classmate

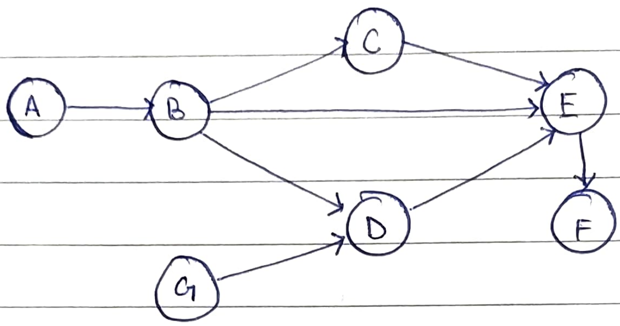
Date 20.1.2020

Page 1

BASICS OF GRAPH AND COMPLEXITY THEORY

~~GRAPH~~ a graph is an ordered pair $G = (V, E)$ comprising:

- V: a set of vertices (also called nodes or points)
- $E \subseteq \{(x, y) \mid (x, y) \in V^2\}$: a set of edges.

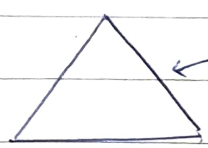


DAG

GRAPH CHARACTERISTICS

- directedness of edges
- degree of vertex
  - directed : indegree, out degree
  - undirected : degree
- edge weights
- connectedness

tree => no cycles
connected



connected component

total degree
= 2 × no. of edges.

degree 2 2 2

GRAPH REPRESENTATIONS

- adjacency matrix →

	A	B
A	e_1	e_2
B	e_3	e_4

- adjacency list → A → (B)   B → (C, D, E)
- incidence matrix →

vertices	(A, B)	(B, C) edges
A		
B	1	
	1	

directed:
-1 source
1 target

TYPES OF GRAPHS

- ⇒ simple graphs
- weighted graphs
- directed graphs
- ⇒ connected graphs
- trees : no cycles.
- ⇒ bipartite graphs : no odd cycles
- complete graphs : fully connected.

GRAPH ALGORITHMS

- minimum vertex cover

min. set of vertices that cover all the edges.

- maximum matching

max. set of disjoint edges

- shortest part

finding shortest path between 2 vertices.

- minimum spanning tree

min. set of edges which span the tree.

- graph flows

max. flow through the graph, given start, end node

COMPUTATIONAL COMPLEXITY

complexity is a measure which evaluates the order of the count of operations, performed by a given algorithm as a function of the size of input data. to put this simpler, complexity is a rough approximation of the no. of steps necessary to execute an algorithm.

- asymptotic complexity
- complexity notation
- inherent complexity of a problem.

ASYMPTOTIC COMPLEXITY

- how to analyze running time and space of algorithm.
- complexity analysis: asymptotic, empirical, others

- different performance measures are of interest.
  - worst case (often easiest to analyze, need one 'bad' ex.
  - best case (often easy for same reason)
  - average case

COMPLEXITY NOTATION

let f, g be positive real valued functions defined on an unbounded subset of the real +ve nos.

- big-O (O):
  $f(x) \in O(g(x))$ (upper bound)
  iff there exists a +ve real no. x_0 st. (c)

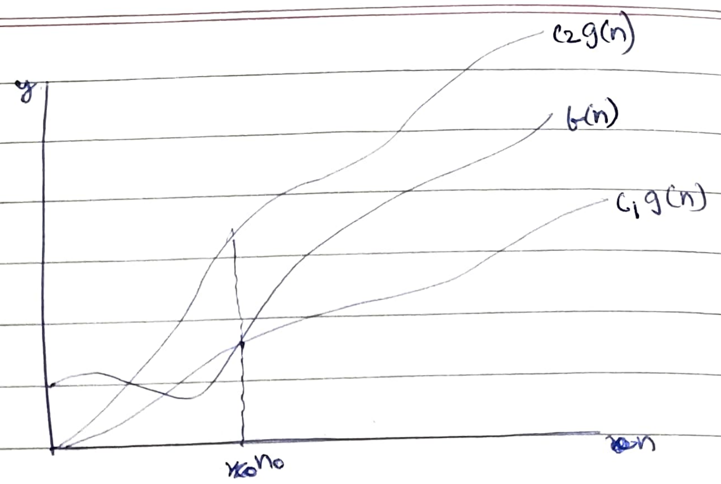
$$f(x) \leq c\,g(x) \quad \forall \; x \geq x_0$$

- omega (Ω): ~~f(x)~~
  $f(x) \in \Omega(g(x))$ (lower bound)
  iff there exists a +ve real no. c and a real no. x_0 st.

$$f(x) \geq c\,g(x) \quad \forall \; x \geq x_0$$

- theta (Θ): $f(x) \in \Theta(g(x))$ (composite bound)
  iff there exists +ve real nos. c_1, c_2 and real no. x_0 st.

$$c_1 g(x) \leq f(x) \leq c_2\, g(x) \quad \forall \; x \geq x_0$$



EXAMPLE: MERGE SORT

$$T(n) = 2T\left(\frac{n}{2}\right) + n$$

$$= 4T\left(\frac{n}{4}\right) + \frac{2n}{2} + n$$

$$= 2^k T\left(\frac{n}{2^k}\right) + kn$$

$$T(1) = 0$$

$$\frac{n}{2^k} = 1$$

$$\Rightarrow 2^k = n$$

$$\Rightarrow k = \log_2 n$$

$$\therefore T(n) = 2^k \times 0 + \log_2 n \times n$$

$$\therefore T(n) = n \log_2 n$$

INHERENT COMPLEXITY OF A PROBLEM

consider the case where we are trying to find the lower bound on the worst case of the sorting problem, where we are using comparisions to sort. any sorting algorithm requires Ω (N log N) comparisions.

- given an input of N distinct nos, choose permutations of N indices.
- algorithm independent proof using interactive approach.
- initially, possible no. of answers (permutations) equals N!.
- each comparision reduces size of possible answer set by at most 2 (in the worst case input).
- $\log(N!) \in \Omega(N \log N)$

$$\log(n!) \geq \frac{n}{2} \log \frac{n}{2}$$

$$\geq \frac{n}{2}(\log n - \log 2)$$

$$\geq \frac{1}{2}(n \log n - n \log 2)$$

$$\geq \frac{1}{2} c\, n \log n - c$$

$$\geq \frac{1}{2} n \log n$$