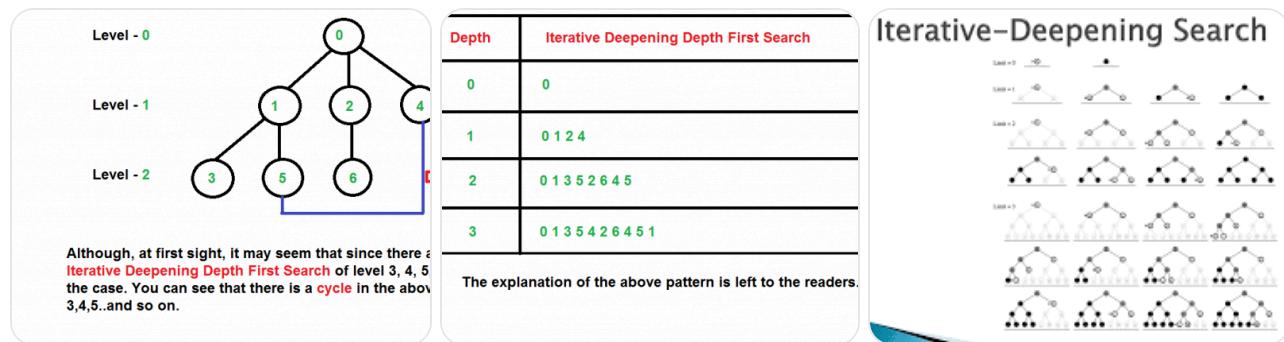


Iterative Deepening Depth First Search. what? why? how?
give me a complete concept with example and contrast
concepts with bfs and dfs. teach me effectively

I'll teach this **as a professor would**, assuming you know **basic DFS & BFS**, but want **deep clarity** for exams, interviews, and implementation.

We'll go **simple → medium → advanced**, with **examples, contrasts, and summaries**.

Iterative Deepening Depth-First Search (IDDFS)



1 WHAT is IDDFS?

Iterative Deepening Depth-First Search (IDDFS) is a **search strategy** that:

- Combines
 - DFS's **low memory usage**
 - BFS's ability to find the **shortest (shallowest) solution**

👉 It performs **Depth-Limited DFS repeatedly**, increasing the depth limit each time.

One-line definition (exam ready):

IDDFS performs DFS with increasing depth limits until the goal is found.

2 WHY do we need IDDFS?

Problem with BFS

- ✓ Finds shortest path
- ✗ Uses **huge memory** ($O(b^d)$)

Problem with DFS

- ✓ Uses very little memory
- ✗ May go **deep into wrong paths**
- ✗ **Not optimal** (may miss shallow solution)

IDDFS solves both

- ✓ Finds **shallowest solution**
 - ✓ Uses **low memory**
 - ✓ Guaranteed to terminate (if solution exists)
-

3 HOW does IDDFS work?

Core Idea

1. Run **DFS with depth limit = 0**
2. Run **DFS with depth limit = 1**
3. Run **DFS with depth limit = 2**
4. Continue until goal is found

Each iteration:

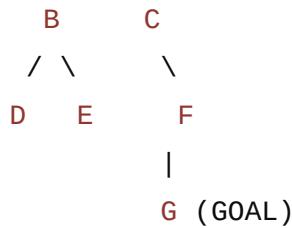
- Starts **from the root again**
 - Explores only up to the current depth limit
-

4 Step-by-Step Example

Search Tree

less





Copy code —

Iteration 0 (Depth limit = 0)

CSS

A

Copy code

✗ Goal not found

Iteration 1 (Depth limit = 1)

CSS

A → B

A → C

Copy code

✗ Goal not found

Iteration 2 (Depth limit = 2)

CSS

A → B → D

A → B → E

A → C → F

Copy code

✗ Goal not found

Iteration 3 (Depth limit = 3)

A → C → F → G

Copy code

✓ GOAL FOUND

- ✓ Found at **minimum depth**
 - ✓ Used DFS-like memory
-

5 Why repeating work is NOT a problem

You might think:

“But IDDFS revisits nodes again and again!”

True — but:

- Upper levels are **very small**
- Most nodes exist at **deepest level**

Cost intuition

pgsql

Total cost ≈ BFS cost

Extra work ≈ small

Copy code

So time complexity remains acceptable.

6 Pseudocode (Conceptual)

text

```
for depth = 0 to ∞:
    if DepthLimitedDFS(root, depth) == SUCCESS:
        return SUCCESS
```

Copy code

text

```

DepthLimitedDFS(node, limit):
    if node is GOAL:
        return SUCCESS
    if limit == 0:
        return FAILURE
    for each child:
        if DepthLimitedDFS(child, limit-1):
            return SUCCESS

```

 Copy code

7 Time & Space Complexity

Let:

- b = branching factor
- d = depth of shallowest solution

Metric	IDDFS
Time	$O(b^d)$
Space	$O(b \cdot d)$

👉 Same time as BFS

👉 Space almost like DFS

8 Contrast: BFS vs DFS vs IDDFS

Comparison Table

Feature	BFS	DFS	IDDFS
Strategy	Level-wise	Deep-first	DFS + increasing depth
Memory	✗ Very high	✓ Very low	✓ Low
Completeness	✓ Yes	✗ No	✓ Yes
Optimal (shortest path)	✓ Yes	✗ No	✓ Yes

Feature	BFS	DFS	IDDFS
Time Complexity	$O(b^d)$	$O(b^m)$	$O(b^d)$
Space Complexity	$O(b^d)$	$O(b \cdot m)$	$O(b \cdot d)$

(m = max depth of tree)

9 When should you use IDDFS?

- ✓ When **solution depth is unknown**
- ✓ When **memory is limited**
- ✓ When **shortest path is required**
- ✓ In **AI search problems, game trees, state-space search**

Real-world usage

- AI planning
 - Puzzle solving (8-puzzle, 15-puzzle)
 - Search in large state spaces
-

10 One-Page Exam Summary

Iterative Deepening DFS repeatedly applies Depth-Limited DFS with increasing depth limits until the goal is found.

It combines **DFS's low space** and **BFS's optimality**, making it **complete and optimal** with reasonable time complexity.

Want more?

If you want next:

- ✓ Dry-run with **stack snapshots**
- ✓ **Code in C / Java / Python**
- ✓ **Exam-oriented answers**
- ✓ **GATE / AI exam tricks**

Just tell me what you want next 

     ... Compare with Claude ▾

i got it how it uses dfs, but i didnt get how iddfs uses bfs concept.

Good question — this confusion is **very common**, and you're thinking in the **right direction** 

Let me clear it **cleanly and precisely**, without hand-waving.

First: what you already understand (good

You understand that **IDDFS internally uses DFS**.

✓ True

IDDFS runs DFS multiple times with different depth limits.

So the confusion is:

Where exactly is BFS used?

Important answer:

IDDFS does NOT use BFS algorithmically.

It uses BFS behavior and guarantees.

Now let's unpack this step by step.

1 What is the *core idea* of BFS?

Forget implementation for a moment.

BFS guarantees ONE key thing:

Nodes are explored in increasing depth order (level order).

That's it.

Example:

mathematica

```
Depth 0 → root
Depth 1 → all children
Depth 2 → all grandchildren
Depth 3 → ...
```

This property gives BFS:

Copy code

- ✓ Completeness
 - ✓ Shortest-path guarantee
-

2 What DFS does (contrast)

DFS behavior:

nginx

```
root → child → grandchild → great-grandchild → ...
```

Copy code

- Goes **deep first**
 - ✗ No depth order
 - ✗ May miss shallow solution
-

3 Now the key insight (THIS is the answer)

IDDFS mimics BFS globally, while using DFS locally

Let's write what IDDFS actually explores:

Iteration-wise exploration

Iteration	Depth limit	Nodes explored
1	0	All nodes at depth 0

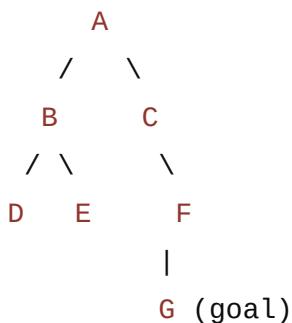
Iteration	Depth limit	Nodes explored
2	1	All nodes at depth ≤ 1
3	2	All nodes at depth ≤ 2
4	3	All nodes at depth ≤ 3

👉 This is EXACTLY BFS's level order guarantee

4 Visual proof (important)

Tree:

less



BFS order:

Copy code

CSS

```

A
B, C
D, E, F
G
  
```

Copy code

IDDFS exploration (simplified)

Depth = 0

CSS

A
 Copy code
Depth = 1

CSS

A → B**A → C**
 Copy code
Depth = 2

CSS

A → B → D**A → B → E****A → C → F**
 Copy code
Depth = 3

CSS

A → C → F → G
 Copy code
🔑 Notice:

- All nodes at depth 2 are fully explored **before** any node at depth 3
 - Exactly the BFS property
-

5 Why this gives BFS's optimality**BFS optimality rule:**

First time you see the goal → shortest path

IDDFS satisfies the same rule because:

- It **never explores deeper nodes** before exhausting shallower depths
 - First discovery of goal is at **minimum depth**
- ✓ Hence: **IDDFS is optimal like BFS**

6 Important clarification (exam gold)

✗ IDDFS does NOT:

- Use a queue
- Store frontier nodes
- Perform level-order traversal explicitly

✓ IDDFS DOES:

- Enforce **depth order across iterations**
 - Guarantee **level-by-level discovery**
 - Achieve BFS behavior using DFS runs
-

7 One-line exam answer (very important)

IDDFS uses DFS for traversal but achieves BFS behavior by exploring nodes in increasing depth limits, thereby ensuring level-order expansion and shortest-path optimality.

8 Analogy (helps lock it in)

BFS = elevator

- Visits floor 0, then 1, then 2, then 3

DFS = staircase

- Goes straight up without checking other floors

IDDFS = staircase with security checks

- You can only go up **one more floor per round**
 - Ensures every floor is fully checked before moving higher
-

9 Ultra-short summary

- IDDFS uses DFS code

- IDDFS behaves like BFS
 - BFS property = *depth order*
 - IDDFS enforces depth order via **iterative depth limits**
-

If you want, next I can:

- Prove **time complexity equality with BFS**
- Show **stack vs queue comparison**
- Give a **1-page handwritten-notes version**
- Give **GATE-style MCQs**

Just tell me.

     ... Compare with Claude ▾