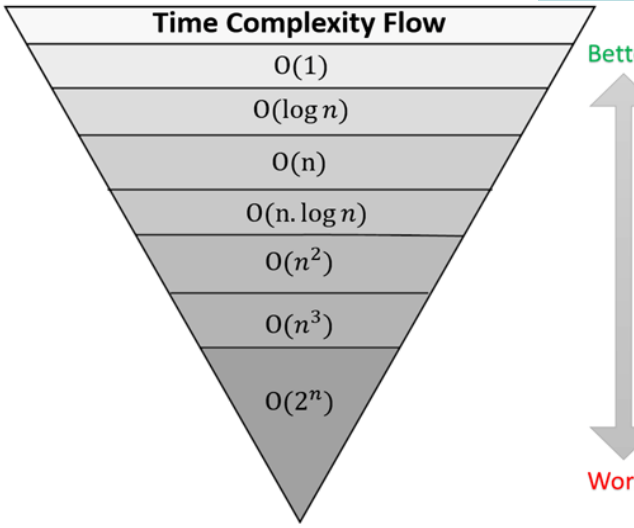
```
1   #include <iostream>
2   using namespace std;
3   int main()
4   {
5       int i, n = 8;
6       for (i = 1; i <= n; i++)
7       {
8           cout << "Hello World !!!\n";
9       }
10      return 0;
11  }
```



Time Complexity with Simple Examples

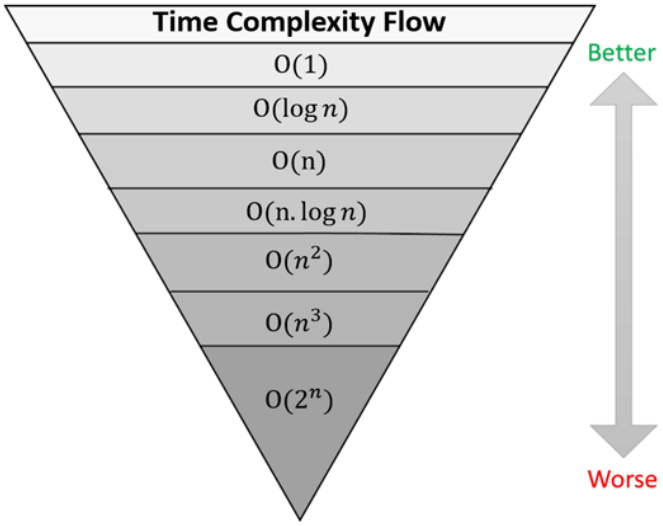
- How is the time complexity of program below:

```
1   #include <iostream>
2   using namespace std;
3   int main()
4   {
5       int i, n = 8;
6       for (i = 1; i <= n; i=i*2) {
7           cout << "Hello World !!!\n
8       }
9       return 0;
10  }
```



Time Complexity with Simple Examples

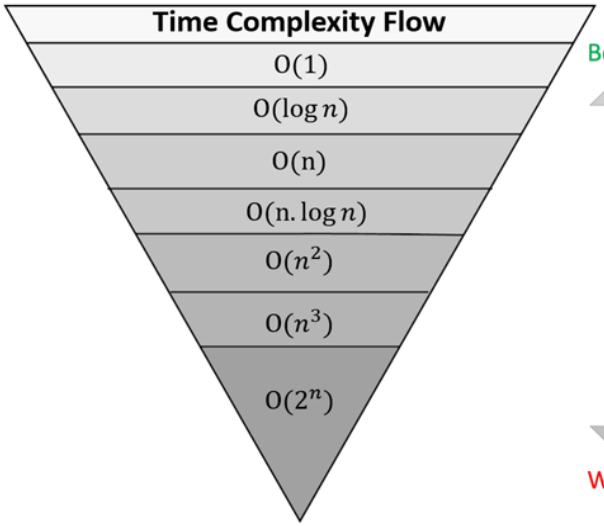
- How is the time complexity of program below:




```
1   #include <iostream>
2   using namespace std;
3   int main()
4   {
5   int a = 0, b = 0;
6   int n = 5;
7   for(int i=0;i<n;i++)
8   {
9       a = a +rand();
10      cout << a << " ";
11  }
12  cout << endl;
13  for(int i=0;i<n;i++)
14  {
15      b = b +rand();
16      cout << b << " ";
17  }
18      return 0;
19  }
```

Time Complexity with Simple Examples

- How is the time complexity of program below:




```
1   #include <iostream>
2   using namespace std;
3   int main()
4   {
5   int value = 0;
6   int n = 5;
7   for(int i=0;i<n;i++)
8   {
9       for(int j=0;j<n;j++)
10      {
11        value += 1;
12        cout<< value << " ";
13      }
14  }
15    return 0;
16  }
```

W8 – Lab

Ex. 1

Write a **Function** code in C++ to implement on time complexity of:

a) Constant time — O(1)

b) Logarithmic time — O(log n)

c) Linear time — O(n)

d) Quadratic time — $O(n^2)$

e) Linear time — O(n + m)

Ex. 2

You are given an integer n.

Count the total of 1 + 2 + . . . + n

Now try to write the function in C++ with could meet 2-time complexity

a) Slow solution — time complexity $O(n^2)$

b) Fast solution — time complexity O(n)

Thanks!