Evaluating a Search Strategy

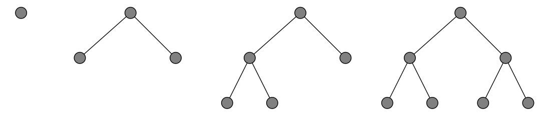
Completeness: is the strategy guaranteed to find a solution when there is one?

Time Complexity: how long does it take to find a solution?

Space Complexity: how much memory does it need?

Optimality: does the strategy find the highest-quality solution when there are several different solutions?

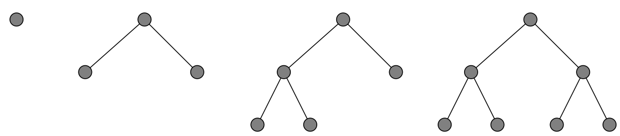
Uninformed search: BFS



Consider paths of length 1, then of length 2, then of length 3, then of length 4,....

Time and Memory Requirements for BFS – $O(b^{d+1})$

Let b = branching factor, d = solution depth, then the maximum number of nodes *generated* is:
$b + b^2 + ... + b^d + (b^{d+1} - b)$



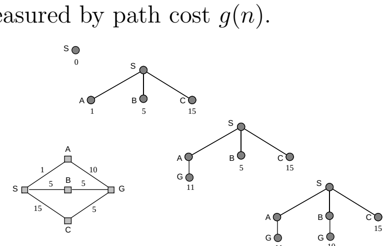
Time and Memory Requirements for BFS – $O(b^{d+1})$

b = 10
10000 nodes/second
each node requires 1000 bytes of storage

depth	nodes	time	memory
2	1100	.11 sec	1 meg
4	111,100	11 sec	106 meg
6	10^7	19 min	10 gig
8	10^9	31 hrs	1 tera
10	10^{11}	129 days	101 tera
12	10^{13}	35 yrs	10 peta
14	10^{15}	3523 yrs	1 exa

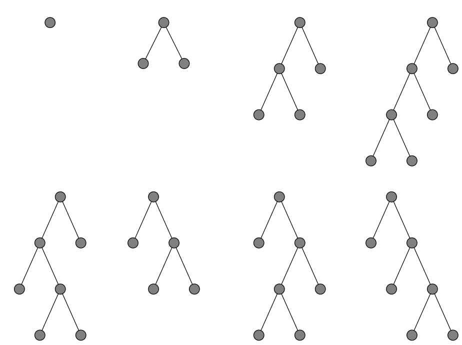
Uniform-cost Search

Use BFS, but always expand the lowest-cost node on the fringe as measured by path cost $g(n)$.



Requirement: $g(\text{Successor}(n)) \geq g(n)$

Uninformed search: DFS



DFS vs. BFS

	Complete?	Optimal?	Time	Space
BFS	YES	YES	$O(b^{d+1})$	$O(b^{d+1})$
DFS	finite depth	NO	$O(b^m)$	$O(bm)$

m is maximum depth

Time

$m = d$ — DFS typically wins

$m > d$ — BFS might win

m **is infinite** — BFS probably will do better

Space

DFS almost always beats BFS

Which search should I use?

Depends on the problem.

If there may be infinite paths, then depth-first is probably bad. If goal is at a known depth, then depth-first is good.

If there is a large (possibly infinite) branching factor, then breadth-first is probably bad.

(Could try **nondeterministic** search. Expand an open node at random.)

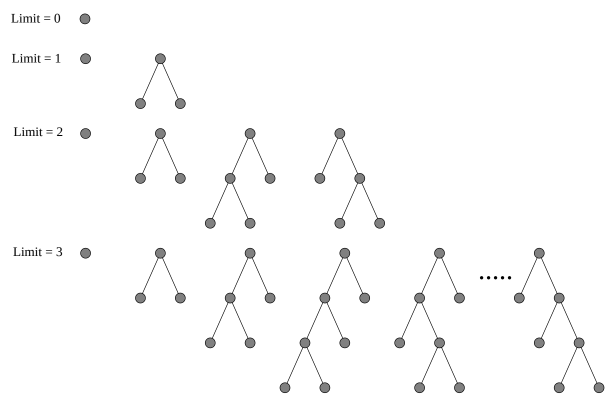
Iterative Deepening [Korf 1985]

Idea:
Use an *artificial* depth cutoff, c.

If search to depth c succeeds, we're done. If not, increase c by 1 and start over.

Each iteration searches using DFS.

Iterative Deepening

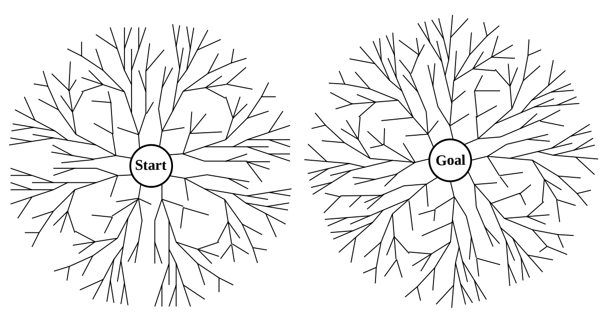


Cost of Iterative Deepening

space: $O(bd)$ as in DFS, **time:** $O(b^d)$

b	ratio of IDS to DFS
2	3
3	2
5	1.5
10	1.2
25	1.08
100	1.02

Bidirectional Search



- Search forward from the start state and backward from the goal state simultaneously and stop when the two searches meet the middle.
- If branching factor = b from both directions, and solution exists at depth d, then need only $O(2b^{d/2}) = O(b^{d/2})$ steps.
- Example b = 10, d = 6 then BFS needs 1,111,111 nodes and bidirectional search needs only 2,222.
  - What does it mean to search backwards from a goal?
  - What if there is more than one goal state? (chess).

Comparing Search Strategies

Criterion	Breadth-First	Uniform-Cost	Depth-First	Iterative Deepening	Bidirectional (if applicable)
Time	b^{d+1}	$b^{1+\frac{C^*}{\epsilon}}$	b^m	b^d	$b^{d/2}$
Space	b^{d+1}	$b^{1+\frac{C^*}{\epsilon}}$	bm	bd	$b^{d/2}$
Optimal?	yes	yes	no	yes	yes
Complete?	yes	yes	no	yes	yes

***Note that many of the "yes's" above have caveats, which we discussed when covering each of the algorithms.