Uninformed Search Strategies

Iterative Deepening Search

ITERATIVE DEEPENING SEARCH

The technique of iterative deepening is based on limited/bounded search idea.

Iterative deepening is depth-first search to a fixed depth in the tree being searched.

If no solution is found up to this depth then the depth to be searched is increased and the whole `bounded' depth-first search begins again.

ITERATIVE DEEPENING SEARCH

Complete, if b is finite

Optimal, if path cost is equal to depth

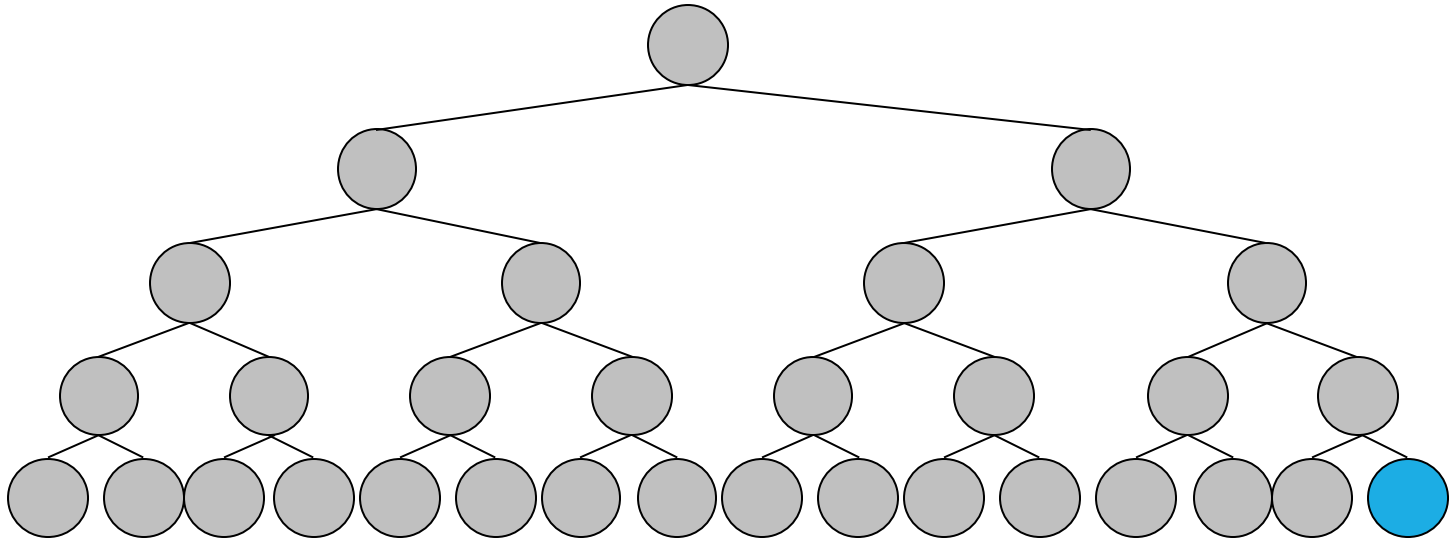
- Guaranteed to return the shallowest goal

Time complexity = $O(b^d)$

Space complexity = O(bd)

Nodes on levels above d are generated multiple times

ITERATIVE DEEPENING SEARCH



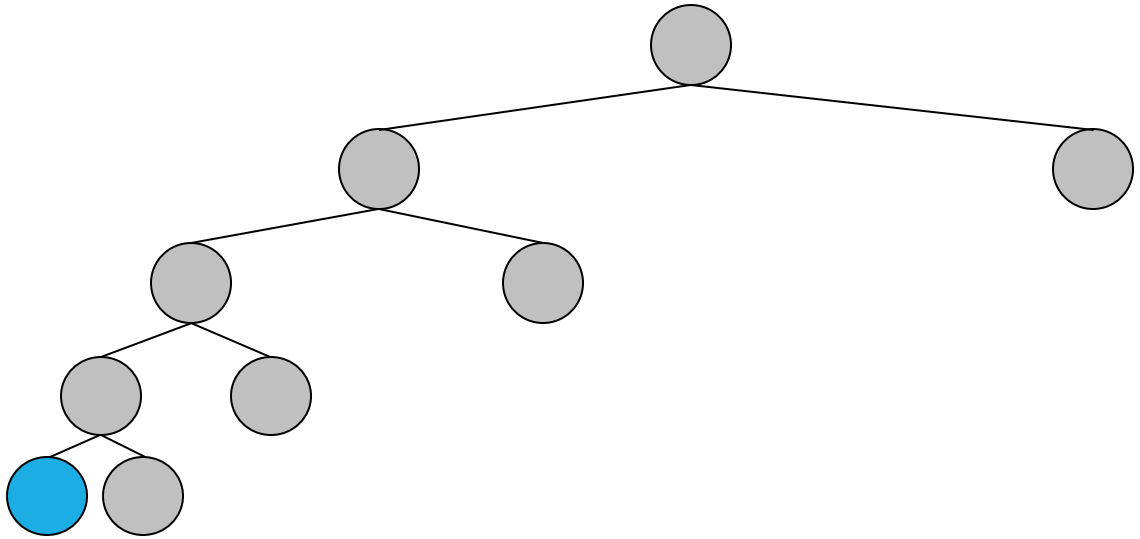
Upper-bound case for time: goal is last node of last branch

- Number of nodes generated:
  - b nodes for each node of d levels (entire tree to depth d)
- Time complexity: all generated nodes $O(b^d)$

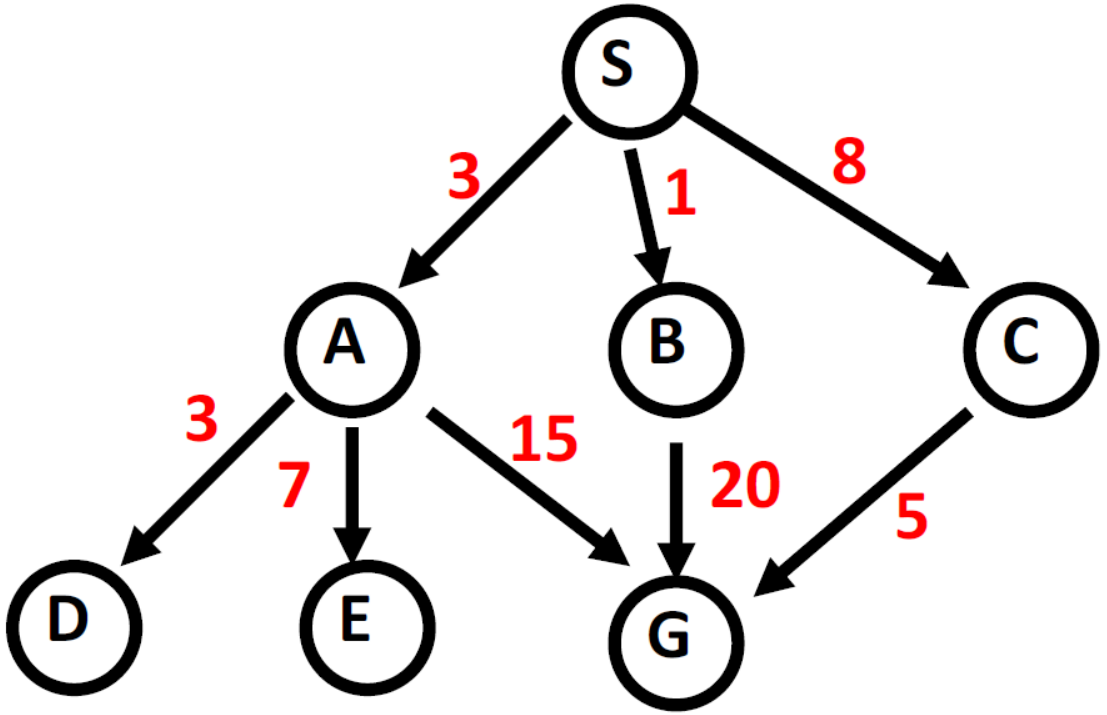
ITERATIVE DEEPENING SEARCH

Upper-bound case for space: goal is last node of first branch

- After that, we start deleting nodes
- Number of generated nodes: b nodes at each of d levels
- Space complexity: all generated nodes = O(bd)

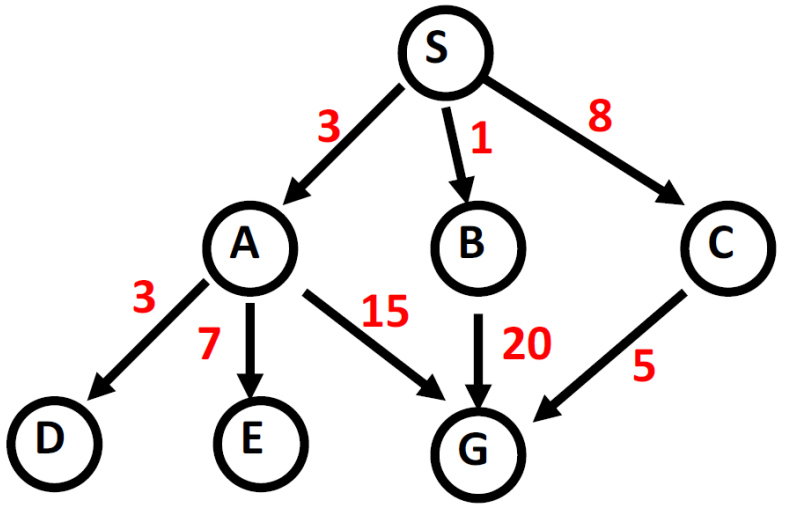


THINK-PAIR-SHARE: IDS (5 MINS)



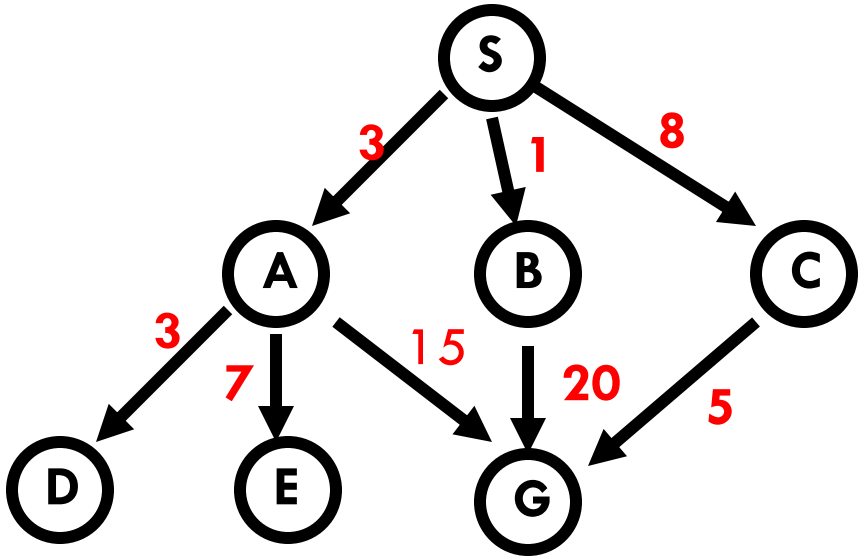
IDS

Expanded node	Nodes list	Limit
	$\{ S^0 \}$	1
S^0	$\{\}$	
	$\{ S^0 \}$	2
S^0	$\{ A^3\ B^1\ C^8 \}$	
A^3	$\{ B^1\ C^8 \}$	
B^1	$\{C^8 \}$	
C^8	$\{\}$	
	$\{ S^0 \}$	3
S^0	$\{ A^3\ B^1\ C^8 \}$	
A^3	$\{ D^6\ E^{10}\ G^{18}\ B^1\ C^8 \}$	
D^6	$\{ E^{10}\ G^{18}\ B^1\ C^8 \}$	
E^{10}	$\{ G^{18}\ B^1\ C^8 \}$	
G^{18}	$\{ B^1\ C^8 \}$	



Solution path found is S A G, cost 18
Number of nodes expanded (including goal node) = 10

COMPARING SEARCH STRATEGIES



- **Breadth-First Search**:
  - Expanded nodes: S A B C D E G
  - Solution found: S A G (cost 18)

- **Depth-First Search:**
  - Expanded nodes: S A D E G
  - Solution found: S A G (cost 18)

- **Uniform-Cost Search** (*This is the only uninformed search that worries about costs*):
  - Expanded nodes: S A D B C E G
  - Solution found: S C G (cost 13)

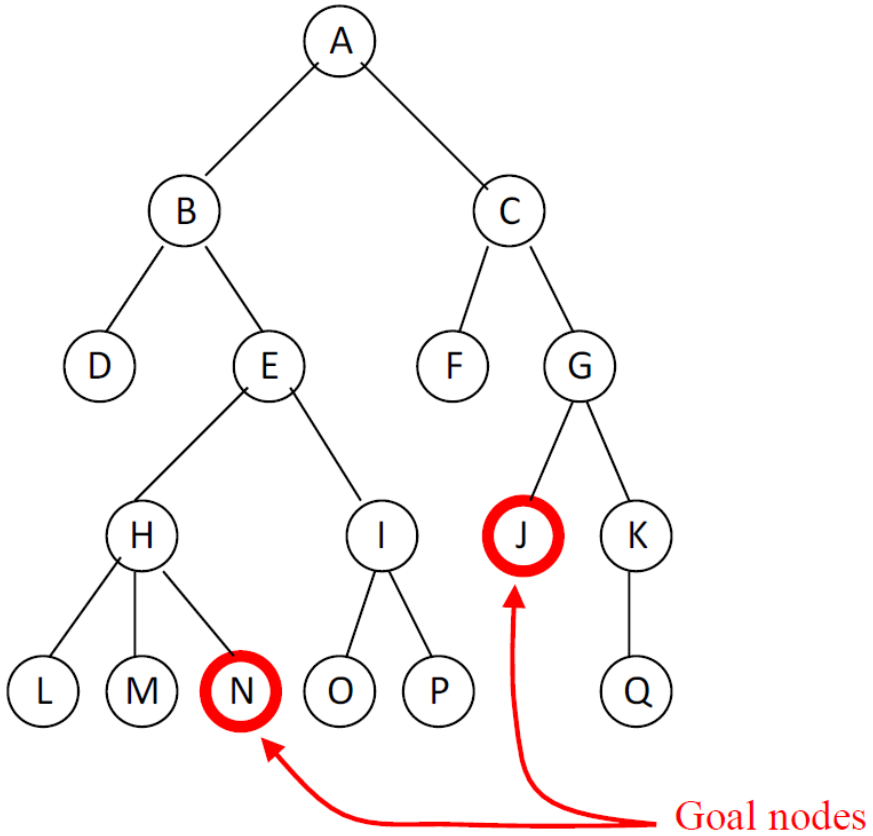
- **Iterative-Deepening Search**:
  - Expanded nodes: S S A B C S A D E G
  - Solution found: S A G (cost 18)

UNINFORMED SEARCH STRATEGIES

Summary and Examples

GRAPH SEARCH

- A tree search starts at the root
- Explores nodes looking for a goal node
  - a node that satisfies certain conditions, depending on the problem
- For some problems, any goal node is acceptable (N or J);
- For other problems, a minimum-depth goal node is needed
  - A goal node nearest the root (only J)



GRAPH SEARCH STRATEGIES

- Depth-first
  - Low memory requirement.
  - Could get stuck exploring infinite paths.
  - Used if there are many solutions and you need only one.

- Breadth-first
  - High memory requirement.
  - Doesn’t get stuck.
  - Finds the shortest path (minimum number of steps).

GRAPH SEARCH STRATEGIES

- Breadth-first search and uniform-cost search are optimal graph search strategies.

- Iterative deepening search and depth-first search can follow a non-optimal path to the goal.

REAL-WORLD APPLICATIONS

- Web crawling (Google search – breadth first, infinite)
- Facebook (friends finder, 2 or 3 levels of separation)
- Network communication (Internet broadcast, wireless sensor networks)
- Garbage collection (Java, Python, …etc – breadth-first search)
- Vehicle Routing
- Task Assignment
- Solving puzzles
- Social Networks
- Fault Detection
- Circuit Verification
- Layout Design
- Traffic Congestion Mangement

DEALING WITH CYCLES AND AVOIDING REPEATED STATES

In increasing order of effectiveness in reducing size of state space and with increasing computational costs:

1. Do not return to the state you just came from.

2. Do not create paths with cycles in them.

3. Do not generate any state that was ever created before.

Net effect depends on frequency of “loops” in state space.

COMPARISON BETWEEN UNIFORMED SEARCH TECHNIQUES

nparison of Search Strategies

me complexity, space complexity, optimality, completeness

Breadth-First	Uniform-Cost	Depth-First	Depth-Limited	Iterative Deepening	Bidirectional (if applicable)
Yes[a]	Yes[a,b]	No	No	Yes[a]	Yes[a,d]
$O(b^{d+1})$	$O(b^{\lceil C^*/\epsilon \rceil})$	$O(b^m)$	$O(b^\ell)$	$O(b^d)$	$O(b^{d/2})$
$O(b^{d+1})$	$O(b^{\lceil C^*/\epsilon \rceil})$	$O(bm)$	$O(b\ell)$	$O(bd)$	$O(b^{d/2})$
Yes[c]	Yes	No	No	Yes[c]	Yes[c,d]

ching factor

h of solution,

mum depth of the search tree,

h limit,

of the optimal solution,

mal cost of an action

Superscripts:

a) b is finite

b) if step costs not less than ϵ

c) if step costs are all identical

d) if both directions use breadth-first search

COMPARING UNINFORMED SEARCH STRATEGIES

Criteria	Breadth-First	Uniform-Cost	Depth-First	Depth-Limited	Iterative Deepening
Complete	Conditionally Yes	Conditionally Yes	Conditionally Yes	Conditionally Yes if l >= d	Conditionally Yes
Optimal	Conditionally Yes	Yes	No	No	Conditionally Yes
Time	b^d	If costs are the same: b^d	b^m	b^l	b^d
Space	b^d	If costs are the same: b^d	bm	bl	bd

b – branching factor
d – depth of optimal solution
m – maximum depth
l – depth limit

INFORMED SEARCH

TYPES OF SEARCH

Uninformed search strategies

- Also known as “blind search,” uninformed search strategies use no information about the likely “direction” to the goal node(s)

Informed Search

- Also known as “heuristic search,” informed search strategies use information about the domain to head in the general direction of the goal node(s)

INFORMED SEARCH

A search strategy which searches the ***most promising*** branches of the state-space first.
Thus, is should:

- find a solution more quickly,
- specially when time is limited,
- often finds a better solution, since more profitable parts of the state-space can be examined, while ignoring the unprofitable parts.

Tries to use the knowledge known about the problem to prune the search.

This knowledge is used in the form of a ***heuristic function***.

The heuristic function should give an estimate of the ***distance to the goal***.

HEURISTIC

Merriam-Webster's Online Dictionary

- Heuristic (pron. \hyu-ˈris-tik\): adj. [from Greek *heuriskein* to discover.] involving or serving as an ***aid to learning***, discovery, or problem-solving by experimental and especially trial-and-error methods.

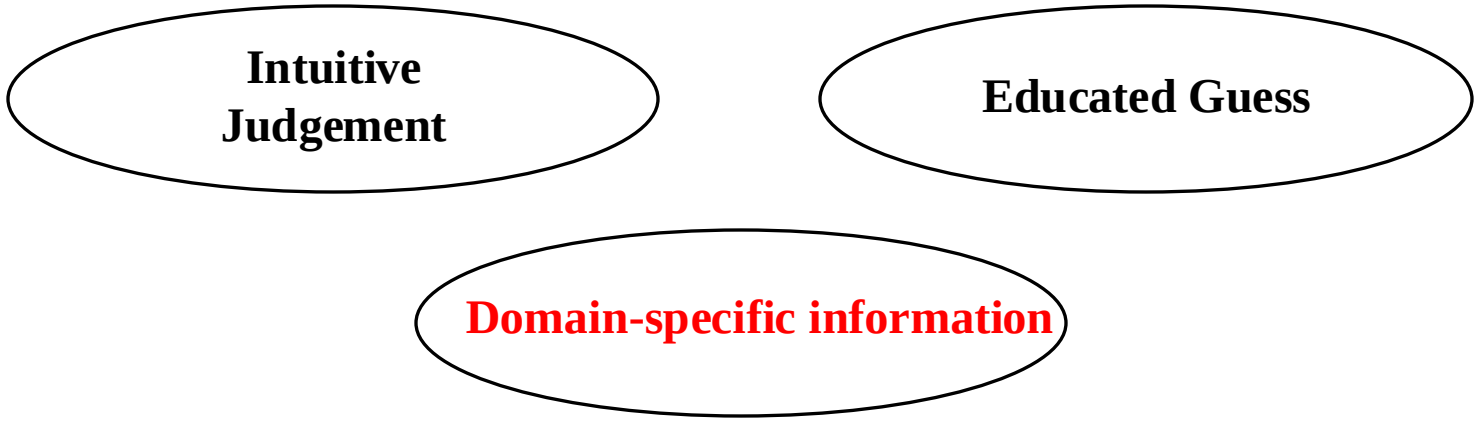
The Free On-line Dictionary of Computing

- (programming) A ***rule of thumb***, ***simplification***, or ***educated guess*** that reduces or limits the ***search for solutions*** in domains that are difficult and poorly understood. Unlike algorithms, heuristics do not guarantee optimal, or even feasible, solutions and are often used with no theoretical guarantee.
- (algorithm) ***approximation algorithm***.

HEURISTIC

From WordNet (r) 1.6

- Heuristic adj
  - (computer science) relating to or using a heuristic rule
  - of or relating to a general formulation that serves to guide investigation [ant: algorithmic] n : a commonsense rule (or set of rules) intended to increase the probability of solving some problem [syn: heuristic rule, heuristic program]



ALGORITHM VS HEURISTIC

*An **Algorithm** is a clearly defined set of instructions or steps to solve a problem, **Heuristics** based problem solving involves utilizing guessing, learning and discovery to seek a solution, though no guarantees.*

*Hence, if you know how to solve a problem then devise an **algorithm for it- assuming the algorithm is fast enough**. If the solution process is more of trial and error, a gut feeling, a guess, or time is of the essence that a good enough solution is acceptable, then consider heuristic based problem solving.*

Interesting read:

When to use which heuristic: A rational solution to the strategy selection problem

By: Falk Lieder (falk.lieder@berkeley.edu)
Thomas L. Griffiths (tom griffiths@berkeley.edu)

INFORMED SEARCH

Add domain-specific information to select the ***best path*** along which to continue searching

Define a heuristic function ***h(n)*** that estimates the "goodness" of a node *n*.

- Specifically, *h*(*n*) = **estimated cost** (or distance) of minimal cost path from node ***n*** **to a goal state**.

The heuristic function is an estimate of how close we are to a goal, based on domain-specific information that is computable from the current state description.

All **domain knowledge** used in the search is encoded in the **heuristic function *h*()**.

HEURISTICS

Heuristic search is an example of a "**weak method**" because of the limited way that domain-specific information is used to solve the problem.

Examples:

- 8-puzzle: Number of tiles out of place
- 8-puzzle: Sum of distances each tile is away from its goal position

In general:

- $h(n) \geq 0$ for all nodes n
- $h(n) = 0$ implies that n is a goal node
- $h(n) = \infty$ implies that n is a dead-end that can never lead to a goal

WEAK VS. STRONG METHODS

Strong methods are those designed to address a specific type of problem

Weak methods are general approaches that may be applied to many types of problems

We use the term weak methods to refer to methods that are extremely *general* and not tailored to a specific situation.

Called "weak" methods because they do not take advantage of more powerful domain-specific heuristics

WEAK VS. STRONG METHODS

Examples of weak methods include

- **Means-ends analysis** is a strategy in which we try to represent the current situation and where we want to end up and then look for ways to shrink the differences between the two.

- **Space splitting** is a strategy in which we try to list the possible solutions to a problem and then try to rule out classes of these possibilities.

- **Subgoaling** means to split a large problem into several smaller ones that can be solved one at a time.

ADMISSIBLE HEURISTIC FUNCTION

A heuristic function is *admissible* if it never overestimates the cost of reaching the goal.

An admissible heuristic is also known as an ***optimistic*** *heuristic*