

Uninformed Search

Introduction to Artificial Intelligence

Dr. Robin Burke



Review

- Problem domain

- Static

- only our actions change the world

- Deterministic

- actions always work the way we expect

- Fully-observable

- we always know everything we need about the world

- To solve

- World = state representation

- Solution = sequence of operators

- transform the initial state into the goal state

State Space



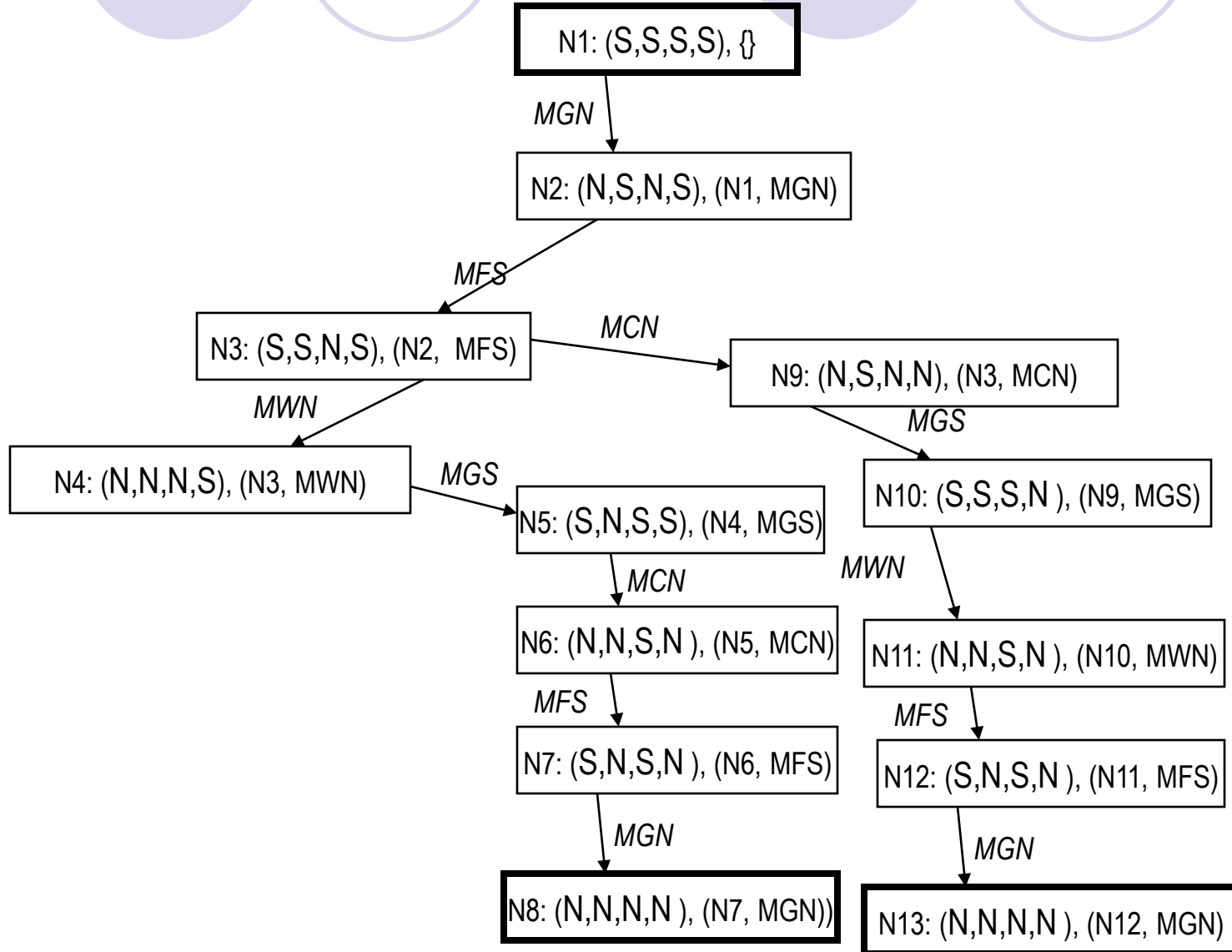
- The collection of all world states
- Connected by edges which are operations
- Solution
 - a path through the state space
 - leads from initial state to goal state
- Problem solving = graph search

Search Tree

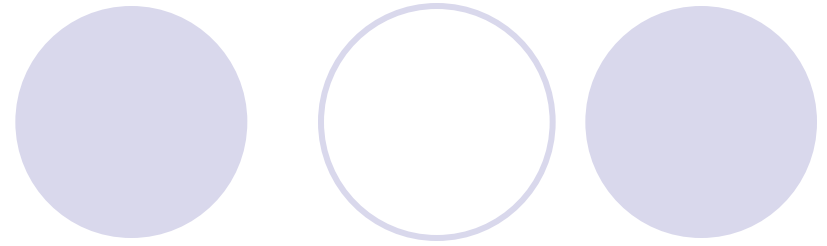


- For better bookkeeping
 - transform the state space into a search tree
- Nodes
 - not just states but
 - state, parent, action, *cost*, *depth*
 - I said path, but you really don't want a million copies of the same path
 - (parent, edge) are sufficient to reconstruct

Search Tree (ignore illegal states)



Basic idea



- Given
 - a set of nodes N
 - a successor function f that
 - takes a node n
 - returns all of the nodes S reachable from n in a single action
- Algorithm
 - pick n from N (somehow)
 - $s = \text{state}(n)$
 - $p = \text{path}(n)$
 - $S = f(s)$
 - for all $s', a \in S$
 - if s is solution, done
 - if s is illegal, discard
 - else
 - create new node $n_a = \langle s', (n,a) \rangle$
 - $N = N - n + n_a$
 - repeat

Search strategies



- All of the interesting action is
 - maintenance of the active nodes
 - “search frontier”
- Classes of strategies
 - uninformed search
 - no knowledge of the search space is assumed
 - do not prefer one state over another
 - informed search
 - use knowledge of the search space
 - heuristic search

Uninformed Strategies

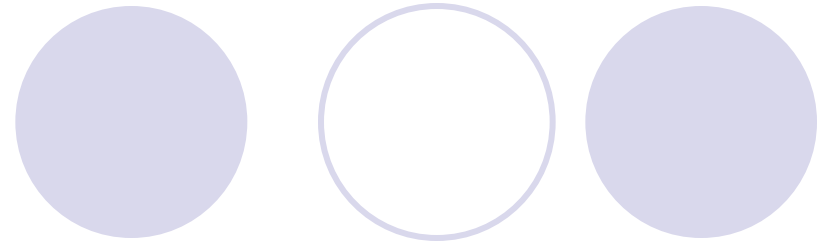


- Depth-first search
- Breadth-first search
- Iterative deepening

Characterizing search strategies

- **Completeness**
 - does it always find the answer?
- **Time**
 - how long does it take?
 - worst-case run time performance / complexity
- **Space**
 - how much memory do we need?
 - worst-case space complexity
- **Parameters**
 - b : branching factor of the search space
 - choices at each node
 - d : depth of the solution
 - # of steps
 - m : maximum depth of the tree
 - could be infinite

Depth-first search



- Unexamined nodes
 - last-in first-out stack
- Meaning
 - grab the top node on the node list
 - replace with its children
 - grab the top one of these
 - etc.

Example: path planning

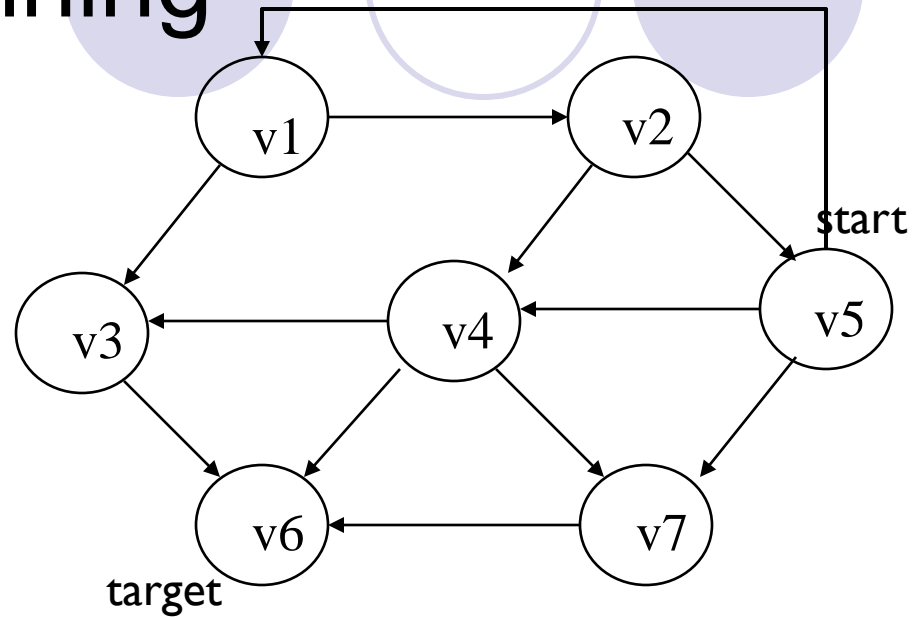
- states = positions

- initial state

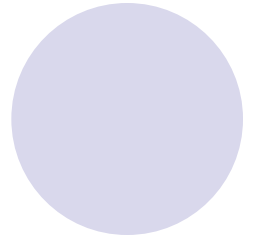
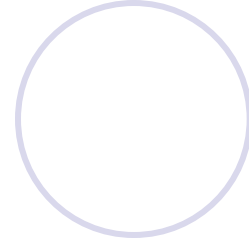
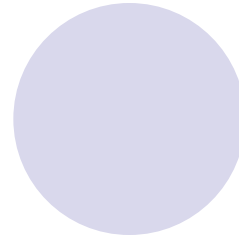
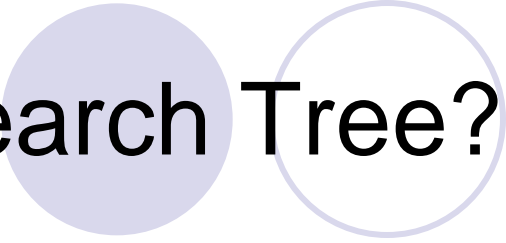
 - v5

- goal state

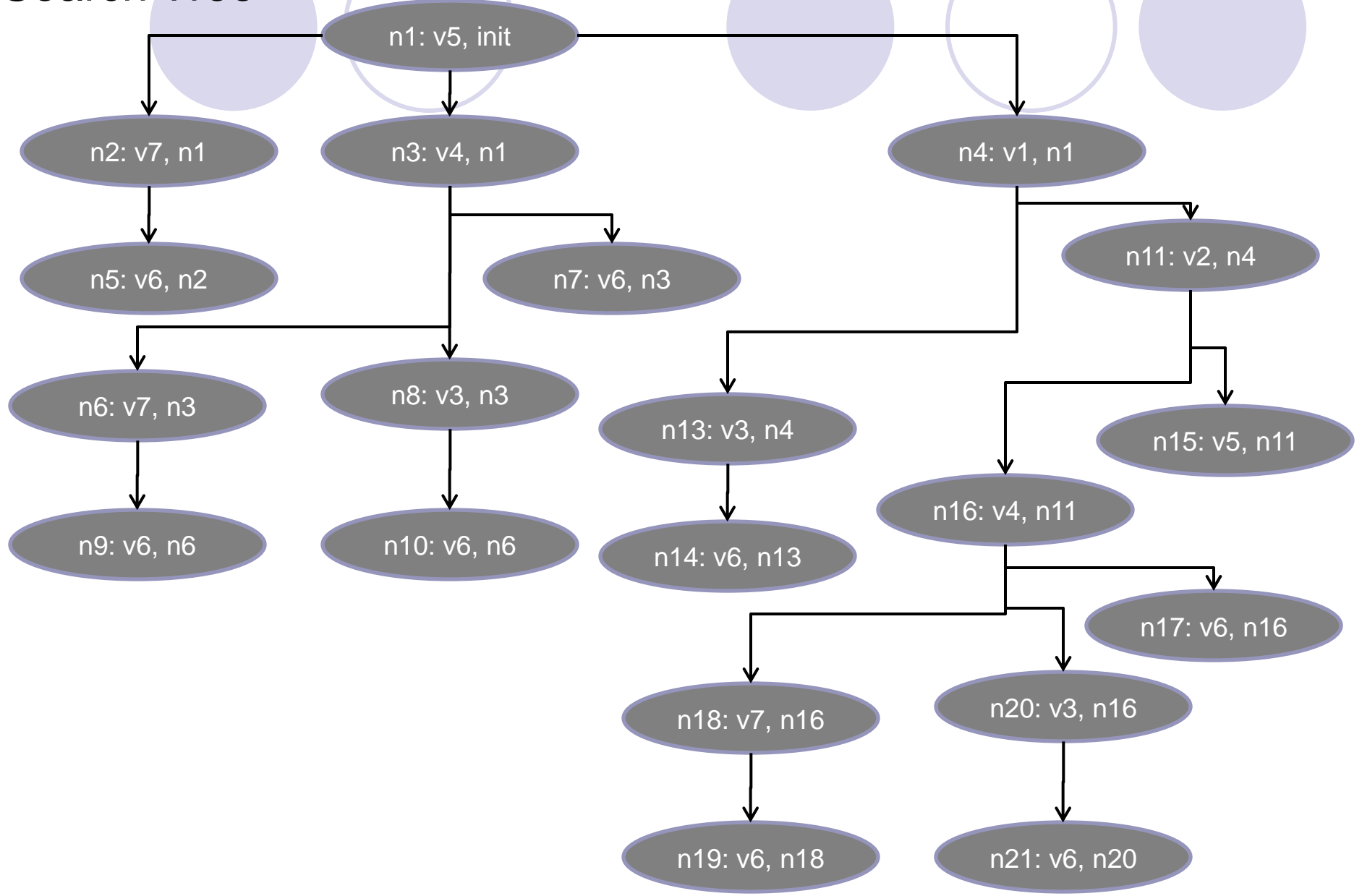
 - v6



Search Tree?



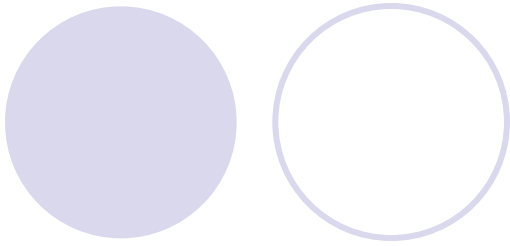
Search Tree



DFS

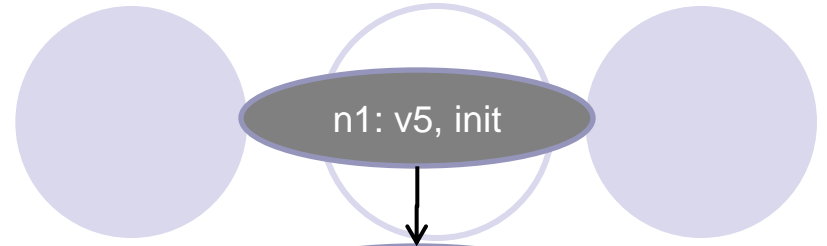
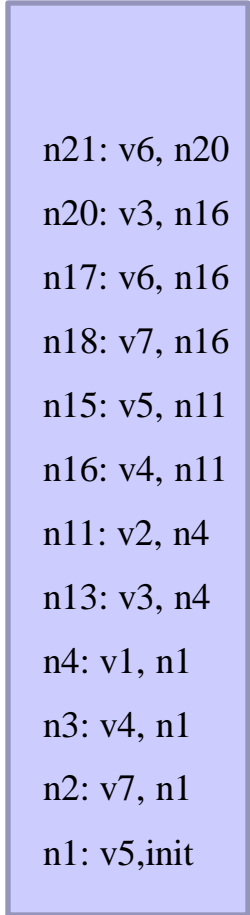


- Start with search node n1
- Add successors to stack
 - *assume we enumerate clockwise from right*
 - n2, n3, n4
- Pop off the most recent one
 - n4
- Expand that node
- etc.



open

closed

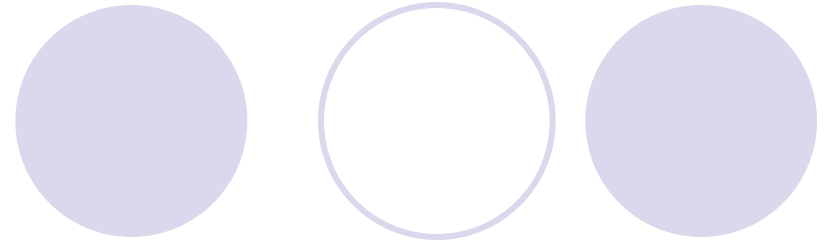


goal
state

repeated
state

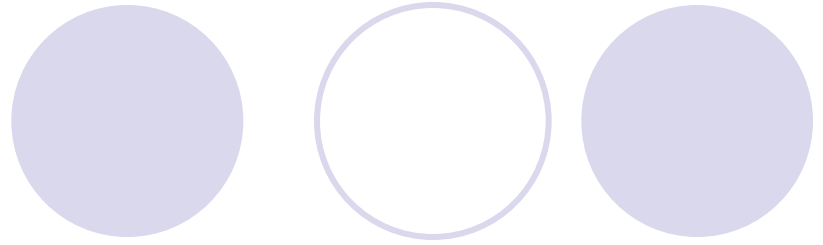
path: v5, v1, v2, v4, v3, v6

Properties of DFS

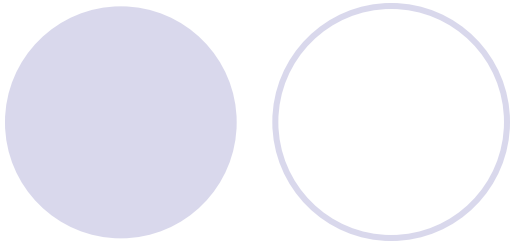


- Complete?
 - Not always
 - fails in infinite-depth spaces
 - Must avoid repeated states along path
- Time?
 - $O(b^m)$: terrible if m is much larger than d
 - but if solutions are dense, may be OK
- Space?
 - $O(bm)$, i.e., linear space, if no repeated states
 - but closed list makes it $O(b^m)$ in worst case
- Optimal?
 - No
 - Returns first answer, not necessarily shortest path

Breadth-first search



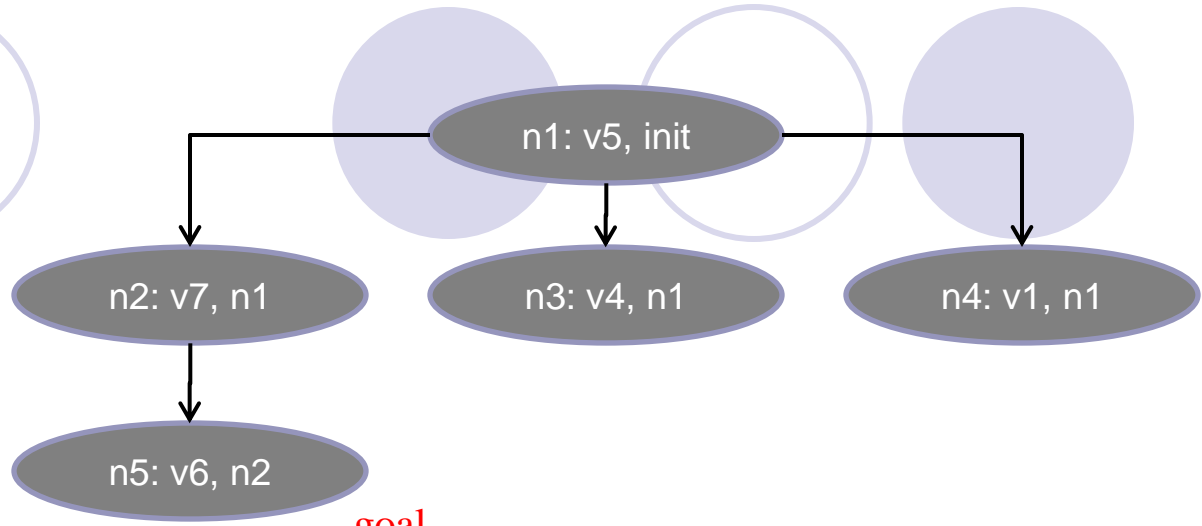
- Unexamined nodes
 - first-in first-out queue
- Meaning
 - grab the first thing on the queue
 - put its children on the end
 - grab the next thing
- Effect
 - we don't examine any paths of length k
 - until we've looked at all paths of length $k-1$
 - achieves optimality



open

closed

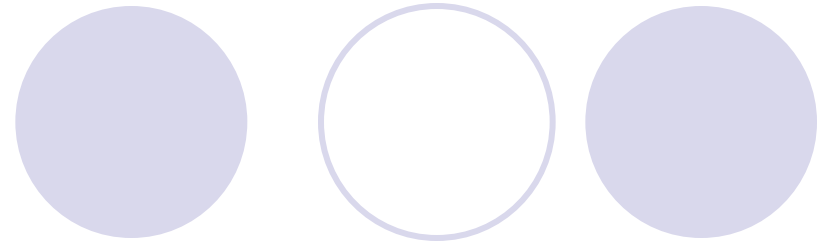
n11: v2, n4
n13: v3, n4
n7: v6, n3
n8: v3, n3
n6: v7, n3
n5: v6, n2
n4: v1, n1
n3: v4, n1
n2: v7, n1
n1: v5,init



goal
state

path: v5, v7, v6

Properties of BFS



- Complete

- If b is finite

- Time

- $1+b+b^2+b^3+\dots +b^d + b(b^d-1) = O(b^{d+1})$

- Looks better than $O(b^m)$

- Space

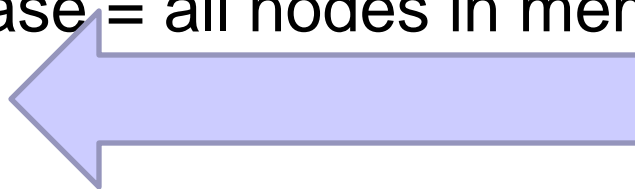
- Worst case = all nodes in mem

- $O(b^{d+1})$

- Optimal

- Yes, for uniform cost case

Often a bigger problem than time

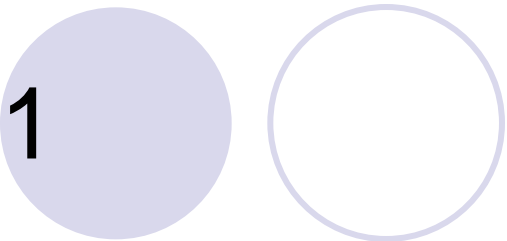


Iterative Deepening



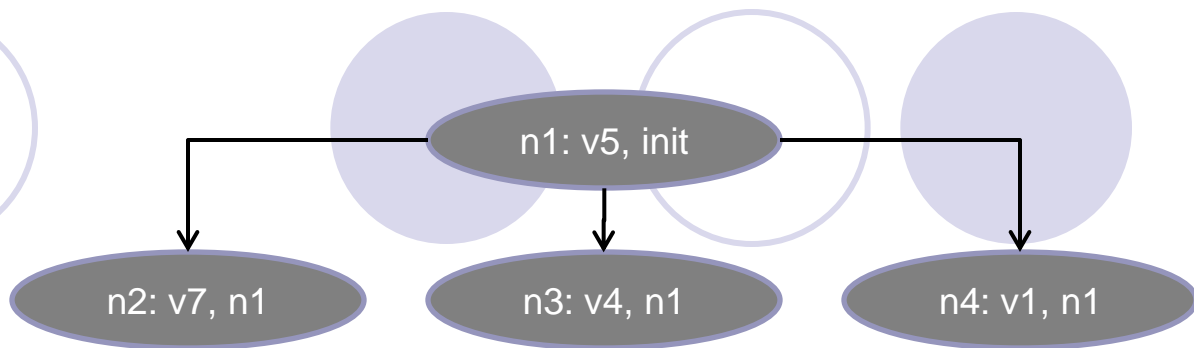
- How to get optimality of BFS
 - without the space penalty
- Counter-intuitive idea
 - create a depth limit for DFS
 - DFS(k)
 - fails if solution is deeper than k
 - do DFS(k) over and over again
 - as k increases from 1 to d
- Seems inefficient
 - throw away b_k nodes after DFS(k)
 - re-generate them for DFS(k+1)

k=1

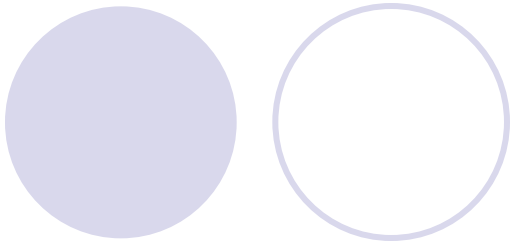


open

closed

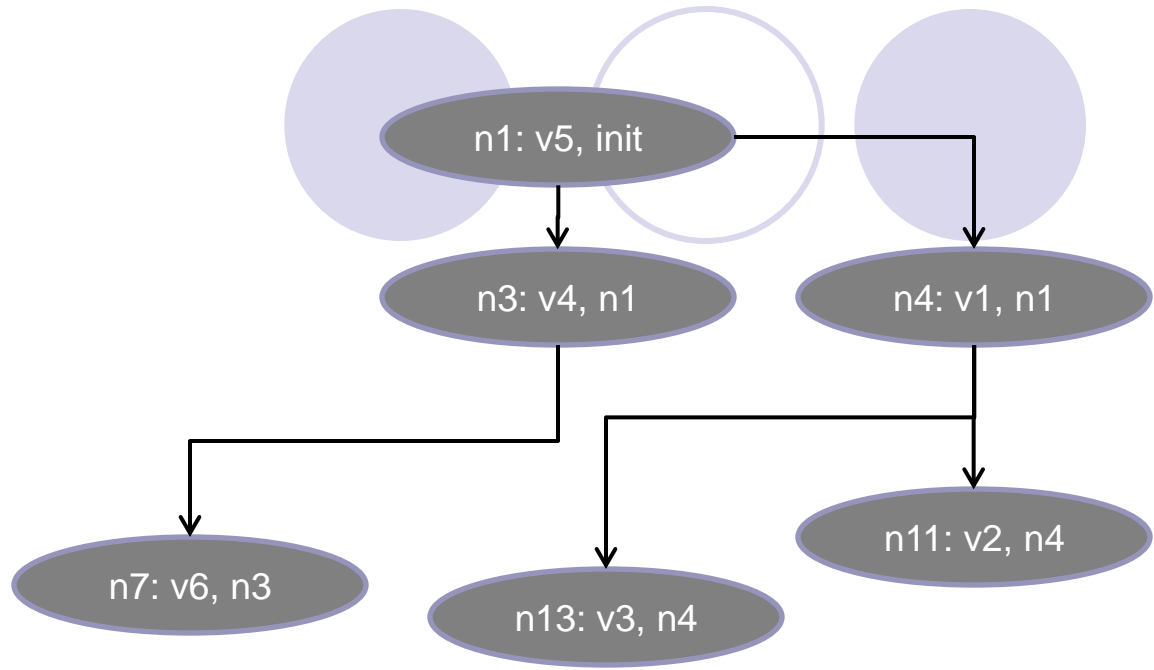
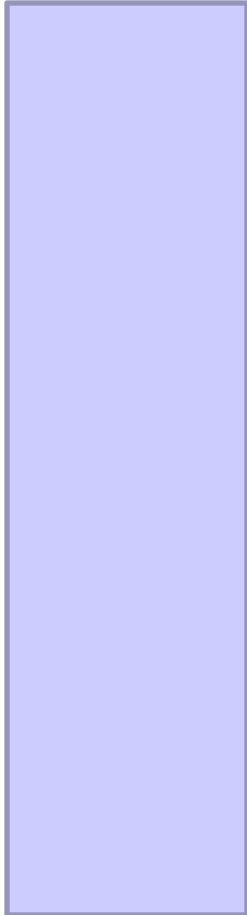
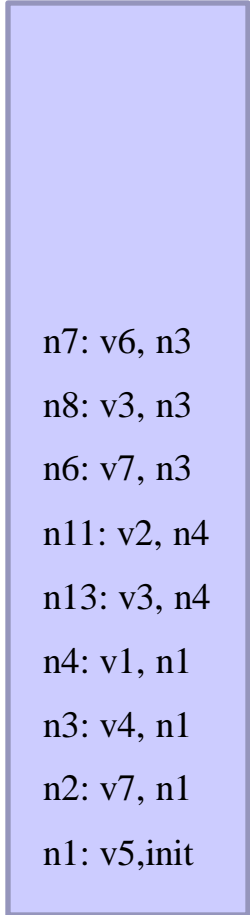


failure



open

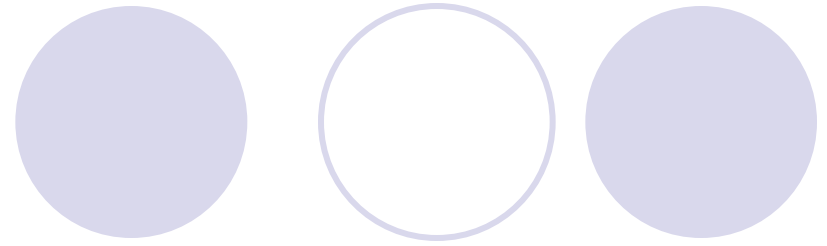
closed



goal
state

path: v5, v4, v6

Wasted time?



- Number of nodes generated in a depth-limited search to depth d with branching factor b :

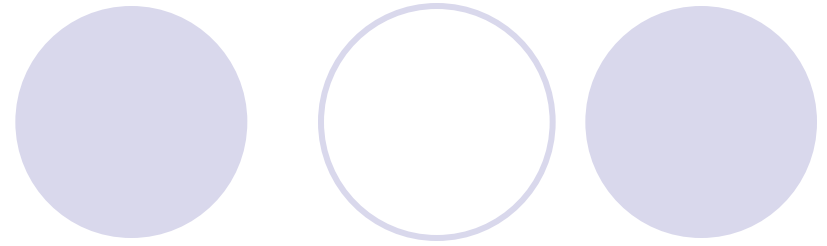
$$N_{DLS} = b^0 + b^1 + b^2 + \dots + b^{d-2} + b^{d-1} + b^d$$

- Number of nodes generated in an iterative deepening search to depth d with branching factor b :

$$N_{IDS} = (d+1)b^0 + db^1 + (d-1)b^2 + \dots + 3b^{d-2} + 2b^{d-1} + 1b^d$$

- For $b = 10$, $d = 5$,
 - $N_{DLS} = 1 + 10 + 100 + 1,000 + 10,000 + 100,000 = 111,111$
 - $N_{IDS} = 6 + 50 + 400 + 3,000 + 20,000 + 100,000 = 123,456$
- Overhead = $(123,456 - 111,111)/111,111 = 11\%$
- In general
 - N_{IDS} is dominated by b^d term
 - $O(b^d)$

Properties of IDS



- Complete?
 - Yes
- Time?
 - $O(b^d)$
 - same as BFS, maybe some overhead
- Space?
 - $O(bm)$, i.e., linear space, if no repeated states
- Optimal?
 - Yes, for uniform cost

Bi-directional search

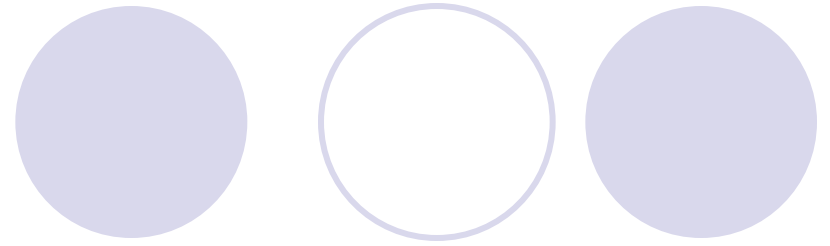


- In some (but not all) search spaces
 - steps are reversible
 - goal state unique
- Counter-example
 - get to work
- If so
 - we can work backwards from end state
 - as well as forward from initial state
 - two BFS operations

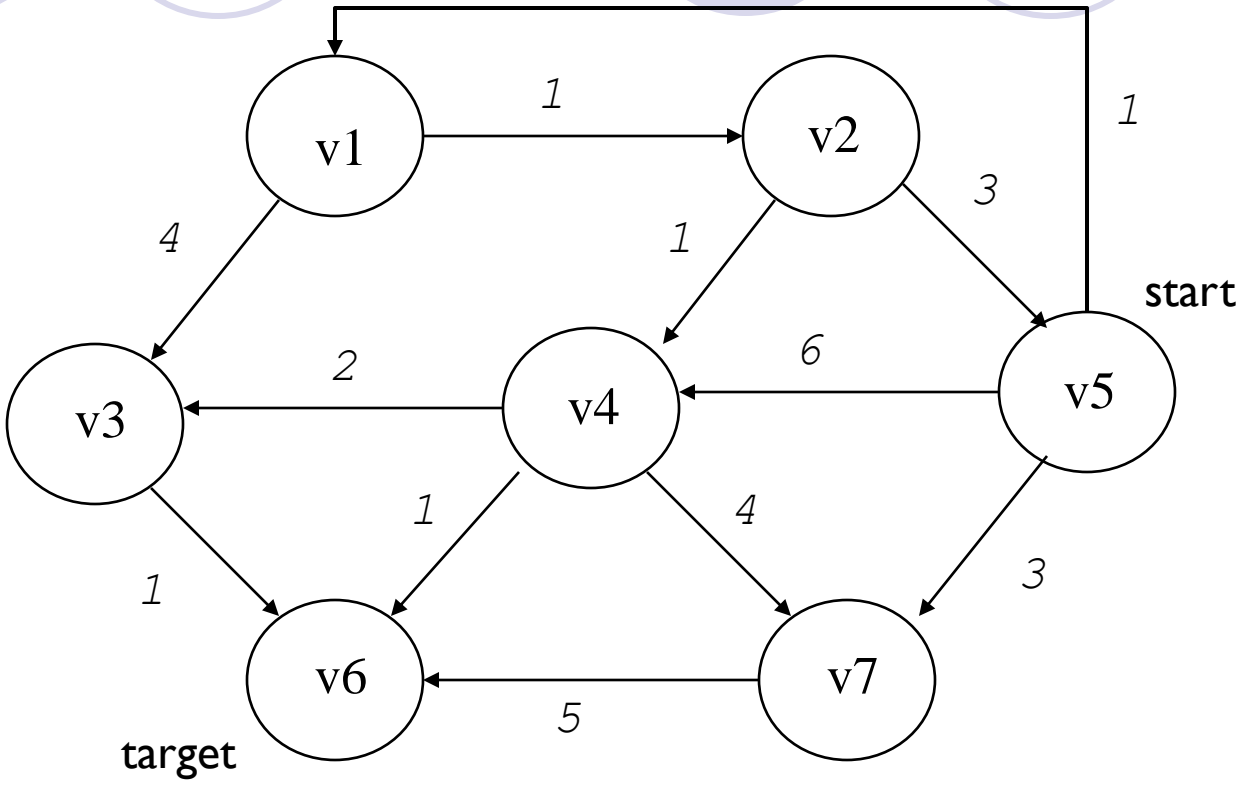
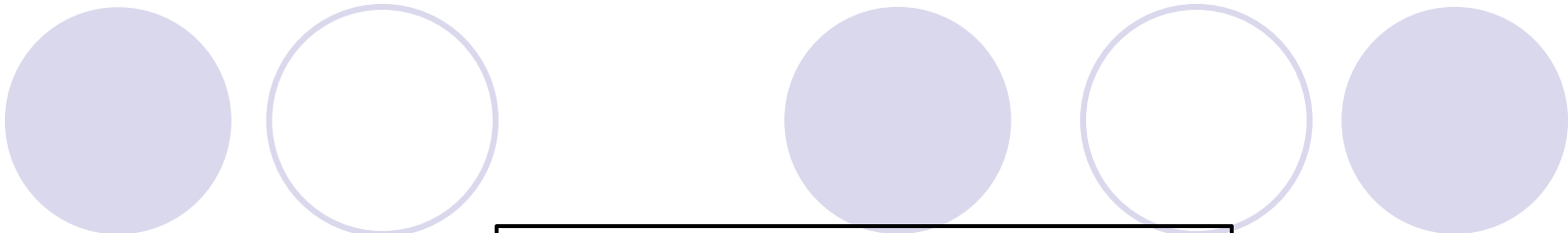
Properties of Bidirectional BFS

- Complete?
 - Yes
- Time?
 - $O(b^{d/2})$
 - can be a big savings
- Space?
 - $O(b^{d/2})$
 - space still a problem
- Optimal?
 - Yes, for uniform cost

Paths with costs



- What if different operations have different costs / benefits?
 - distance
 - \$\$
 - energy
 - *in games, health*





Costs

- If costs are not uniform
 - then search must take this into account
 - otherwise, optimality is lost
 - shortest in terms of # operations may be very expensive
- Question
 - how to organize search to take cost into account?

A decorative graphic at the top of the slide consists of two groups of three circles. The left group has a solid light purple circle on the left, a white circle with a light purple outline in the middle, and a solid light purple circle on the right. The right group has a solid light purple circle on the left, a white circle with a light purple outline in the middle, and a solid light purple circle on the right.

Thursday

- Read Chapter 4.1-4.2