18CSC204J - DAA

2017 MARCH 28 Algorithmic Design THURSDAY 30

Week 13 89-276

9 → Introduction - Algorithm Design
→ Fundamentals of Algorithm
10 → Correctness of Algorithm
→ Time complexity analysis
11 → Insertion sort line count, Operation count.
12 → Algorithm Design paradigms

1 → Designing an algorithm
→ Best / Worst / Average case
2 → Asymptotic notations based on growth functions
3 → O Θ ω Ω
→ Mathematical analysis
4 → Induction, recurrence relations

5 → Solution of recurrence relation
→ Substitution method.
6 → Solution of recurrence relation
→ Recursion tree
→ Solution of recursion relation

Notes

→ Examples.

I never follow the clock: hours were made for man, not man for hours.

Computational thinking:

The knowledge of using computers to perform our day to day activities

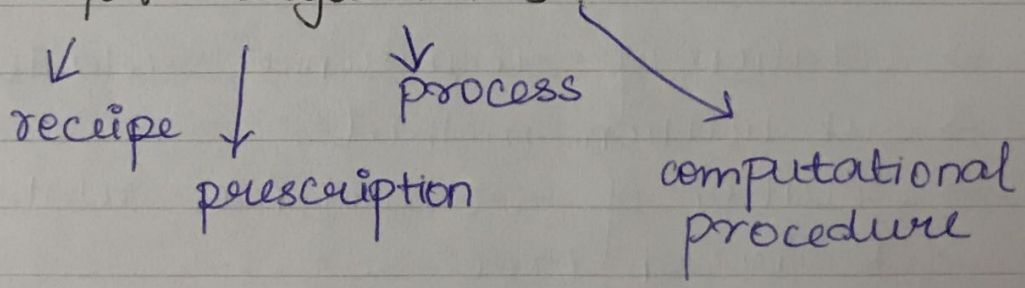
Algorithmic thinking:

analytical skill that is required for writing effective programs in order to solve given problems

Algorithmics:

The art of designing, implementing and analyzing algorithms.

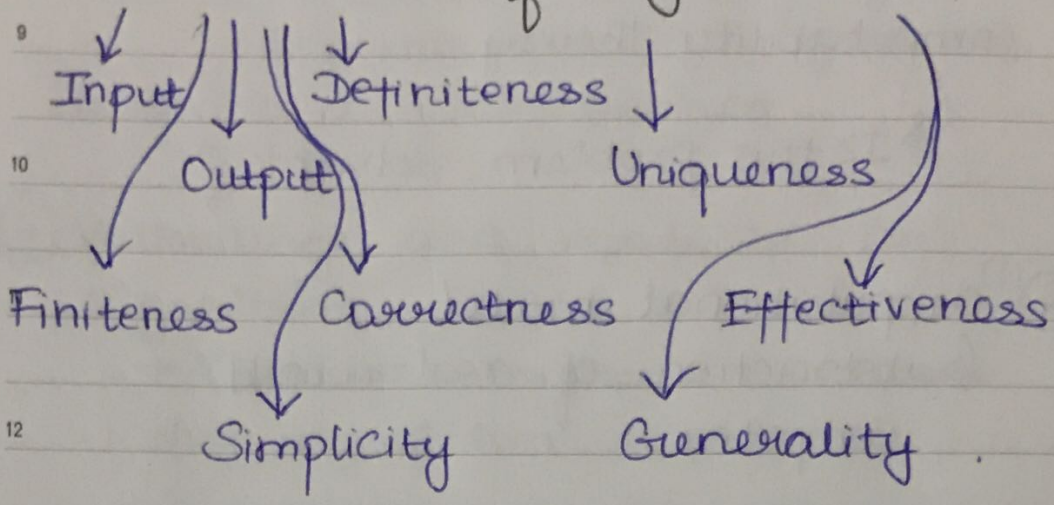
Other terminologies for algorithms



Algorithmic thinking

Notes Knowledge on how to solve a problem.

Characteristics of Algorithms



Need for Algorithm Efficiency

Eg: TSP Travelling Sales Person.

n = no. of cities

(n-1)! = No of routes.

Fundamental stages of problem solving.

1) Problem
2) Planning
3) Algorithmic design
4) Algorithm correctness
5) Algorithm analysis
6) Implementation & empirical analysis
7) Post Analysis
8) OK? → NO Go to Before Steps
9) Exit.

Positive attitude is better than negative thinking.

SUNDAY 2

Take small step every day and one day you will get there.

① Study of algorithm starts with computability theory
↳ Is this problem solvable?

② (i) computational model
↳ abstraction of real-world computer.

* facilitates machine independent analysis.

(ii) data organization

algorithm + data structure = program

③ (i) How to design an algo?
(ii) How to express an algo?

Algorithmic Design:
Developing algorithmic solutions using an appropriate design strategy.

Notes

Algorist
Skilled algorithm designer.

Algorithm specification.

Communication of algorithm design to a programmer.

(4) Validating and verifying an algorithm.

→ These are methods to check for algorithm correctness.

Process of checking whether an an algorithm gives correct outputs for valid inputs - Algorithm validation.

mathematical proof that an algorithm works correctly - Algorithm verification (or) Algorithm proving.

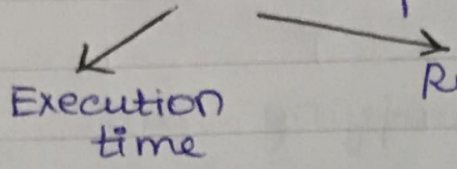
(5) complexity analysis results in optimal algorithms with few resource usage.

Notes

Measures

↳ Subjective : ease of implementation, algorithm style, readability. (Factors are not quantified)

95-270 ↳ Objective.
(they are quantified)
Time & Space.



(6) Analysis carried out after the program is executed is empirical analysis (or) priori analysis (or) theoritical analysis.

Benchmark dataset provides vital statistical information about algorithm behaviour - empirical analysis (or) posteriori analysis.

Profiling: measuring time/space requirement of the program.

(7) Algorithmic gap: difference b/w upper and lower bounds. Technically it should be zero.

Notes

Week 96-269

Classification of algorithms

① based on implementation

② based on design

③ based on area of specialization

④ based on tractability.

① Problem reduction - scientific principle for problem solving.

Recursion: The problem reduction process is continued till the given problem is reduced to a smaller problem that can be solved directly. Results of sub-problems combined gives the result.

Recursive function
- $n! \rightarrow (n-1)!\ n \geq 1$ → recursive (inductive) case
- $n! \rightarrow 1\ \ n = 0$ → base case

↳ works by principle of work postponement or delaying the work.

Notes

Non-recursive algo: (iterative)
- → deductive in nature
- → rely on loop constructs.

→ Recursive
→ Non-recursive.

① → Sequential
→ parallel
→ distributed.

deterministic
↓
non-deterministic

→ exact
→ approximation

② → brute force
→ divide and conquer
→ dynamic programming.
→ greedy approach.
→ branch & bound algo.

③ → General Eg: Sorting.
→ Domain Specific → Convex hull prob

String → Graph colouring
Graph → Graph colouring
TSP → Geometric algo
Chinese postman → closest pair prob

④ → Easily solvable.
→ Unsolvable.
→ Intractable.

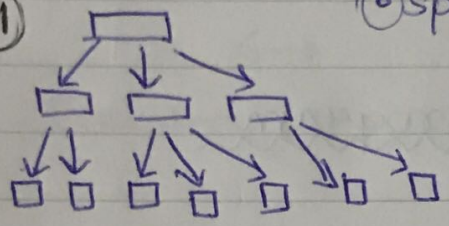
Notes

Design approaches

① Stepwise Refinement (or) Top-down design.

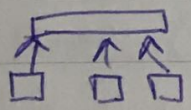
② Bottom up Approach. (or) synthesis / composional approach.

③ Structured Programming

① 

- splits problems to sub problems.
- Design supports functional, procedural & OOP.
- Removes irrelevant info.
- Redundant overlapping works.
- Dependencies on testing.

② → Reusability.



① + ② → mixed design.

③ → rigorous discipline
→ Begin & End ✓
→ Go to ✗
→ well designed, readable, easy to understand.

- Sequence
- Selection
- Iterative.

Weakness of attitude becomes weakness of character.

SUNDAY 9

Morale is a state of mind. It is and courage and hope.

100-265

Algorithm writing guidelines

9

%% comments.

10

integer
11 float-
char
12 boo

} all supported explicitly not mentioned.

1 block structured program

2 Begin
S_1
S_2
3 ⋮

End.
4

5 a = 20 + r Assignment.

6 Input x, y. Statements
Print x, y.

Notes if (c) then
⟷
else
⟷
End if

Conditional statement

```
case (c)
9  Begin
       C1 : <->
10     C2 : <->
   End.
11 End case.
```

Case Statement

```
12 for i = 0 to 10 [2] do
          <->
1  End for.
```

For Statement.

```
2  while (c) do
          <->
3  End while
```

While Statement

```
4  repeat
          <->
5  until (c)
```

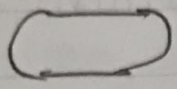
Repeat Statement

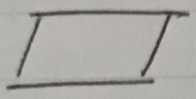
```
6  break,
   return   } allowed.
```

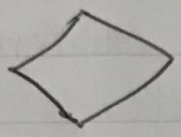
Notes

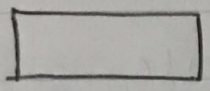
Flowchart.

represents algorithms graphi-cally.

 → Start / End

 → I/O

 → Decision.

 → Subroutine / process

 ⟷ Connector.

Algorithm Correctness. (Pg 53)

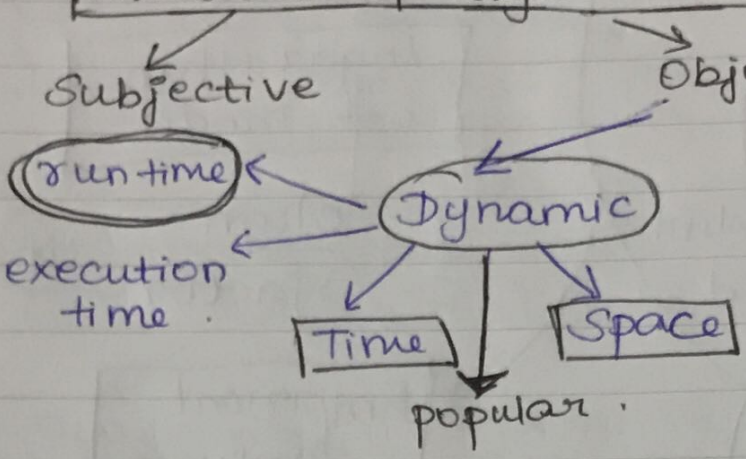
Testing → Proof.

- Assertion
- Precondition
- Post condition.

Notes **Binary / Exponential Recursion.**

$$A(m,n) = \begin{cases} n+1 & \text{if } m=0 \\ A(m-1,1) & \text{if } n=0 \\ A(m-1, A(m,n-1)) & \text{otherwise.} \end{cases}$$

Measure of algorithm analysis



Runtime is expressed in terms of CPU cycles.

RAM - Random Access Machine.

has an Input device,
an Output device,
a control unit
data storage,
accumulator.

RAM is a computational model that approximates real world sequential machine.

Notes Any algorithm can be simulated using RAM.

RAM

- Unit cost model
  - Not dependent on operand.
- Logarithm cost model.
  - clogn
  - factor.
- Mathematical Analysis
  - (or) priori
  - (or) theoritical
  - } analysis.
- Empirical Analysis

- independent of machine
- Applied before algo is converted to prg.

Analysis of Iterative Algo

→ Measure Input size

→ Measure Run time
- operation count
- step count

→ Best, Worst, Average case } efficiencies

Notes → Identify rate of growth.

Scaling : Increasing input size to access the algorithm.

Measuring input size.

- Integers
- Matrices
- Arrays
- Graphs.

Measuring run time.
(Algorithm analysis)

- Step counts
- Operations count
- Asymptotic analysis
- Recurrence relations.
- Amortized analysis.

Step count

count the number of steps / instructions.

- Declarative = 0
- Initialization = 1
- Comment
- bracket
- begin / end.
- if / while

} = 0

- Expression = 1
- Assignment
- func. call.
- return
- break
- continue
- go / for.

} = 1

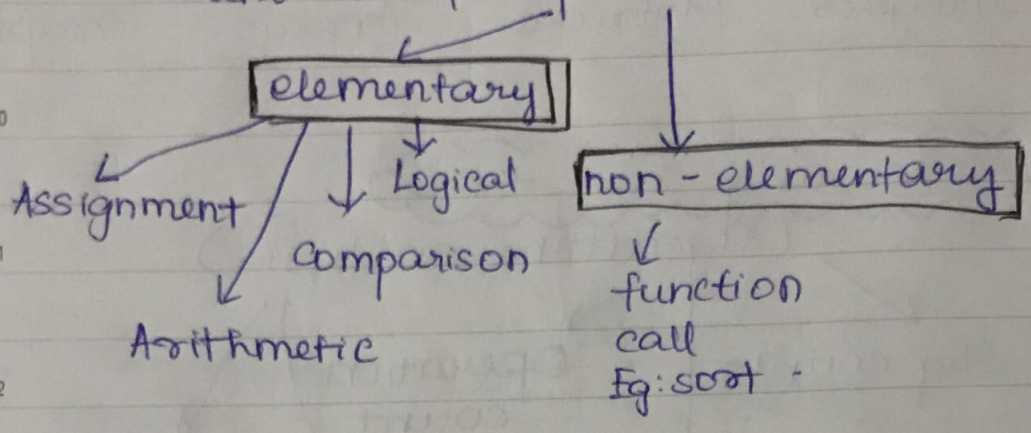
→ easy

→ not dependent on operands.
- a = a + 10
- a = a + 10^10

107-258

Operation Count

count no. of operations



→ Count basic operation & express formula.
→ Simplify formula.
→ Represent as a function.

Sequence

Begin:
S1 → m operation
S2 → n operation
End

m+n — addition principle.

Selection.

Notes

if (c) then
P → m operation

else
Q → n operation

max (m, n)

Repitition.

for i=1 to n → n times
    p. → m operation.
End for

m * n multiplication principle.

Reasons for Algorithms Analysis

1) Exponential - ~~easy~~ difficult to solve.

2) Polynomial
   - Constants vary from machine to machine
   - Addition & composition (+ *).

Time Complexity

T(n) or t(n) is → operation count (or) → step count.

w(n) → Worst case (upper bound).

Notes pessimistic analysis / worst scnario

B(n) → Best case (lower bound.

ideal case.

109-256

$A(n)$ → average performance.

Rate of Growth

→ measuring larger inputs.

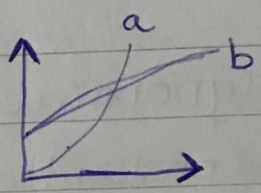
Eg $T_A = n^2$.

$$\frac{T(1000)}{T(10)} = \frac{10^3 \times 10^3}{10 \times 10} = 10^4$$

→ Comparison framework.

$t_A = n^2$ ✗

$t_b = 40n$ ✓.



Asymptotic Analysis

→ closed form

→ exact formula is not possible.

approx eg

Sterling formula. $n! = \sqrt{2n\pi}\, n^n e^{-n}$

Harmonic series $H_n = \sum_{i=1}^{n} \frac{1}{k} = \log(n) + \gamma + O\left(\frac{1}{n}\right)$.

Notes

Name	Meaning.	Limits	Rep.	Mathematic ≈
$O()$	$t(n) < g(n)$	$\lim_{n\to\infty} \frac{tn}{g(n)} = c$ $c>=0$.	$t(n) = O(g(n))$	$\leq$
$\Omega()$	$tn) \geq g(n)$	$\lim_{n\to\infty} \frac{t(n)}{g(n)} \neq 0$	$t(n) = \Omega(g(n)$	$\geq$
$\Theta()$	$th) \approx g(n)$	$\lim_{n\to\infty} \frac{t(n)}{g(n)} = c$ $c>0$.	$t(n) = \Theta(g(n))$	$\approx$
$o()$	$t(n) << g(n)$	$\lim_{n\to\infty} \frac{t(n)}{g(n)} = 0$	$t(n) = o(g(n))$	$<$
$\omega()$	$t(n) >> g(n)$	$\lim_{n\to\infty} \frac{t(n)}{g(n)} = \infty$	$t(n) = \omega(t(n))$	$>$
$\sim()$	$t(n) = g(n)$	$\lim_{n\to\infty} t(n)/g(n) = 1$	$t(n) \sim g(n)$	$=$

Notes

Happiness depends upon ourselves.

111-254

Rules

$t(n) = O(g(n))$
$t(n) = \Omega(g(n))$
$t(n) = \Theta(g(n))$
} Reflexivity Rule.

$t(n) = O(g(n))$
$g(n) = O(h(n))$
$t(n) = O(h(n))$
} $t(n)$ $h(n)$ $g(n)$ are complexity func of 2 diff algo.

$O(O(t(n))) \rightarrow$ Law of composition.

$t(n) + g(n) = X(\max(t(n), g(n)))$

$X = O, \Theta, \Omega$

Space Complexity

↙ variable ↘ fixed.

Note: **Empirical** (experimental)

Benchmark dataset → Test Algo → Result.

Statistical Test → Power test.
↙ Ratio
↘ Linear Regression Test.

Recursive Algorithms

Recurrence relation
↙ Linear
↘ Non-Linear.

Linear:
- Based on order
  - t_n → 1st
  - t_{n-1} → 2nd
  - t_{n-2} → 3rd
  - → order.
- Based on coeff
  - → const coeff
  - → variable coeff.
- Based on homogeneity.
  - → Homogenous $f(n) = 0$
  - → Non-homogenous

$f(n) = a_0 t_n + a_1 t_{n-1} + \ldots a_k t_{n-k}$.

Linear Eg:

Factorial	Binary search
TOH	Merge sort
Hanshake Prob	Convex hull
Fibonnacci	Random select.
Permutation.	

Eg: 1000, 2000, 4000, 8000, ...

$t_n = 2 \times t_{n-1}$ → recurrence

$t_n = 2^n \times t_0$ → actual solution.

Techniques to solve

(1) Guess & verify.
(solution by mathematical induction).

Guess the actual solution from recurrence & verify.

(2) Substitution. Plug & Chug.
(iteration).

Plug → substitute.
Chug → simplify.

Notes backward substitution

→ backtracking / backward iteration method.

Forward substitution.

↳ forward iteration.

(3) Recurrence - tree.

→ helps to obtain asymptotic bounds.

- Formulate equation.
- Collect info → Level
  - ↳ Cost per level.
  - ↳ Total cost.
- Express complexity.
- Verify by other method.

Other methods.

- Difference method
- Polynomial reduction method.
- Generating function method.
- Table look-up method / master theorem.

Notes