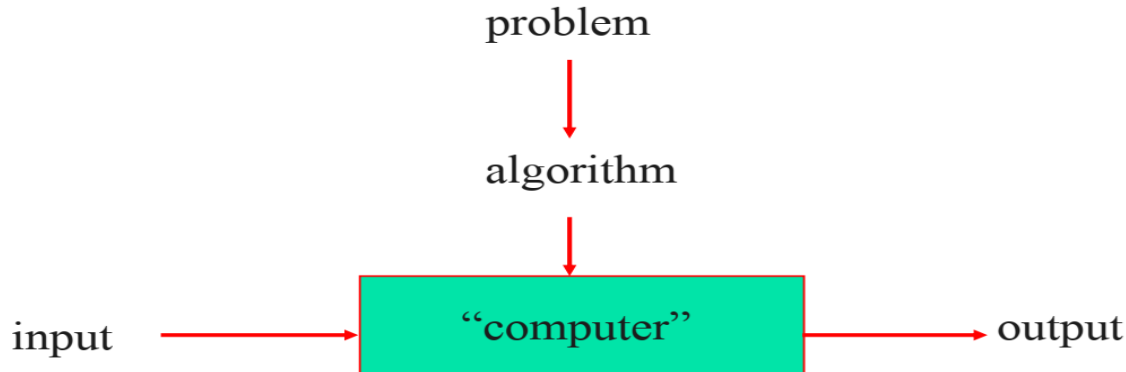


CSE 221 Algorithms: Ch1

Instructor: Saraf Anika
CSE, SoSET, EDU
Spring 2022

What is Algorithms?

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



Algorithm VS Program

- Do all kinds of computer program requires Algorithm?

Algorithm	Program
An algorithm is a systematic approach to solving a specific problem.	A program is a set of instructions for a computer to follow.
Written at Design time	Written at Implementation time
Person with Domain Knowledge	Programmer
Any Language i.e English, Mathematical Notations	Programming Language(C, Python, Java etc.)
H/W & OS independent	H/W & OS dependent
Analyze whether the goal has been met: Priori Analysis	Testing for errors: Posteriori Testing
Time & Space Function	Watch Time & Bytes

Why Study Algorithms?

- Al Khawarizmi

“A great Iranian mathematician, geographer and astronomer. He introduced the zero, negative numbers, algebra, and the decimal system to the West. He also invented mathematical programming using a set of instructions to perform complex calculations. The term algorithm is named after a variation of his name, Algorithmi”.

- Cornerstone of computer science.
- Building block of programming. Used to identify which steps are necessary to implement in a program.
- Algorithms like safe-search, indexing algorithm, Google instant, and many more are used in the Google search engine. Most of the time, there are multiple algorithms in a big application that work together to provide us with the best result.

Example of Algorithms: Sorting

- Statement of problem:
 - ❖ Input: A sequence of n numbers $\langle a_1, a_2, \dots, a_n \rangle$
 - ❖ Output: A reordering of the input sequence $\langle a'_1, a'_2, \dots, a'_n \rangle$ so that $a'_i \leq a'_j$ whenever $i < j$
 - ❖ Instance: The sequence $\langle 5, 3, 2, 8, 3 \rangle$
- Different types of sorting Algorithms: Selection sort, Insertion sort, Bubble sort etc.

Properties of Algorithms

What distinguish an algorithm from a recipe, process, method, technique, procedure, routine...?

- **Finiteness**

Terminates after a finite number of steps i.e stops when the required output is achieved

- **Definiteness**

Algorithms must determine each step and each of its steps should be clear in all behaviors and must direct to only one meaning: clear and unambiguous.

- **Input**

Valid inputs must be clearly specified.

- **Output**

The data that result upon completion of the algorithm must be specified.

- **Effectiveness**

The algorithm should be effective which implies that all those means that are needed to get to output must be feasible with the accessible resources.

How to Write An Algorithm?

Pseudocode for swapping two numbers:

BEGIN

#define variables and assign values to variables
NUMBER num1=10, num2=15, temp

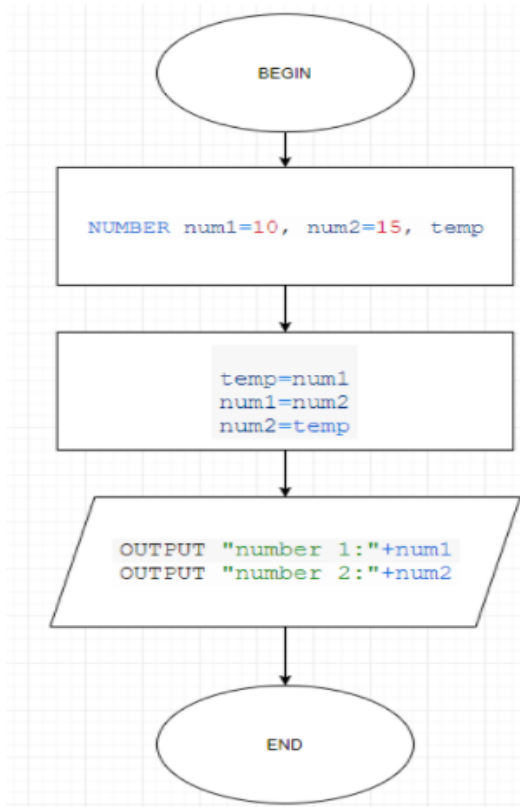
#assign num1 to temp
temp=num1

#assign num2 to num1
num1=num2

#assign temp to num2
num2=temp

OUTPUT "number 1:" + num1
OUTPUT "number 2:" + num2

END



```
Algorithm Swap(a, b)
{
    temp = a;
    a = b;
    b = temp;
}
```

The image shows a handwritten version of the swap algorithm. It starts with 'Algorithm Swap(a, b)' followed by an opening curly brace. The next three lines are 'temp = a;', 'a = b;', and 'b = temp;'. The algorithm ends with a closing curly brace.

How to Analyze an Algorithm?

- Time
 - How much time is it taking?
 - Will get the time measurement as function [P.S. Not your clock time!]
- Space
 - How much memory space is it going to consume while running in the machine
- N/W
 - How much data transfer is done
- Power Consumption
 - How much power is it going to consume in our handheld devices(Mobile/Tablet PCs etc.)
- CPU Register
 - How many registers is it going to consume to execute as a program

How to Analyze an Algorithm? Cont.

- a,b,temp are considered as Word, not bytes
- Every **simple** statement of an algorithm takes 1 unit of time
- $O(1)$ means constant

Algorithm Swap(a, b)

```
{
    temp = a;
    a = b;
    b = temp;
}
```

Time Complexity: $f(n) = 3$

Space Complexity:

a	—	1
b	—	1
temp	—	1
		<hr/>
		$S(n) = 3$

Frequency Count Method

8	3	9	7	2
0	1	2	3	4

 $n=5$

space

A — n
n — 1
s — 1
i — 1

$\left. \begin{array}{l} A \\ n \\ s \\ i \end{array} \right\} S(n) = n + 3$
 $O(n)$

Algorithm Sum(A, n)

```
{  
  s = 0; _____ 1  
  for (i = 0; i < n; i++) _____ n+1  
  {  
    s = s + A[i]; _____ n  
  }  
  return s; _____ 1  
}
```


$f(n) = 2n + 3$
 $O(n)$

Frequency Count Method Cont.

An algorithm to add two matrices A and B of size n

Algorithm Add(A, B, n)

$n \times n$
 3×3



```
for(i=0; i<n; i++)  
{  
    for(j=0; j<n; j++)  
    {  
        C[i, j] = A[i, j] + B[i, j];  
    }  
}
```

Frequency Count Method Cont.

Space Complexity:

$$A = n^2$$

$$B = n^2$$

$$C = n^2$$

$$n = 1$$

$$i = 1$$

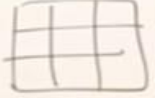
$$j = 1$$

$$S(n) = 3n^2 + 3$$

$$O(n^2)$$

Algorithm Add(A, B, n)

$n \times n$
 3×3



for (i=0; i<n; i++) ——— $n+1$

for (j=0; j<n; j++) ——— $n \times (n+1)$

$C[i, j] = A[i, j] + B[i, j];$ — $n \times n$

$f(n) = 2n^2 + 2n + 1$

$O(n^2)$

Frequency Count Method Cont.

Do it yourself: Find out Time
and Space Complexity

```
Algorithm Multiply(A, B, n)
{
    for(i=0; i<n; i++)
    {
        for(j=0; j<n; j++)
        {
            c[i, j]=0;
            for(k=0; k<n; k++)
            {
                c[i, j]=c[i, j]+A[i, k]*B[k, j];
            }
        }
    }
}
```

Time Complexity

Find out Time Complexity of this Algorithm.

③

```
for(i=0; i<n; i++)  
{  
    for(j=0; j<i; j++)  
    {  
        stmt;  
    }  
}
```

Time Complexity Cont.

```
for(i=0; i<n; i++)
```

```
{
```

```
    for(j=0; j<i; j++)
```

```
    {
```

```
        stmt; _____
```

```
    }
```

```
}
```

$$1 + 2 + 3 + \dots + n = \frac{n(n+1)}{2}$$

$$f(n) = \frac{n^2 + 1}{2}$$

$$O(n^2)$$

i	j	no of time
0	0x	0
1	0✓ 1x	1
2	0 1 2x	2
3	0 1 2 3	3
⋮	⋮	⋮
n		n

Time Complexity Cont.(DIY)

for($i=1; i < n; i=i*2$)
{
 stmt;
}

Assume $i \geq n$
 $\therefore i = 2^k$
 $\therefore 2^k \geq n$
 $2^k = n$
 $k = \log_2 n$

i
1
 $1 \times 2 = 2$
 $2 \times 2 = 2^2$
 $2^2 \times 2 = 2^3$
:
:
 2^k

$O(\log_2 n)$

$p=0;$
for($i=1; p \leq n; i++$)
{
 $p=p+i;$
}

Assume $p > n$
 $\therefore p = \frac{k(k+1)}{2}$
 $\frac{k(k+1)}{2} > n$
 $k^2 > n$
 $k > \sqrt{n}$

i p
1 $0+1=1$
2 $1+2=3$
3 $1+2+3$
4 $1+2+3+4$
:
:
:
 k $1+2+3+4+\dots+k$

$O(\sqrt{n})$

Using 'if' and 'while'

```
i = 0;            1
while (i 10 < n)        n+1
{
    stmt;        n
    i++;        n
}
```

$$f(n) = 3n + 2$$

$O(n)$

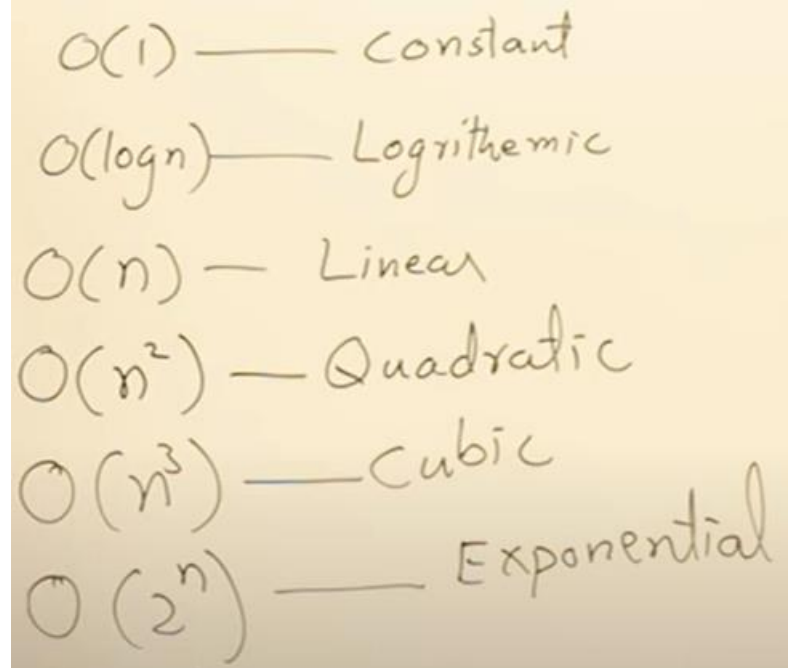
```
while (m != n)
{
    if (m > n)
        m = m - n;
    else
        n = n - m;
}
```

<u>m=16</u>	<u>n=2</u>	
14	2	
12	2	
10	2	
8	2	$\frac{16}{2}$
6	2	
4	2	$\frac{n}{2}$
2	2	2

min $O(1)$ $O(n)$

Time Function Summary

- We might come across the Exponential type of Time Function in the upcoming classes.



$O(1)$	Constant
$O(\log n)$	Logarithmic
$O(n)$	Linear
$O(n^2)$	Quadratic
$O(n^3)$	Cubic
$O(2^n)$	Exponential

Compare Time Function

$$1 < \log n < \sqrt{n} < n < n \log n < n^2 < n^3 < \dots < 2^n < 3^n \dots < n^n$$

$\log n$	n	n^2	2^n
0	1	1	<u>2</u>
$\log_2^2 = 1$	2	<u>4</u>	4
2	4	<u>16</u>	16
3	8	64	<u>256</u>
3.1	9	81	512