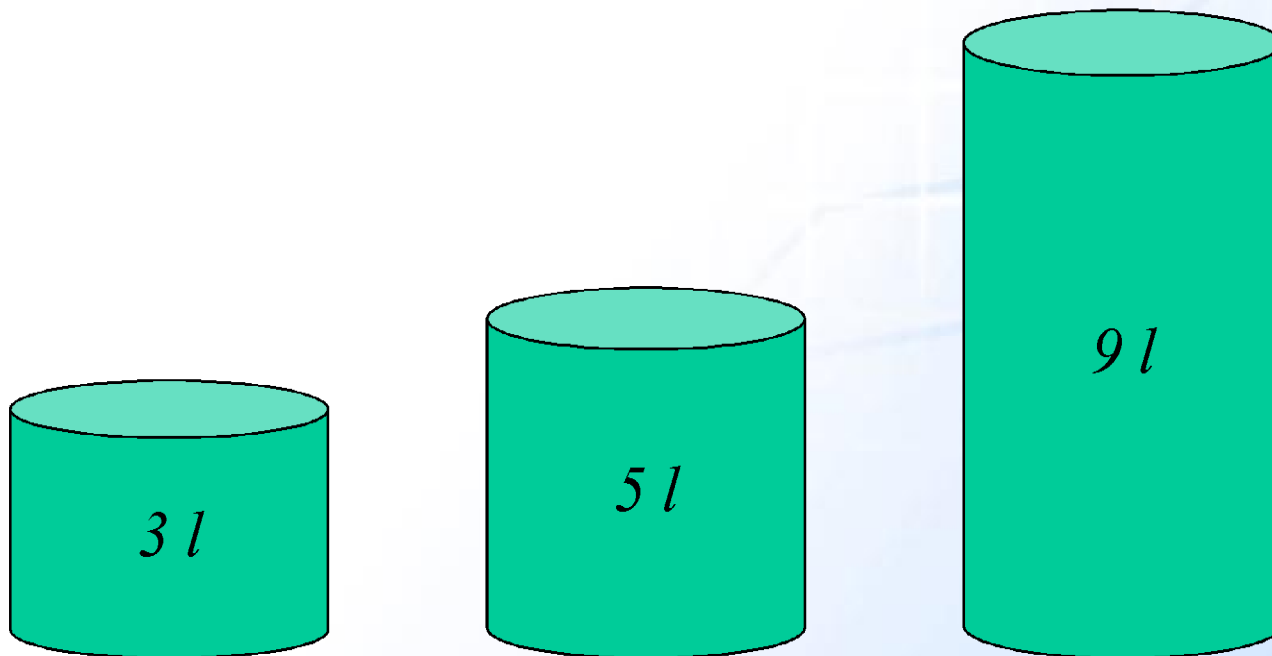


Pengantar Kecerdasan Buatan

AI sebagai pelacakan
(SEARCHING)



Search Example



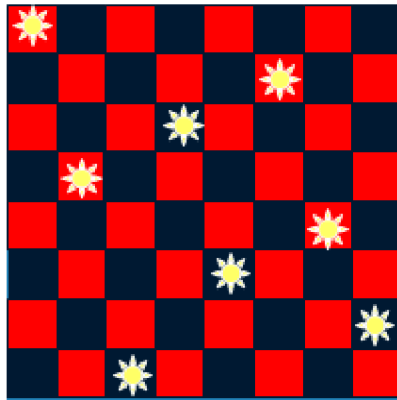
Using these 3 buckets,
measure 7 liters of water.

Search Example

Move cars forward and backward to “escape”

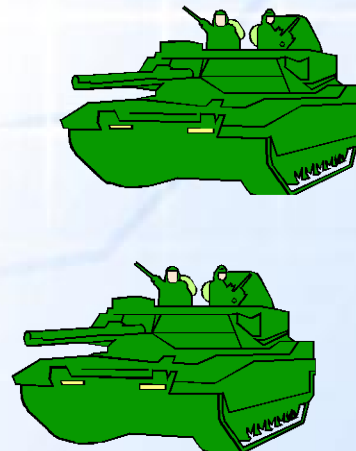
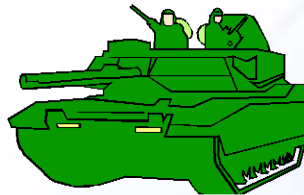


Search Example : 8 Queen Puzzle



Logistics

Very sophisticated. What goes where when?



Desert Storm logistics “paid for AI research”

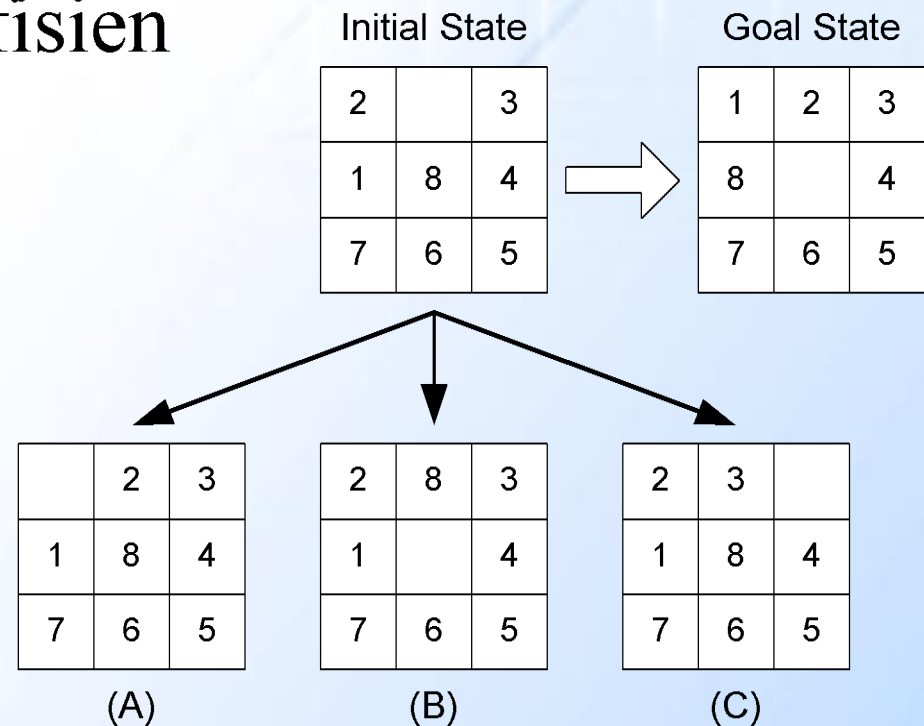
Job-Shop Scheduling

Industrial problem:

- Allocate machines and machinists to time slots
- Constraints on orders in which parts are serviced

Pelacakan

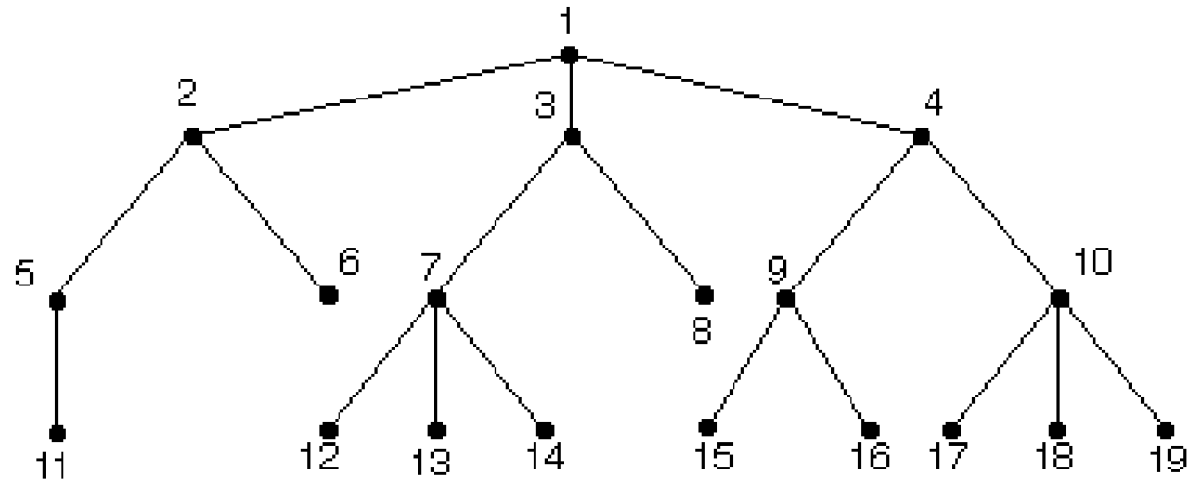
- Setiap masalah adalah “pohon virtual” dari seluruh solusi yang mungkin (berhasil atau tidak berhasil)
- Tujuannya menentukan/mencari strategi pelacakan yang efisien



Jenis-jenis Pelacakan

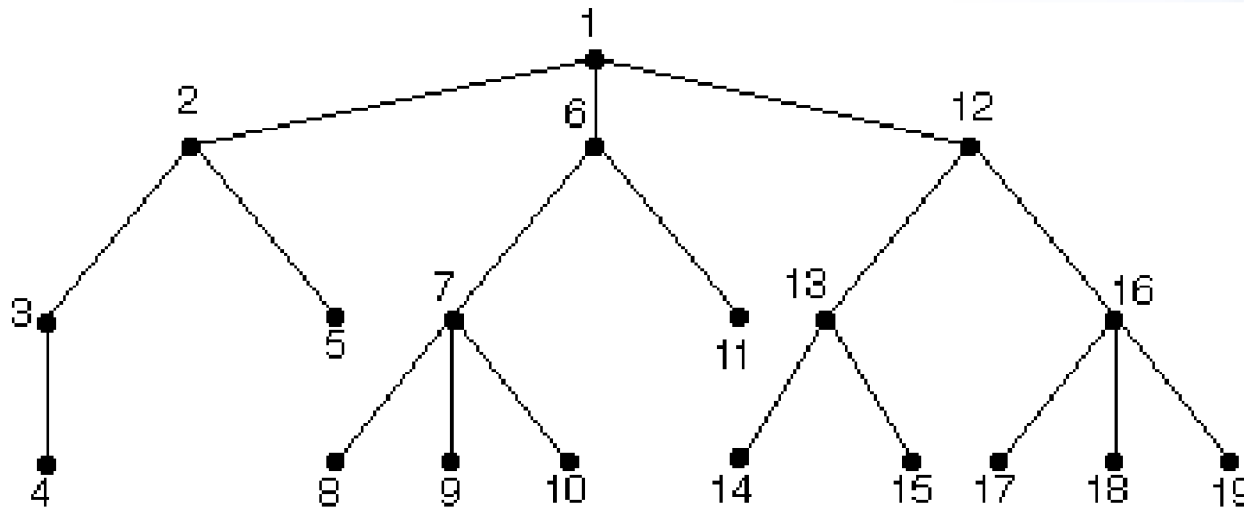
- Uninformed (Blind) Search
 - Breadth First Search (BFS)
 - Depth First Search (DFS)
 - Depth Limited Search
 - Iterative Deepening Search
 - Bidirectional Search
- Informed (Heuristik/Intelligent) Search

Breadth-first search



- Evaluasi dilakukan terhadap simpul-simpul pada suatu level sebelum dilanjutkan pada level berikutnya

Depth-first search



- Simpul yang lebih dalam diperiksa terlebih dahulu → Penelusuran simpul-simpul pada suatu cabang sampai kedalaman yang ditentukan

Pengisian Bak



4 liter



3 liter

Bagaimana mendapatkan air sebanyak 2 liter pada bak 4 liter?

Kaidah:

1. $(X, Y \mid X < 4) \rightarrow (4, Y)$
2. $(X, Y \mid Y < 3) \rightarrow (X, 3)$
3. $(X, Y \mid X > 0) \rightarrow (X - D, Y)$
4. $(X, Y \mid Y > 0) \rightarrow (X, Y - D)$
5. $(X, Y \mid X > 0) \rightarrow (0, Y)$
6. $(X, Y \mid Y < 0) \rightarrow (X, 0)$
7. $(X, Y \mid X + Y > = 4 \wedge Y > 0) \rightarrow (4, Y - (4 - x))$
8. $(X, Y \mid X + Y > = 3 \wedge Y > 0) \rightarrow (X - (3 - Y), 3)$
9. $(X, Y \mid X + Y < = 4 \wedge Y > 0) \rightarrow (X + Y, 0)$
10. $(X, Y \mid X + Y < = 3 \wedge X > 0) \rightarrow (0, X + Y)$

Bak 4 Liter

Bak 3 Liter

Kaidah yang diterapkan

0

0

2

0

3

9

3

0

2

3

3

7

4

2

5

0

2

9

2

0

BFS

- Completeness?
 - Yes
- Time complexity?
 - $O(b^d)$
- Space complexity?
 - $O(b^d)$ 😞
- Optimality?
 - yes

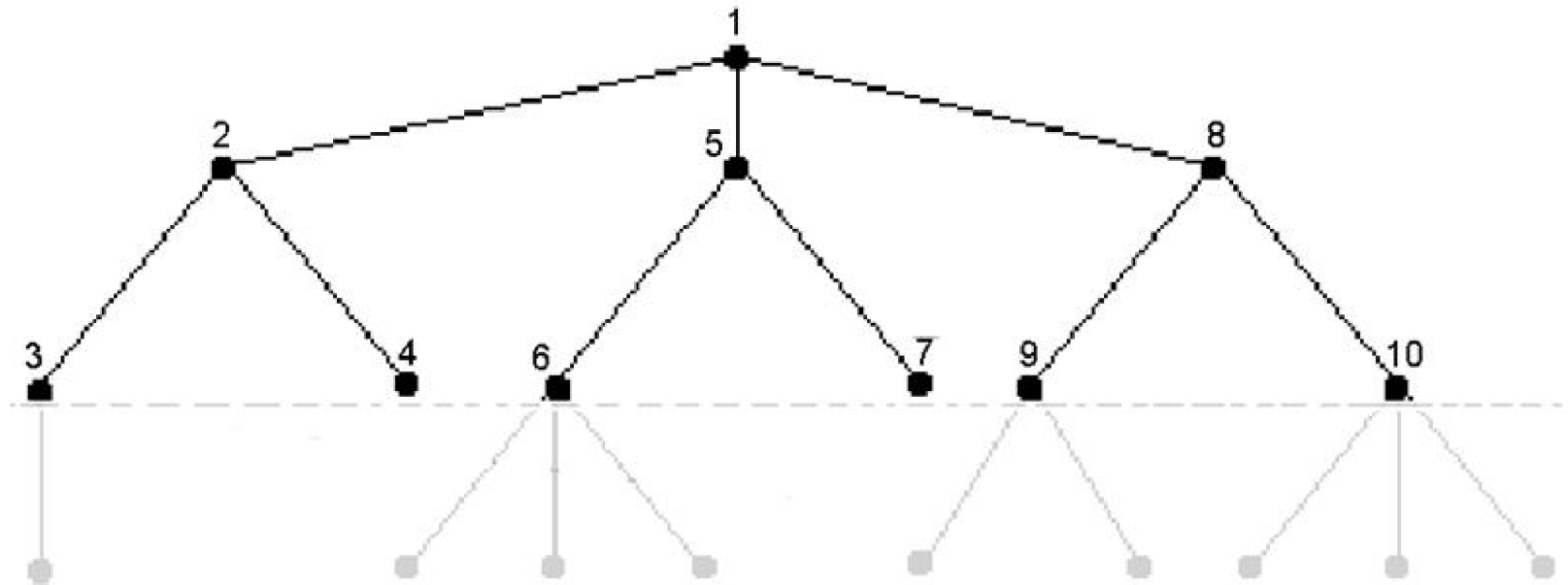
DFS

- Completeness?
 - Yes, assuming state space finite
- Time complexity?
 - $O(|S|)$, can do well if lots of goals
- Space complexity?
 - $O(n)$, n deepest point of search
- Optimality?
 - No 😞

Depth-Limited Search

- **“Practical” DFS**
- However, if we choose a depth limit that is too small, then DLS is not even complete.
- The time and space complexity of DLS is similar to DFS.
- **Completeness:** Yes, if $l \geq d$
- **Time complexity:** b^l
- **Space complexity:** bl
- **Optimality:** No *(b -branching factor, l -depth limit)*

Depth Limited Search (con't)

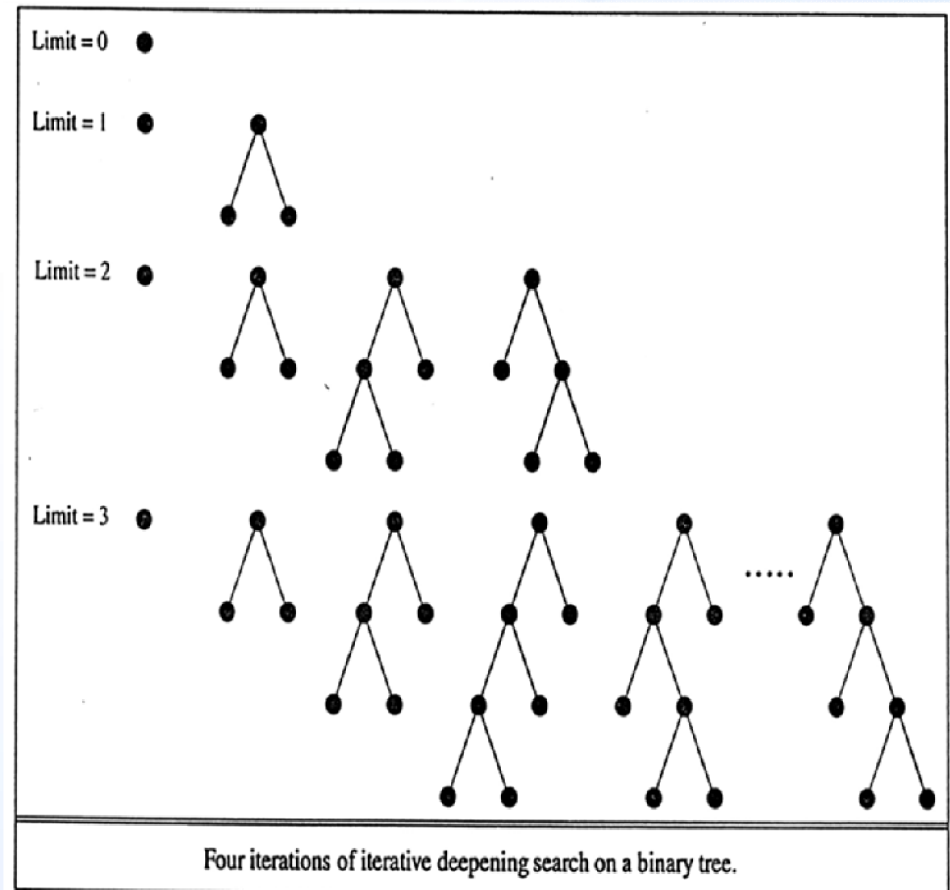


Iterative Deepening Search

- The **hard part** about DLS is **picking a good limit**.
- IDS is a strategy that sidesteps the issue of choosing the best depth limit by **trying all possible depth limits**: first depth 0, then depth 1, the depth 2, and so on.
- In effect, it combines the **benefits of DFS and BFS**.
- It is **optimal and complete**, like BFS and has **modest memory requirements** of DFS.
- IDS may seem wasteful, because so many states are expanded multiple times.
- For most problems, however, the **overhead** of this multiple expansion is actually **rather small**.

Iterative Deepening Search (con't)

- **IDS** is the **preferred** search method when there is a **large search space** and the **depth** of the solution is **not known**.
- **Completeness**: Yes
- **Time complexity**: b^d
- **Space complexity**: bd
- **Optimality**: Yes



Bi Directional Search

- **Search forward** from the Initial state
- And **search backwards** from the Goal state..
- Stop when two meets in the **middle**.
- **Completeness:** Yes
- **Time complexity:** $b^{d/2}$
- **Space complexity:** $b^{d/2}$
- **Optimality:** Yes

Depth-limited Search

DFS, only expand nodes depth $\leq l$.

- Completeness?
 - No, if $l \leq d$. ☹️
- Time complexity?
 - $O(b^l)$
- Space complexity?
 - $O(l)$
- Optimality?
 - No ☹️

Iterative Deepening

Depth limited, increasing 1.

- Completeness?
 - Yes. 😊
- Time complexity?
 - $O(b^d)$, even with repeated work! 😊
- Space complexity?
 - $O(d)$ 😊
- Optimality?
 - Yes 😊

Bidirectional Search

BFS in both directions

Need N^{-1}

How could this help?

– b^l vs $2b^{l/2}$

What makes this hard to implement?