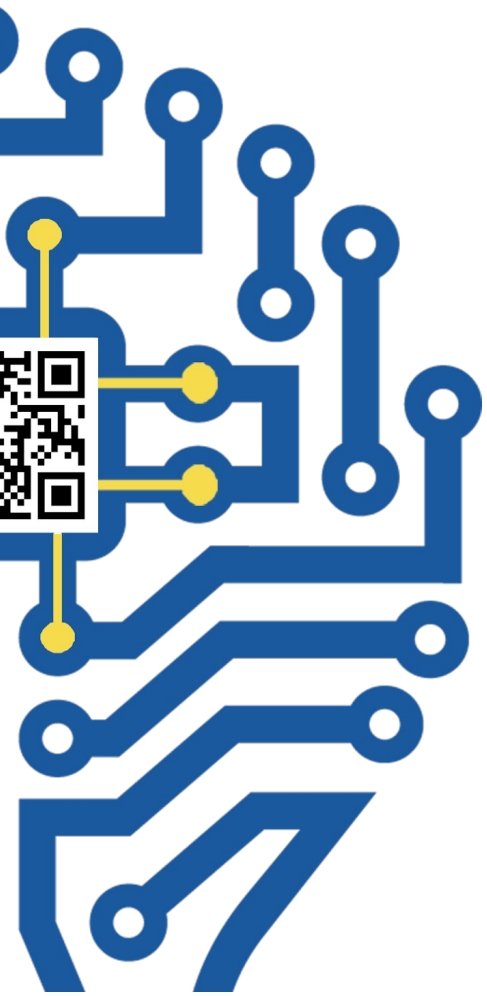


UGANDA CHRISTIAN UNIVERSITY
A Center of Excellence in the Heart of Africa

CSC2107: Design and Analysis of Algorithms

BY

JB Wabwire Habere & Ayebare Moses

Course Content

- Python review
- Intro to DAA
- Asymptotic Analysis
- Searching and Sorting
- Graph Algorithms
- Divide and Conquer
- Data Structures
- Search Trees
- Greedy Algorithms
- Dynamic Programming
- Extra Topics (if time is on our side)

Assessment

1 SAMUEL 7:12

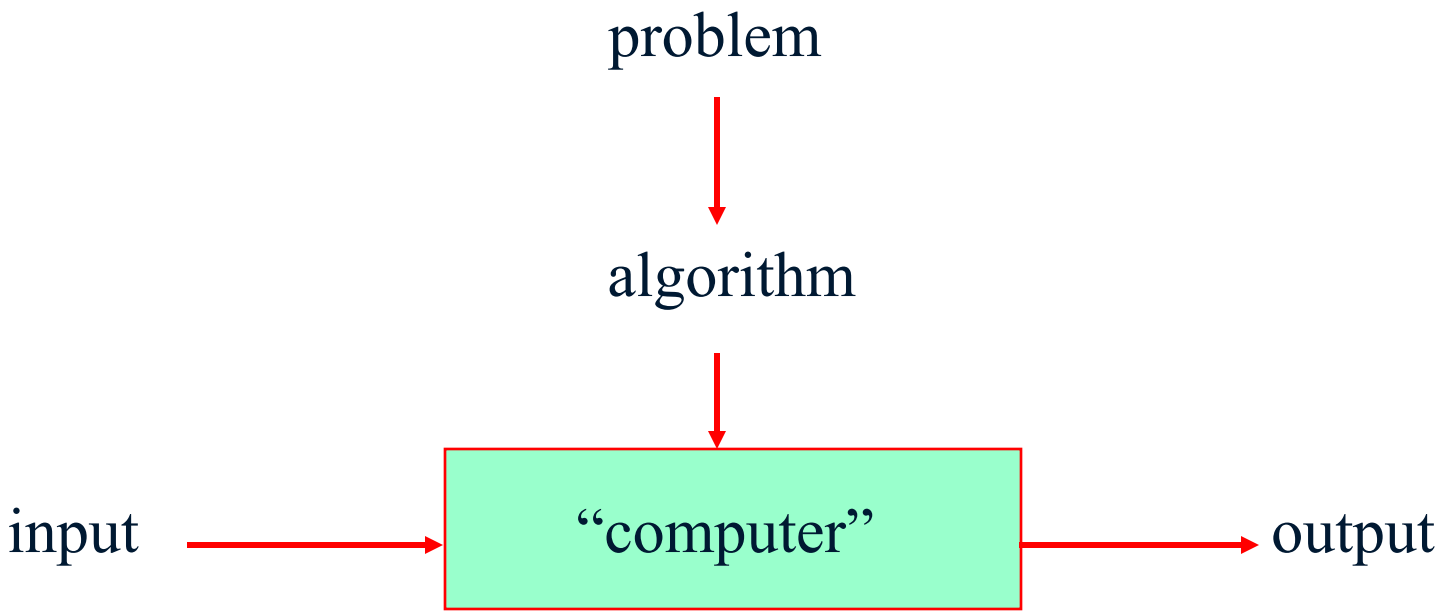
SN	Assessment	Mark %
1.	Assignments	15
2.	Quizzes	10
3.	Tests	20
4.	Project	10
5	Exam	40
6	Attendance	05
TOTAL		**100**

Resources

- Primary
  - Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Cliffo - *Introduction to Algorithms* (2022, The MIT Press)
  - Levitin, Anany - *Introduction to the design and analysis of algorithms*-Pearson (2019)
- Secondary
  - Kumar, Amit_ Sen, Sandeep - *Design and analysis of algorithms_ a contemporary perspective*-Cambridge University Press (2019)
  - Arthur Nunes - *Introduction to the Design and Analysis of Algorithms - A Multi-Paradigm Approach*-Kendall Hunt (2022)
  - Any Python/C/C++/Java Books for Algorithm Implementation

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.



Why study algorithms?

- Theoretical importance
  - the core of computer science
- Practical importance
  - A practitioner's toolkit of known algorithms
  - Framework for designing and analyzing algorithms for new problems

Two main issues related to algorithms

- How to design algorithms

- How to analyze algorithm efficiency

Algorithm design techniques/strategies

- Brute force
- Divide and conquer
- Decrease and conquer
- Transform and conquer
- Space and time tradeoffs
- Greedy approach
- Dynamic programming
- Iterative improvement
- Backtracking
- Branch and bound

Analysis of algorithms

- How good is the algorithm?
  - time efficiency
  - space efficiency

- Does there exist a better algorithm?
  - lower bounds
  - optimality

Important problem types

1 SAMUEL 7:12

- Sorting e.g. insertion sort
- searching
- string processing
- graph problems
- combinatorial problems
- geometric problems
- numerical problems

Fundamental data structures

- list
  - array
  - linked list
  - string
- stack
- queue
- priority queue
- graph
- tree
- set and dictionary

Fast computers *vs* efficient algorithms

- Many recent innovations rely on
  - fast computers
  - efficient algorithms.

- Which is more important?

Algorithm Example

1 SAMUEL 7:12

Arrays and its representations

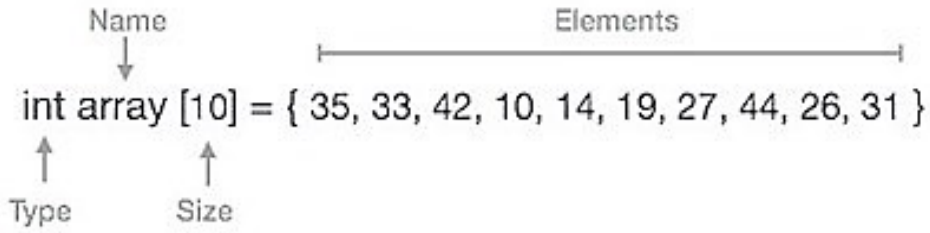
Arrays

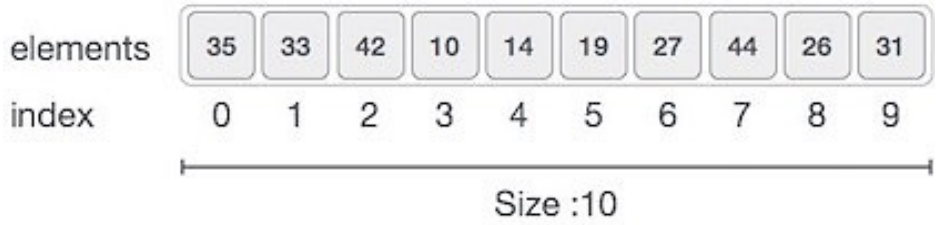
An array is a container that can hold a fixed number of items, and these items should be of the same type.

- **Element** – Each item stored in an array is called an element.
- **Index** – Each location of an element in an array has a numerical index, which is used to identify the element.

Array Representation

- Arrays can be declared in various ways in different languages. For illustration, let's take the C array declaration.





Basic Operations

Following are the basic operations supported by an array.

- **Traverse** – print all the array elements one by one.
- **Insertion** – Adds an element at the given index.
- **Deletion** – Deletes an element at the given index.
- **Search** – Searches an element using the given index or by the value.
- **Update** – Updates an element at the given index.

Search Operation

- You can perform a search for an array element based on its value or its index.

Pseudocode

```
Procedure Search()
Begin procedure
ITEM // what we are searching for
 Set i = 0
while i < N
Begin
 if A[i]= ITEM
   Print “Element found at the position” i
  i = i +1
End While
End procedure
```

```c
#include <stdio.h>
int main() {
   int  arr[] = { 1,3,5,7,8 };
   int item = 7, n = 5;
   int j = 0;
```

```c
while (j < n) {
      if (arr[j] == item) {
         break;
      }
      j = j + 1;
   }
   printf("Found element %d at position %d\n", item, j);
}
```

Exercise – To Do

- Write an Algorithm to find the maximum element in an Array
- Implement the algorithm with a C/C++ code.

- Upload your solution on GitHub and share your link via Discord DM.