Properties of Asymptotic Growth rates:

1. Big "oh" (O): The function $f(n) = O(g(n))$ (read as "f of n is big oh of g of n") iff (if and only if) there exist positive constants c and n_0 such that $f(n) \leq c * g(n)$ for all n, $n \geq n_0$. It also indicates the upper bound on the worst-case complexity.

2. Omega (Ω): The function $f(n) = \Omega(g(n))$ (read as "f of n is omega of g of n") iff there exist positive constants c and n_0 such that $f(n) \geq c * g(n)$ for all n, $n \geq n_0$.

3. Theta (Θ): The function $f(n) = \Theta(g(n))$ iff there exists positive constants c_1, c_2 and n_0 such that $c_1 g(n) \leq f(n) \leq c_2 g(n)$ for all n, $n \geq n_0$.

Comparing Sorting Algorithms

Sorting Algorithm	Proeedure	Time complexity (Best case / Average case / Worst case)	Memory Requirement	Best for large Data?
Insertion sort	place each elements inits correct position	O(n) O(n²) O(n²)	O(1)	NO. Because of its time complexity
Merge sort	Divide and conquer	O(nlogn) O(n logn) O(nlogn)	O(n)	Yes, Because of its time complexity
Heap sort	Builds a heaps	O(nlogn) O(nlogn) O(nlogn)	O(1)	Yes, Because of its time complexity
Quick sort	Partitoning based, recursive sort	O(nlogn) O(nlogn) O(n²)	O(logn)	Yes, Because of its time complexity
Bubble sort	Repeated swaps until sorted	O(n) O(n²) O(n²)	O(1)	NO. Because of its high time complexity

Cycle Detection in an Undirected graph

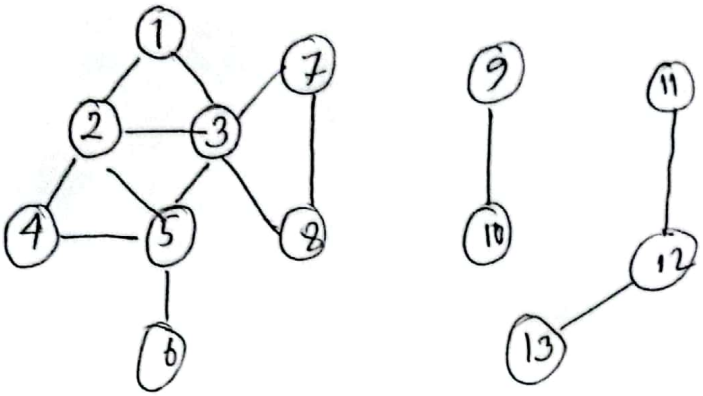
To detect cycles in an undirected graph, we can use DFS with parent tracking:

Algorithms:

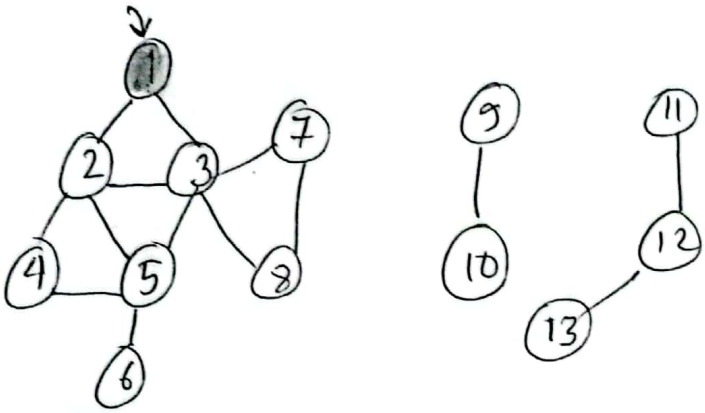
1. Start DFS traversal from any node.
2. Mark each visited node.
3. If a visited node is encountered again (excluding the immediate parent), a cycle exists.
4. Stop when the first cycle is found.

Time complexity: $O(m+n)$ (where n = nodes, m = edges)

i) Breadth first search

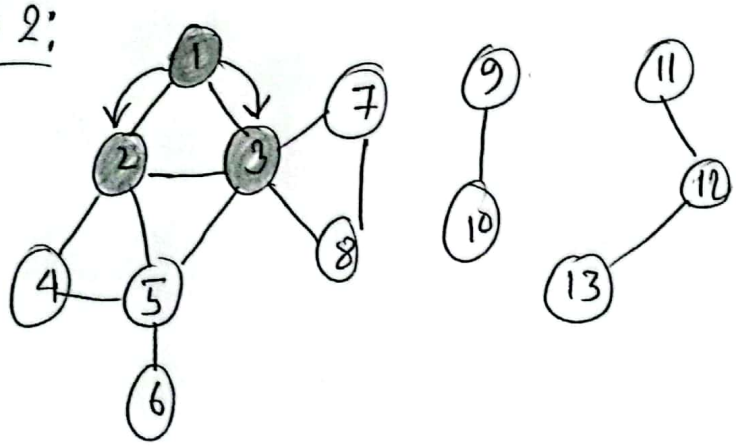


Step 1:



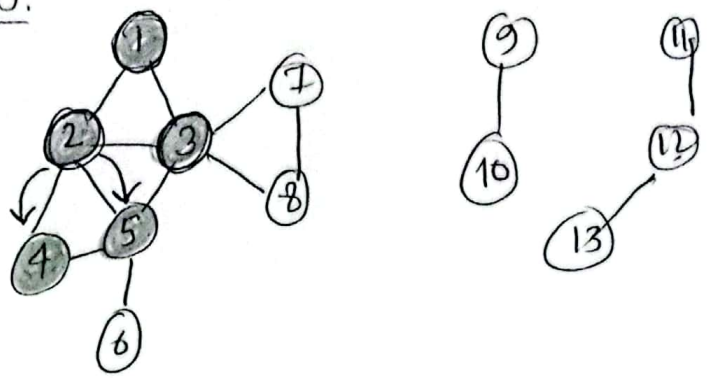
visited nodes: (1) →

Step 2:



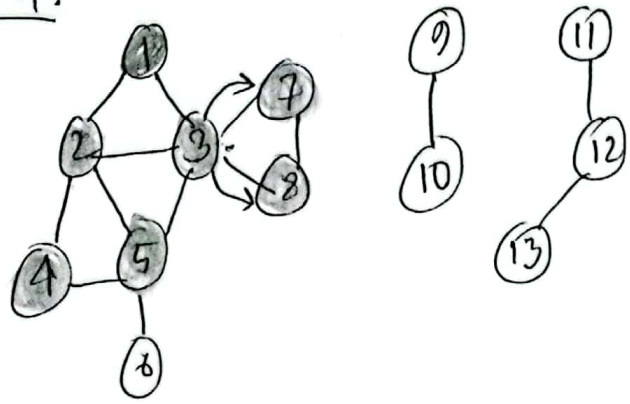
visited nodes: 1 → 2 → 3

Step 3:



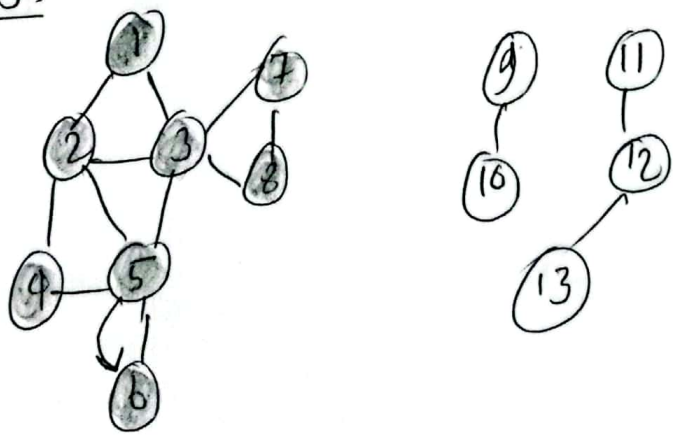
Visited nodes: 1 → 2 → 3 → 4 → 5

Step 4:



Visited nodes: 1 → 2 → 3 → 4 → 5 → 7 → 8

Step 5:

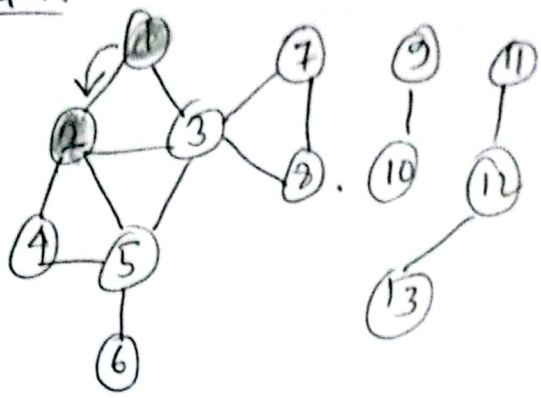


visited nodes: 1 → 2 → 3 → 4 → 5 → 7 → 8 → 6

nodes 9, 10, 11, 12, 13 are not possible to visit from node 1. Because there is no path from 1 to 9, 10, 11, 12, 13

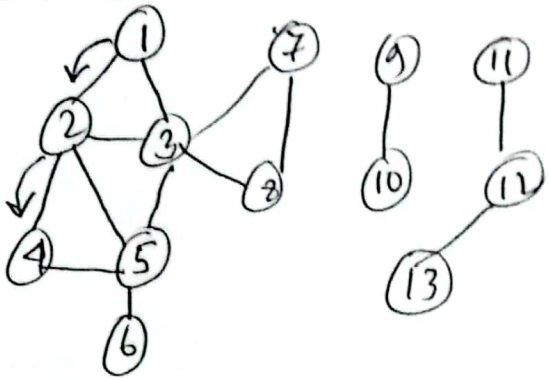
(11) Depth first search

Step 1:



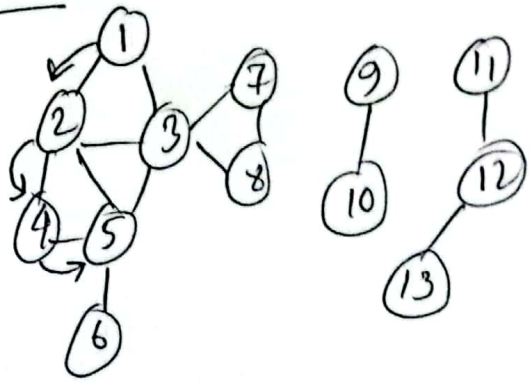
visited nodes : 1 → 2

Step 2:



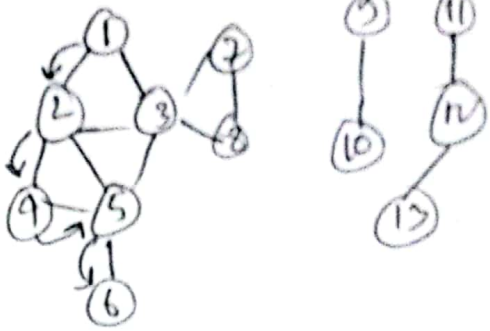
visited nodes: 1 → 2 → 4

Step 3:



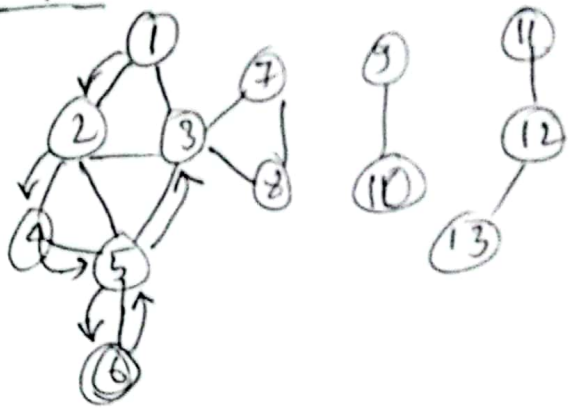
visited: 1 → 2 → 4 → 5

Step 4:



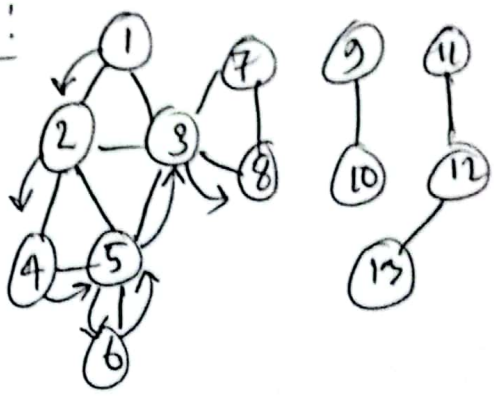
visited: 1 → 2 → 4 → 5 → 6

Step 5:



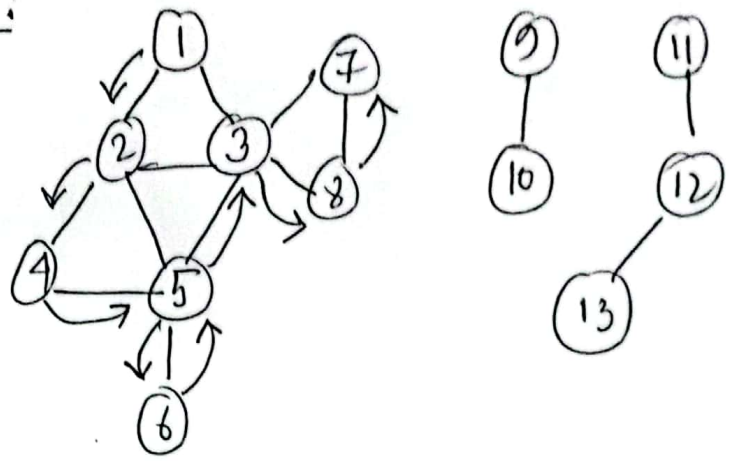
visited: 1 → 2 → 4 → 5 → 6 → 3

Step 6:



visited: 1 → 2 → 4 → 5 → 6 → 3 → 8

step 7:



visited : 1 → 2 → 4 → 5 → 6 → 3 → 8 → 7

nodes 9, 10, 11, 12, 13 are not possible to visited from node 4. Because there is not path from 4 to 9, 10, 11, 12, 13.