classmate
Date
Page

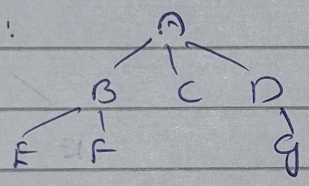
Homework

* Derive the time and space complexities of both BFS and DFS algorithms in term of the number of nodes N and edges E in a graph.

→ Breadth - first search:

BFS explores the graph level by level. Starting from a chosen node, it visits all immediate neighbors first, then moves outward to next layer of neighbors and so on

Eg illustration Tree:



BFS order: Starts with A → B, C, D → E, F, G

BFS uses a Queue (FIFO) → first node discovered is the first to be visited

* Time complexity:

Every node is visited once → O(N)

Every edge is explored once → O(E²).

Time = O(N+E²)

* Space complexity:

Queue may hold an entire level of nodes in the worst case

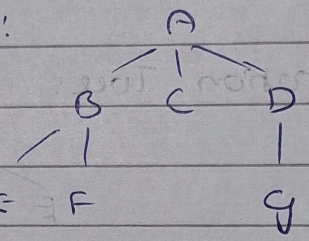
Space = O(N)

* Sparse vs Dense Graphs (BFS):
  - Sparse (few edges): BFS ≈ O(N)
  - Dense (many edges): close to N^2): BFS ≈ $O(N^2)$

→ Depth - first search (DFS):

It explores a graph branch by branch
start at the root (or chosen node)
go as deep as possible into one path
backtrack when no more nodes are left, then
then explores the next path

Example illustration of Tree:



DFS order: A → B → E → F → C → D → G
It divides deep first, then comes back

DFS uses stack (LIFO): This can be an explicit stack or the recursion call stack

* Time Complexity:
  - Every node is visited once → O(N)
  - Every edge is explored once → O(E)

  Time = O(N+E)

* Space Complexity

Stack depth can go as deep as the number of nodes (worst case: a long chain)

Space = O(N)

Date
Page

* Sparse vs Dense Graphs:
  - sparse: DFS $\approx O(N)$
  - dense: DFS $\approx O(N^2)$

Note:
- BFS is best for the shortest path in a unweighted graph
- DFS is best when you want to explore all possible paths or check connectivity.