DESIGN AND ANALYSIS OF ALGORITHM (20CP209T)

Presented by:

Dr. Shivangi K. Surati

Assistant Professor,
Department of Computer Science and Engineering,
School of Technology,
Pandit Deendayal Energy University

Outline

- Course Introduction- Teaching Scheme
- Syllabus
- Text/Reference Books
- Prerequisite
- OBE (Outcome Based Education), PEOs, POs, COs
- Traditional Education v/s OBE
- Pedagogy
- Office Hour
- What is an Algorithm?
- Characteristics of an Algorithm
- Analysis of an Algorithm

Course Introduction- Teaching Scheme

20CP209T					Design and Analysis of Algorithm					
Teaching Scheme					Examination Scheme					
L	T	P	C	Hrs/Week	Theory			Practical		Total Marks
					MS	ES	IA	LW	LE/Viva	
3	0	0	3	3	25	50	25	-	-	100

Course Objectives:

- Analyze the asymptotic performance of the algorithms.
- Implement time and space efficient optimized algorithms.
- Demonstrate a familiarity with major algorithms and data structures.
- Apply important algorithmic design paradigms and methods of analysis.

Prepared by: Dr. Shivangi Surati

Syllabus

UNIT 1 INTRODUCTION Elementary Algorithmic: Efficiency of Algorithms, Average & worst-case analysis, Elementary Operation Analysis Techniques. Analyzing control structures, Amortized analysis	**10 Hrs.**
UNIT 2 RECURRANCE AND GREEDY ALGORITHMS Intelligent guesswork, Homogeneous recurrences, Inhomogeneous Recurrences, Change of variable, Master Theorem, Recurrence Tree. Greedy Algorithms: Graphs: Minimum spanning trees-Kruskal's algorithm, Prim's algorithm, Graphs: Shortest paths.	**10 Hrs.**
UNIT 3 DIVIDE & CONQUER AND DYNAMIC PROGRAMMING Divide-and-Conquer: Multiplying large integers, Binary search, finding the median, Matrix multiplication, Exponentiation. Dynamic Programming: Making Change, The principle of optimality, The Knapsack Problem, Shortest path, Chained matrix multiplication.	**10 Hrs.**
UNIT 4 BACKTRACKING, BRANCH & BOUND, NP THEORY Design of some classical problems using branch and bound and Backtracking approaches. Brief Overview of NP theory, approximation algorithms.	**9 Hrs.**
	Max. 39 Hrs.

Text/Reference Books

- Charles E. Leiserson, Thomas H. Cormen, Ronald L. Rivest, Clifford Stein - Introduction to Algorithms, PHI
- Gilles Brassard & Paul Bratley, Fundamentals of Algorithmic, PHI
- Ellis Horowitz, Sartaj Sahni, Sanguthevar Rajasekharan, Fundamentals of Computer Algorithms, Galgotia.

Prerequisite

- Fundamental knowledge of Data Structures
- Any Programming Language Course

OBE (Outcome Based Education)

- An education in which an emphasis is placed on
  - Clarity of Focus: a clearly articulated idea of what students are expected to know and be able to do
  - Designing down: the curriculum design must start with a clear definition of the intended outcomes about skills and knowledge students need to have, when they leave the school system
  - High expectations: high and challenging standards of performance in order to encourage students to engage deeply in what they are learning
  - Expanded opportunities: provide expanded opportunities for all students.
- Students are assisted when and where they have challenges.

OBE (Outcome Based Education)…

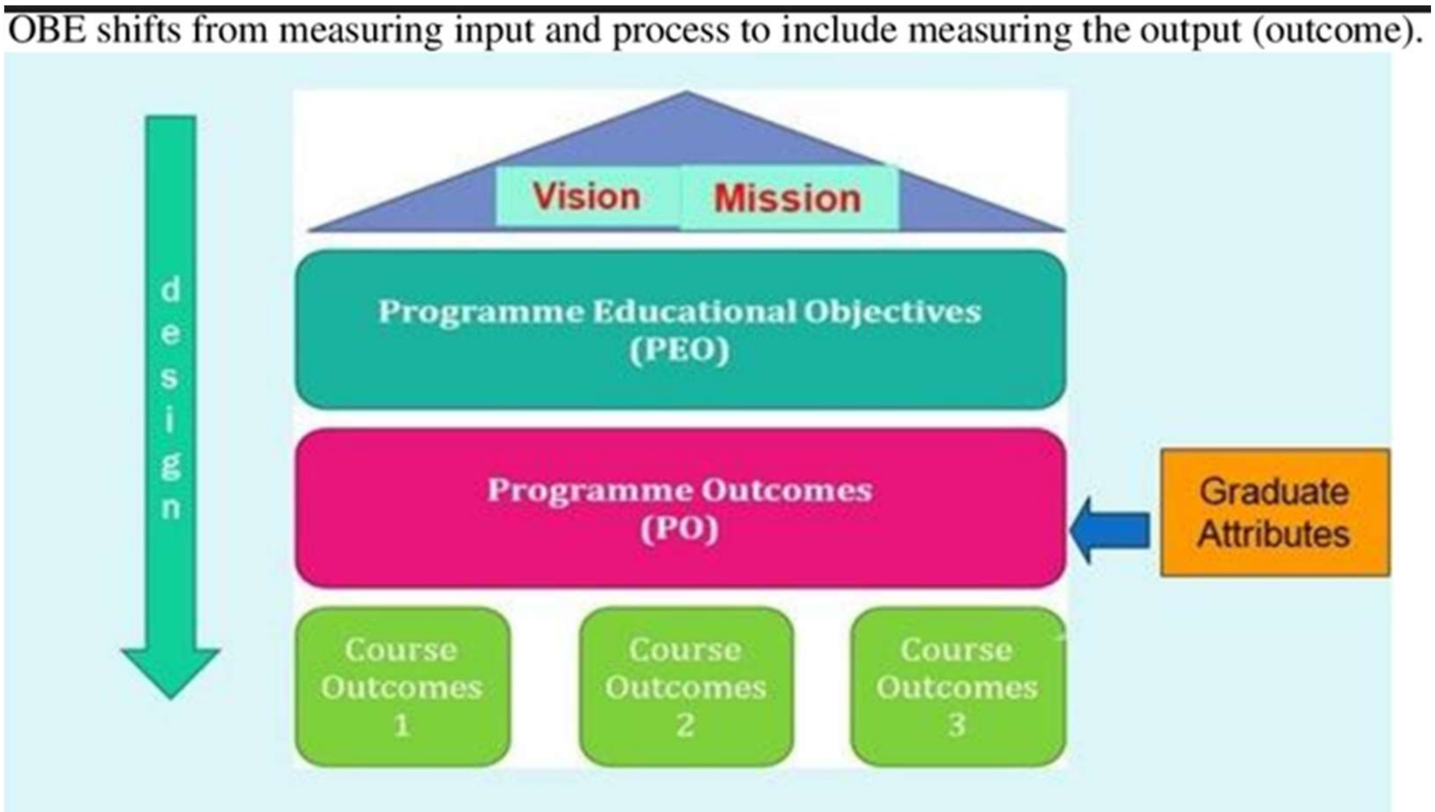


Image Source: "Importance of Outcome Based Education (OBE) to Advance Educational Quality and enhance Global Mobility." (2018).

- 4 PEOs, 12 POs and 6 COs

Program Education Objectives (PEOs)

1. To prepare graduates who will be successful professionals in industry, government, academia, research, entrepreneurial pursuit and consulting firms
2. To prepare graduates who will make technical contribution to the design, development and production of computing systems
3. To prepare graduates who will get engage in lifelong learning with leadership qualities, professional ethics and soft skills to fulfill their goals
4. To prepare graduates who will adapt state of the art development in the field of computer engineering

Prepared by: Dr. Shivangi Surati

Program Outcomes (POs)

1. Engineering knowledge
2. Problem analysis
3. Design / development of solutions
4. Conduct investigations of complex problems
5. Modern tool usage
6. The engineer and society
7. Environment and sustainability
8. Ethics
9. Individual and team work
10. Communication
11. Project management and finance
12. Life-long learning

Prepared by: Dr. Shivangi Surati

Program Specific Outcomes (PSOs)

- The graduates of CSE department will be able to:

1. Develop computer engineering solutions for specific needs in different domains applying the knowledge in the areas of programming, algorithms, hardware-interface, system software, computer graphics, web design, networking and advanced computing.
2. Analyze and test computer software designed for diverse needs.
3. Pursue higher education, entrepreneurial ventures and research.

Course Outcomes (COs)

- On completion of the course, student will be able to
  - CO1- Understand need of complexity analysis of the algorithm
  - CO2- Solve Homogenous and Inhomogeneous recurrence relations using Master Theorem, Substitution method, and Recurrence tree.
  - CO3- Apply Dynamic Programming, Divide and Conquer Strategy and greedy method to solve computational and graph problems.
  - CO4- Compare different algorithmic Strategies on efficiency parameters for optimization problems.
  - CO5- Evaluate Classical problems through Backtracking and Branch & Bound techniques.
  - CO6- Create algorithms for real time problems. Design algorithms for computational problems of moderate complexity.

Traditional Education v/s OBE

Traditional Education	Outcome Based Education
The approach is exam-driven.	Students are assessed on an ongoing basis.
Learning is textbook/worksheet-bound, subjective and teacher-centered	Learning is objective learner-centered, the teacher facilitates and constantly applies group work and team work for new tasks
Learners are passive	Learners are active
Emphasis on what lecturer hopes to achieve	Emphasis on specific outcomes
Rote learning	Critical thinking, reasoning and action
Textbook/worksheets focused and teacher centered	Learner centered and educator use group/team work

Pedagogy

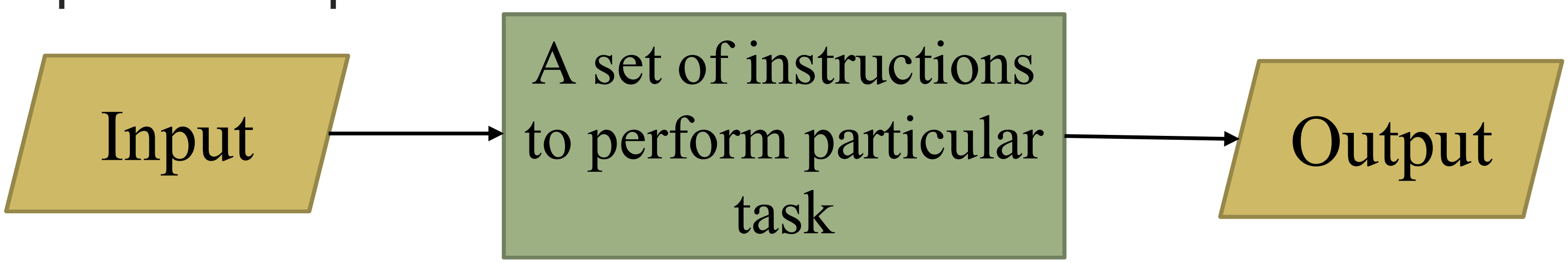
- Use of
  - Board
  - Powerpoint Presentations
  - Puzzles
  - Program Execution

Office Hour

- Tuesday, 9:30 AM to 11:00 AM
- Take prior permission through mail and then come to meet in E-215 faculty cabin

What is an Algorithm?

- A set of commands (steps) that must be followed for a computer to perform calculations or other problem-solving operations.
- A finite set of instructions carried out in a specific order to perform a particular task.



Algorithm

- **Example:** sorting

  input: A sequence of numbers.

  output: An ordered permutation of the input.

  issues: correctness, efficiency, storage, etc.

Characteristics of an Algorithm

- Input - Zero or more quantities are externally supplied
- Output - At least one quantity is produced
- Finiteness - If we trace out an algorithm, then for all cases, the algorithm terminates after a finite number of steps.
- Definiteness- Each instruction is clear and Unambiguous. Proper order of sequence of instructions.
- Effectiveness - Every instruction must be very basic so that it can be carried out feasibly.
- Language independence

Analysis of Algorithm

- **RAM (Random Access Machine) model**
- The rules or the instructions of the RAM model are:

1. Arithmetic operations like addition(+), subtraction(-), multiplication(*), division(/), floor, ceiling, and assignment operation etc. take a constant amount of time.
2. Memory access like read, save, copy, etc. take a constant amount of time.
3. Subroutine calls, control, return, etc. take a constant amount of time.

Analysis of Algorithm…

- Each operation in an algorithm
  - takes certain time
  - has a cost (EX: i = i+1) Cost: C1 (constant)
- A sequence of operations
  - i = i+1 Cost:C1 (constant)
  - a = i + 2 Cost:C2 (constant)
  - Total cost = c1 + c2

Prepared by: Dr. Shivangi Surati

Analysis of Algorithm: Control Structure

- Execution time of an algorithm

n=10

if (n < 0)

        absval = -n

else

        absval=n

- Analysis:

Statement	Cost	Frequency
n=10	C1	1
if (n < 0)	C2	1
absval = -n	C3	1
else		
absval=n	C4	1

Total cost= C1*1+ C2*1 + max (C3 *1, C4*1)

Time required is constant

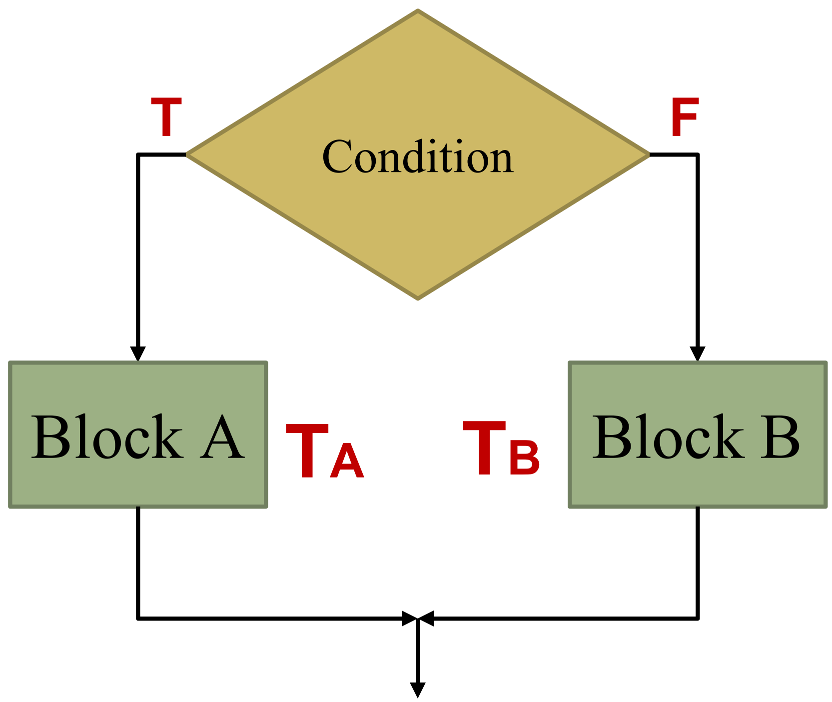
Algorithm and Analysis: Example-1

- Control Structure:

1. a=5
2. b=10
3. if (a < b)
4. {
5. max=a;
6. a=0;
7. }
8. else
9. {
10. max=b;
11. }

- Analysis:

1. Constant
2. Constant
3. Constant
4. -
5. Constant
6. Constant
7. -
8. -
9. -
10. Constant
11. -



Computation Time:
Max (T_A,T_B)

Analysis of Algorithm: While Loop

- While Loop (LX-1):

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

- Analysis:

Statement	Cost	Frequency
i=1;	C1	1
sum = 0;	C2	1
while (i <= n)	C3	n (true) +1 (false)
i = i + 1;	C4	n
sum = sum + i;	C5	1

Total cost= C1*1 + C2*1 + C3 * (n+1) + C4 * n + C5 * 1

Time required is proportional to n

Algorithm and Analysis: For Loop

- Loop (LX-2):

```
for (i=1 ; i<=n ; i++)
    a = i + 5;
```

Statement	Cost	Frequency	Remarks
i=1	C1	1	Constant
i <= n	C2	n (true) + 1 (false)	Depends on value of n
a = i + 5;	C3	n	Depends on value of n
i++	C4	n	Depends on value of n

$T= c1*1 + c2*(n+1) + c3 * n + c4 * n$

Time required is proportional to n

Algorithm and Analysis: Example-2

1. sum=0 //initialize sum with 0
2. for i in 1 to n
3. sum = sum + i
4. print sum

Statement	Cost	Frequency	Remarks
sum = 0	C1	1	Constant
for i in 1 to n	C2	n (true) + 1 (false)	Depends on value of n
sum = sum + i	C3	n	Depends on value of n
print sum	C4	1	Constant

Algorithm and Analysis: Example-3

Finding smallest no. in array X containing n elements.

```
Algorithm Smallest (X, n)
{
1.  Let min = X_1
2.  For i = 2 to n
3.  Do
4.      If X_i < min
5.      Then  min = X_i
6.  Done
7.  Print min
}
```

Statement	Frequency	Total
Algorithm Smallest (X, n)	-	0
{	-	0
Let min=X1	1	1
for i= 2 to n	n	n
if Xi < min	n-1	n-1
min = Xi	n-1	n-1
print min	1	1
}	-	0
	Total	**3n**

Loop Examples

LX-3: c is Constant

```
for (i=1 ; i<=c ; i++)
        a = i + 5;
```

Constant time

LX-4: n is Variable

```
for (i=n ; i>0 ; i--)
        a = i + 5;
```

LX-5:

```
for (i=1 ; i<=n ; i+=2)
        a = i + 5;
```

(n/2)

LX-6:

```
for (i=1 ; i<=n ; i*=2)
        a = i + 5;
```

log (n) base 2 = $\log_2(n)$

LX-7:

```
initialize a
initialize b
if (a < b)
        for (i=1 ; i<n ; i++)
                a = i + 5;
else
        for (i=m ; i>0 ; i--)
                b = i + 5;
print a
print b
```

Nested Loops

NLX-1:

```
for (i=1 ; i<=n ; i++)
        for (j=1 ; j<=n ; j++)
                a = i +j
        b = i + 5
print a, b
```

Frequency:

$n+1$

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

$n * n = n^2$

n

1

Find total

Nested Loops

NLX-2:

```
for (i=1 ; i<=m ; i++)
      for (j=1 ; j<=n ; j++)
            a = i +j
      b = i + 5
print a, b
```

Frequency:

m+1

m*(n+1) = mn+m

m * n

m

1

Find total

Prepared by: Dr. Shivangi Surati

NLX-3:

```
for (i=1 ; i<=n ; i++) {
        for (j=1 ; j<=i ; j++) {
                a = i + j
        }
        b = i + 5
}
print a, b
```

NLX-4:

```
for (i=1 ; i<=n ; i++) {
        for (j=1 ; j<=n-i ; j++) {
            a = i + j
        }
        b = i + 5
}
print a, b
```

NLX-5 (HW):

```
for (i=1 ; i<=n ; i++)
        for (j=i ; j<=n ; j++)
            a = i +j
        b = i + 5
print a, b
```

NLX-6 (HW):

```
Algorithm Add(a,b,c,m,n) {
    for i=1 to m do {
        count = count + 2
            for j=1 to n do
                count = count + 2
    }
count = count + 1
}
```

C-1 (HW):

```
for (i=1 ; i<=m ; i++)
        a = a + i
for (j=1 ; j<=n ; j++)
        b = b + j
print a,b
for (k=1 ; k<=p ; k++)
        c = c + k
print c
```

Challenges

CH-1:

```
for (i=1 ; i<=n ; i*=c)
        a = a + i
```

CH-2 (HW):

```
for (j=n ; j>0 ; j/=c)
        b = b + j
```

CH-3 (HW):

```
for (i=k ; i<=n ; i++)
        a = a + i
```

CH-4 (HW):

```
for (i=k ; i<=n ; i*=c)
        b = b + i
```

Bubble Sort Algorithm

```
Algorithm BubbleSort(A , n):
for i ←1 to n do
        noPass++
        for j ←1 to n-i do
                noIteration++
                if (A[ j ] > A[ j+1])
                        swap (A[ j ], A[ j+1]) // A[ j ] ↔ A[ j+1]
                        noExchange++
        end for
end for
```

Bubble Sort Algorithm-Trace

Trace of Bubble Sort Algorithm

Input: $a[9] = \{54, 26, 93, 17, 77, 31, 44, 55, 20\}$

For PASS-I only (i=1)

54	26	93	17	77	31	44	55	20	Exchange
26	54	93	17	77	31	44	55	20	No Exchange
26	54	93	17	77	31	44	55	20	Exchange
26	54	17	93	77	31	44	55	20	Exchange
26	54	17	77	93	31	44	55	20	Exchange
26	54	17	77	31	93	44	55	20	Exchange
26	54	17	77	31	44	93	55	20	Exchange
26	54	17	77	31	44	55	93	20	Exchange
26	54	17	77	31	44	55	20	93	93 in its right position

Bubble Sort Algorithm-Analysis

Complexity in Detail

Bubble Sort compares the adjacent elements.

Cycle	Number of Comparisons
1st	(n-1)
2nd	(n-2)
3rd	(n-3)
....	......
last	1

Hence, the number of comparisons is

```
(n-1) + (n-2) + (n-3) +.....+ 1 = n(n-1)/2
```

Bubble Sort Algorithm-Optimized

```
Algorithm BubbleSort(A , n):
for i ←1 to n do
          flag ← 1
          for j ←1 to n-i do
                    if (A[ j ] > A[ j+1])
                              swap (A[ j ], A[ j+1])
                              flag ← 0
          end for
          if (flag)   //if (flag == 1)
                    break
end for
```

Insertion Sort Algorithm

```
Algorithm InsertionSort(A , n):
for j ←2 to n     //A.length
        key=A[ j ]
        i=j-1
        while (A[ i ] > key and i > 0)
                A[ i+1 ] ← A[ i ]) //assignment
                i ← i - 1
        A[ i + 1 ] ← key
```

Algorithm- Find sum of Digits

```
Algorithm sumDigits (no):
sum ← 0
while no > 0
        d ← no % 10
        sum ← sum + d
        no ← no / 10
end while
```

Analysis?

Put counter in while loop

HW

- Analyze the following programs:

(1) Find factorial of n (iterative approach)

(2) Linear search

(3) Selection Sort

(4) Matrix addition (Both matrix of n*n)

(5) Matrix addition (Both matrix of m*n)

(6) Matrix multiplication (Both matrix of n*n)

(7) Matrix multiplication (Matrix-1 of m*n and Matrix-2 of n*p)