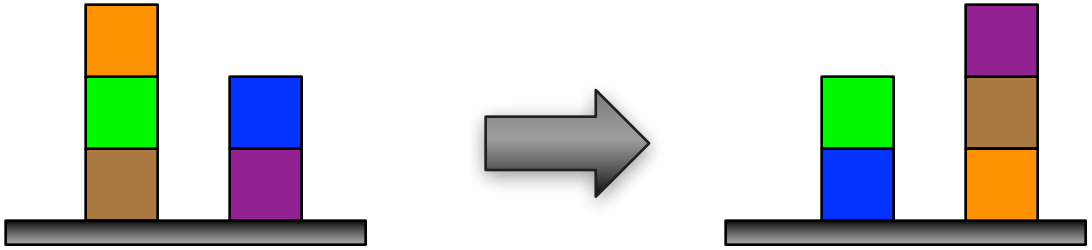
Basic Search Methods

© Gunnar Gotshalks

Block World

- Rearrange blocks in a block world

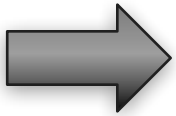


© Gunnar Gotshalks

Word Puzzle

- Rearrange letters into a particular order
  - **Tiles can move horizontally or vertically into the empty space**

L	A	T	E
Y	O	U	R
M	I	N	D
P	A	R	

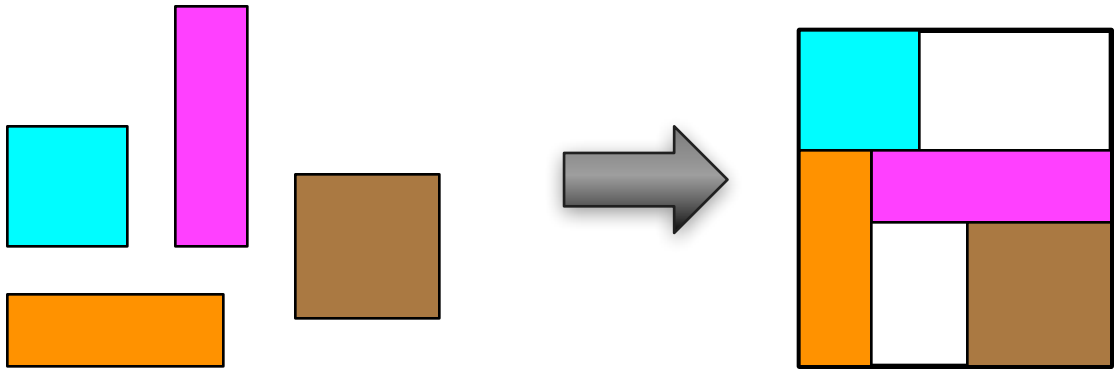


R	A	T	E
Y	O	U	R
M	I	N	D
P	A	L	

© Gunnar Gotshalks

Knapsack Packing

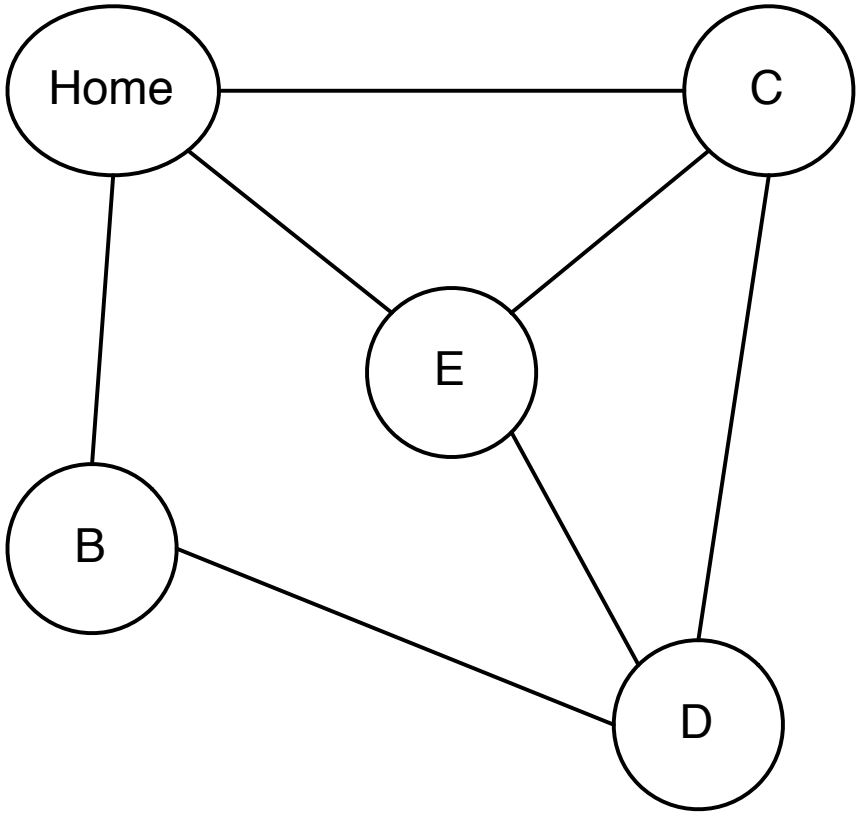
- Put items into a container
  - **Here, put rectangles into a box**



© Gunnar Gotshalks

Travelling Salesman

- Travel from home to each of the cities once and return home.



© Gunnar Gotshalks

Commonality

- When one is working on the previous problems what is common about the work?

© Gunnar Gotshalks

Commonality – 2

- When one is working on the previous problems what is common about the work?
  - Actions take place
    - Blocks are moved, one at a time

© Gunnar Gotshalks

Commonality – 3

- When one is working on the previous problems what is common about the work?
  - Actions take place
    - Blocks are moved, one at a time
    - Tiles are moved, one at a time

© Gunnar Gotshalks

Commonality – 4

- When one is working on the previous problems what is common about the work?
  - Actions take place
    - Blocks are moved, one at a time
    - Tiles are moved, one at a time
    - Rectangles move, one at a time

© Gunnar Gotshalks

Commonality – 5

- When one is working on the previous problems what is common about the work?
  - Actions take place
    - Blocks are moved, one at a time
    - Tiles are moved, one at a time
    - Rectangles move, one at a time
    - Salesman moves, one city at a time

© Gunnar Gotshalks

Commonality of actions

- When one is working on the previous problems what is common about the work?
  - Actions take place
    - Blocks are moved, one at a time
    - Tiles are moved, one at a time
    - Rectangles move, one at a time
    - Salesman moves, one city at a time
- What is a common property of the actions?

© Gunnar Gotshalks

Commonality of actions – 2

- What is a common property of the actions?
  - The problem situation changes
  - I.e. the state of the problem changes

© Gunnar Gotshalks

Solving a problem

- What do you do in solving a problem?

© Gunnar Gotshalks

Solving a problem – 2

- What do you do in solving a problem?
  - Given an initial state

© Gunnar Gotshalks

Solving a problem – 3

- What do you do in solving a problem?
  - Given an initial state
  - Given a goal state

© Gunnar Gotshalks

Solving a problem – 4

- What do you do in solving a problem?
  - Given an initial state
  - Given a goal state
  - Given a set of actions

© Gunnar Gotshalks

Solving a problem – 5

- What do you do in solving a problem?
  - Given an initial state
  - Given a goal state
  - Given a set of actions
  - Traverse from state to state, by applying actions

© Gunnar Gotshalks

Solving a problem – 6

- What do you do in solving a problem?
  - Given an initial state
  - Given a goal state
  - Given a set of actions
  - Traverse from state to state, by applying actions
    - As an action is done you can consider the new state as being the new initial state
      - You have a subproblem

© Gunnar Gotshalks

Solving a problem – 7

- What do you do in solving a problem?
  - Given an initial state
  - Given a goal state
  - Given a set of actions
  - Traverse from state to state, by applying actions
    - As an action is done you can consider the new state as being the new initial state
      - You have a sub-problem
  - Stop when a goal state is reached

© Gunnar Gotshalks

Programming a problem solution

- What do you need to do in creating a programming solution for a problem?

© Gunnar Gotshalks

Programming a problem solution – 2

- What do you need to do in creating a programming solution for a problem?
  - Get a representation for a state

© Gunnar Gotshalks

Programming a problem solution – 3

- What do you need to do in creating a programming solution for a problem?
  - Get a representation for a state

- What is a representation?

© Gunnar Gotshalks

Programming a problem solution – 4

- What do you need to do in creating a programming solution for a problem?
  - Get a representation for a state

- What is a representation?
  - The data structure

© Gunnar Gotshalks

Programming a problem solution – 5

- What do you need to do in creating a programming solution for a problem?
  - Get a representation for a state

- What is a representation?
    - The data structure

- What else do you need?

© Gunnar Gotshalks

Programming a problem solution – 6

- What do you need to do in creating a programming solution for a problem?
  - Get a representation for a state
    - The data structure
  - Get a representation for the actions

© Gunnar Gotshalks

Programming a problem solution – 7

- What do you need to do in creating a programming solution for a problem?
  - Get a representation for a state
    - The data structure
  - Get a representation for the actions
- What is the representation?

© Gunnar Gotshalks

Programming a problem solution – 8

- What do you need to do in creating a programming solution for a problem?
  - Get a representation for a state
    - The data structure
  - Get a representation for the actions

- What is the representation?
    - The routines

© Gunnar Gotshalks

Programming a problem solution – 9

- What do you need to do in creating a programming solution for a problem?
  - Get a representation for a state
    - The data structure
  - Get a representation for the actions
    - The routines

© Gunnar Gotshalks

Abstract state space

- What is the abstract structure of a state space?

© Gunnar Gotshalks

Abstract state space – 2

- What is the abstract structure of a state space?
  - A graph

© Gunnar Gotshalks

Abstract state space – 3

- What is the abstract structure of the state space?
  - A graph

- What are its vertices and edges?

© Gunnar Gotshalks

Abstract state space – 4

- What is the abstract structure of the state space?
  - A graph
- What are its vertices and edges?
  - Vertices are the states
  - Each edge is an action joining two states

© Gunnar Gotshalks

Abstract state space – 5

- What is the abstract structure of the state space?
  - A graph
- What are its vertices and edges?
  - Vertices are the states
  - Each edge is an action joining two states
- Is the graph directed or un-directed?

© Gunnar Gotshalks

Abstract state space – 6

- What is the abstract structure of the state space?
  - A graph
- What are its vertices and edges?
  - Vertices are the states
  - Each edge is an action joining two states
- Is the graph directed or un-directed?
  - Problem dependent
    - Undirected – Blocks, Word puzzle
    - Directed – Knapsack, travelling salesman

© Gunnar Gotshalks

State space solution

- In state space, what is a solution?

© Gunnar Gotshalks

State space solution – 2

- In state space, what is a solution?
  - A path from the start state to the goal state

- How do you find a path in the state space graph?

© Gunnar Gotshalks

State space solution – 3

- In state space, what is a solution?
  - A path from the start state to the goal state

- How do you find a path in the state space graph?
  - Explore the graph by trying different paths

© Gunnar Gotshalks

Path exploration

- During the exploration, how do you view the graph?

© Gunnar Gotshalks

Path exploration – 2

- During the exploration, how do you view the graph?
  - As a tree

- What is the problem?

© Gunnar Gotshalks

Path exploration – 3

- During the exploration, how do you view the graph?
  - As a tree

- What is the problem?
  - Graphs have loops

© Gunnar Gotshalks

Path exploration – 4

- During the exploration, how do you view the graph?
  - As a tree
- What is the problem?
  - Graphs have loops
  - Need to remove the loops

© Gunnar Gotshalks

Path exploration – 4

- During the exploration, how do you view the graph?
  - As a tree
- What is the problem?
  - Graphs have loops
  - Need to remove the loops
  - Implies checking that nodes do not repeat in the path

© Gunnar Gotshalks

Path exploration – 5

- During the exploration, how do you view the graph?
  - As a tree

- What is the problem?
  - Graphs have loops
  - Need to remove the loops
  - Implies checking that nodes do not repeat in the path
  - Logically breaking edges in the graph

© Gunnar Gotshalks

Tree searching

- The state space is logically searched as a tree.
  - We say the tree is traversed

- What are the two fundamental tree traversal methods?

© Gunnar Gotshalks

Tree searching – 2

- The state space is logically searched as a tree.
  - We say the tree is traversed
- What are the two fundamental tree traversal methods?
  - Depth first
  - Breadth first

© Gunnar Gotshalks

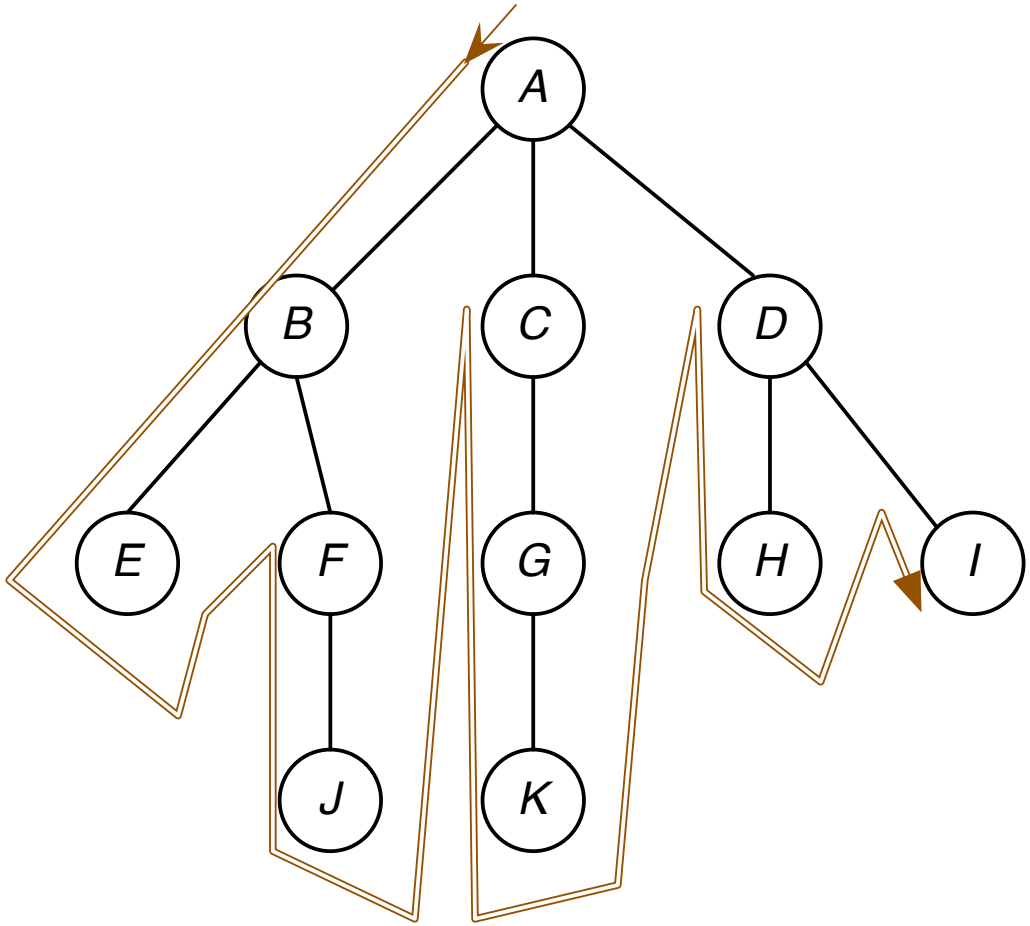
Tree searching – 3

- The state space is logically searched as a tree.
  - We say the tree is traversed

- What are the two fundamental tree traversal methods?
  - Depth first
  - Breadth first
  - Iterative-deepening ← Variation 1
  - Bidirectional ← Variation 2

© Gunnar Gotshalks

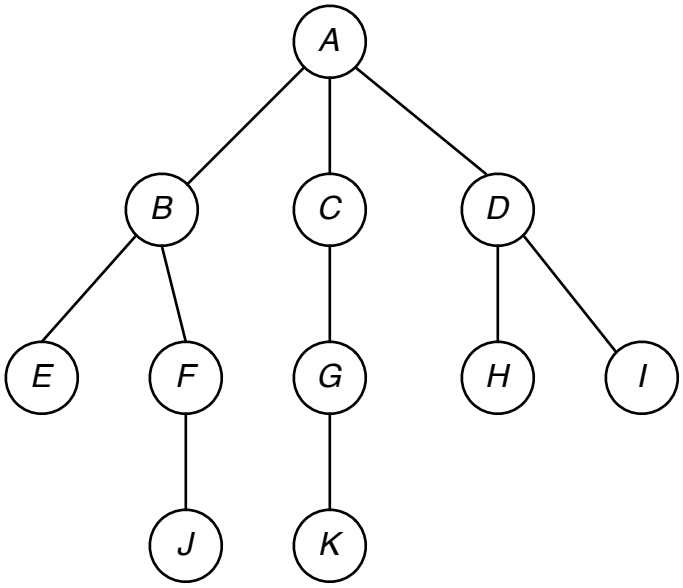
Depth-first search



© Gunnar Gotshalks

Depth-first search first 6 steps

- Candidate paths (accumulator) in a list
- Extend it with a successor at the front of the list



- **[A]** **Start**
- **[B, A]**
- **[E, B, A]**
- **[F, B, A]** **Backtrack**
- **[J, F, B, A]**
- **[C, A]** **Backtrack**
- **[G, C, A]**

Why backwards?
- **Each path.**
- **Candidate paths.**

© Gunnar Gotshalks

Depth-first search properties

- Find shortest solution? (Y, N)
- Time complexity
  - Consider B (branching factor) and Dmax (max depth of search)
    - Why not D?
- Space complexity
  - Consider B (branching factor) and Dmax (max depth of search)

© Gunnar Gotshalks

Depth-first search properties – 2

- Shortest solution not guaranteed

© Gunnar Gotshalks

Depth-first search properties – 3

- Shortest solution not guaranteed
- Infinite loops possible in cyclic graphs

© Gunnar Gotshalks

Depth-first search properties – 4

- Shortest solution not guaranteed
- Infinite loops possible in cyclic graphs
- Time complexity is $O(B^{Dmax})$
  - **On average have to follow half the paths up to the maximum depth**

© Gunnar Gotshalks

Depth-first search properties – 5

- Shortest solution not guaranteed
- Infinite loops possible in cyclic graphs
- Time complexity is $O(B^{Dmax})$
  - **On average have to follow half the paths up to the maximum depth**
- Space complexity is O(Dmax)
  - **Current path is most of the space, with linear overhead for backtracking**

© Gunnar Gotshalks

Depth-first problem

- What is the major problem with depth-first search?

© Gunnar Gotshalks

Depth-first problem – 2

- What is the major problem with depth-first search?
  - Potentially can go down an infinite path, miss the goal state

© Gunnar Gotshalks

Depth-first problem – 3

- What is the major problem with depth-first search?
  - Potentially can go down an infinite path, miss the goal state

- What can be done to prevent this?

© Gunnar Gotshalks

Depth-first problem – 4

- What is the major problem with depth-first search?
  - Potentially can go down an infinite path, miss the goal state

- What do can be done to prevent this?
  - Set a maximum depth

© Gunnar Gotshalks

Maximum depth problem

- What is the problem with setting a maximum depth of search?

© Gunnar Gotshalks

Maximum depth problem – 2

- What is the problem with setting a maximum depth of search?
  - Goal state may be deeper, never find it

© Gunnar Gotshalks

Maximum depth problem – 3

- What is the problem with setting a maximum depth of search?
  - Goal state may be deeper, never find it
- What can we do to overcome this problem?

© Gunnar Gotshalks

Maximum depth problem – 4

- What is the problem with setting a maximum depth of search?
  - Goal state may be deeper, never find it
- What can we do to overcome this problem?
  - Explore by increasing depth

© Gunnar Gotshalks

Maximum depth problem – 5

- What is the problem with setting a maximum depth of search?
  - Goal state may be deeper, never find it
- What can we do to overcome this problem?
  - Explore by increasing depth
  - Incremental deepening

© Gunnar Gotshalks

Iterative-deepening search

- Depth-first search done repetitively with increasing depth
  - **Why is this good?**

© Gunnar Gotshalks

Iterative-deepening search – 2

- Depth-first search done repetitively with increasing depth
  - **Why is this good?**
    - **Avoids unbounded descent on any path**

© Gunnar Gotshalks

Iterative-deepening search properties

- Find shortest solution? (Y, N)
- Time complexity
  - Consider B (branching factor) and D (depth of search)
- Space complexity
  - Consider B (branching factor) and D (depth of search)

© Gunnar Gotshalks

Iterative-deepening search properties – 2

- Shortest solution guaranteed

© Gunnar Gotshalks

Iterative-deepening search properties – 3

- Shortest solution guaranteed
- Infinite loops possible in cyclic graphs
  - **Still have to break cycles**

© Gunnar Gotshalks

Iterative-deepening search properties – 4

- Shortest solution guaranteed
- Infinite loops possible in cyclic graphs
  - **Still have to break cycles**
- Time complexity is $O(B^D)$
  - **Generate all nodes up to depth D**
    - **How many times is the successor relation used**

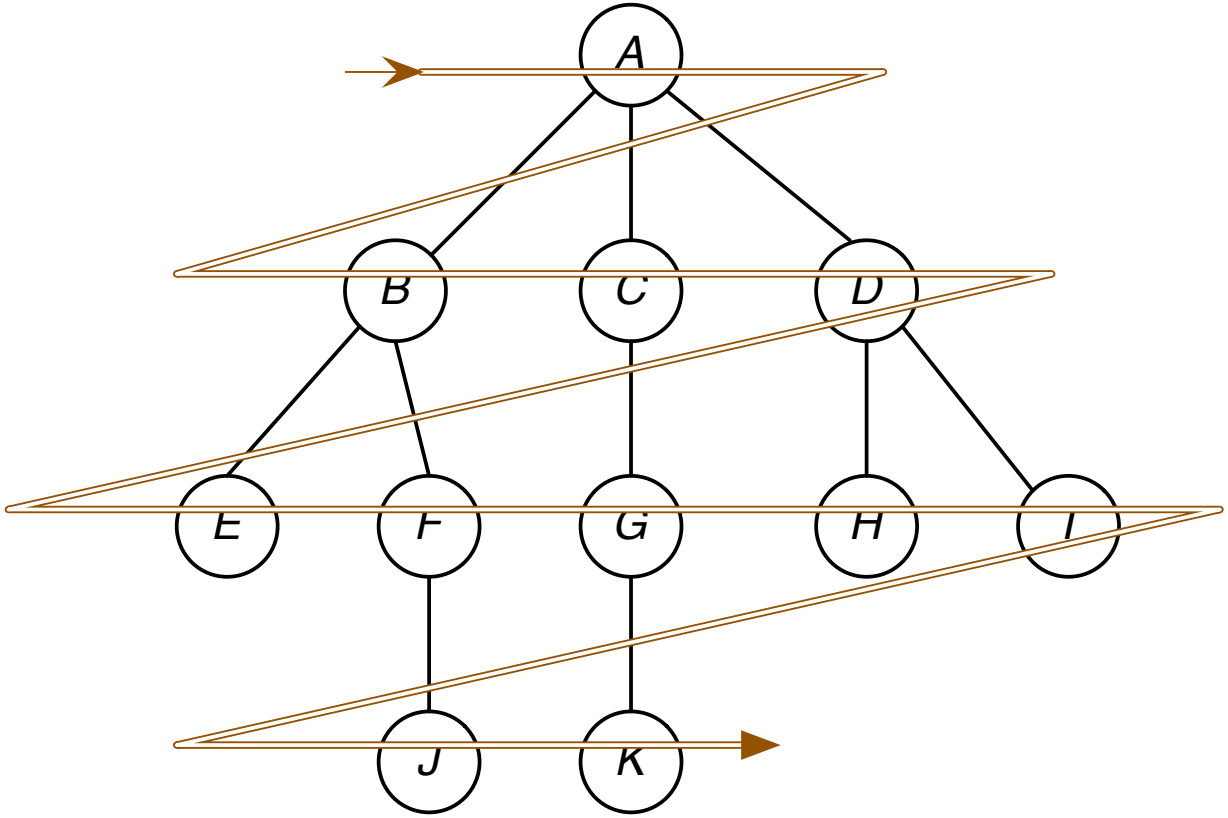
© Gunnar Gotshalks

Iterative-deepening search properties – 5

- Shortest solution guaranteed
- Infinite loops possible in cyclic graphs
  - **Still have to break cycles**
- Time complexity is $O(B^D)$
  - **Generate all nodes up to depth D**
    - **How many times is the successor relation used**
- Space complexity is O(D)
  - **Performs (D+1) depth-first searches**

© Gunnar Gotshalks

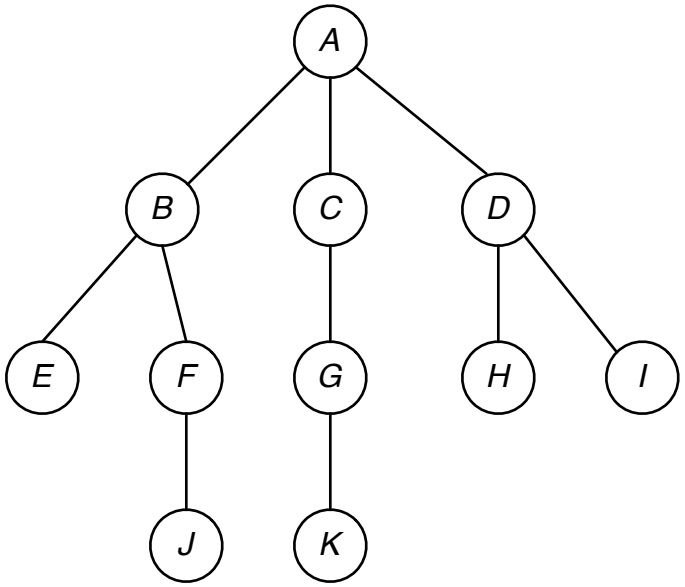
Breadth-first search



© Gunnar Gotshalks

Breadth-first search first two levels

- Candidate paths in a list
- Remove first path
  extend it
- Add extensions
  at the end of the list



- [[A]] Start
- [[B,A], [C,A], [D,A]]
- [[C,A], [D,A], [E,B,A], [F,B,A]]
- [[D,A] ,[E,B,A], [F,B,A], [G,C,A]]
- [[E,B,A], [F,B,A], [G,C,A], [H,D,A], [I,D,A]]

© Gunnar Gotshalks

Breadth-first search properties

- Finds shortest solution? (Y, N)
- Time complexity
  - Consider B (branching factor) and D (depth of search)
- Space complexity
  - Consider B (branching factor) and D (depth of search)

© Gunnar Gotshalks

Breadth-first search properties – 3

- Guaranteed to find the shortest path to a solution

© Gunnar Gotshalks

Breadth-first search properties – 4

- Guaranteed to find the shortest path to a solution
- Infinite loops possible in cyclic graphs
  - **May need to break cycles**

© Gunnar Gotshalks

Breadth-first search properties – 5

- Guaranteed to find the shortest path to a solution
- Infinite loops possible in cyclic graphs
  - **May need to break cycles**
- Time complexity is $O(B^D)$
  - **Have to explore all paths**

© Gunnar Gotshalks

Breadth-first search properties – 6

- Guaranteed to find the shortest path to a solution
- Infinite loops possible in cyclic graphs
  - **May need to break cycles**
- Time complexity is $O(B^D)$
  - **Have to explore all paths**
- Space complexity is $O(B^D)$
  - **Need to keep all paths to be able to lengthen them**

© Gunnar Gotshalks

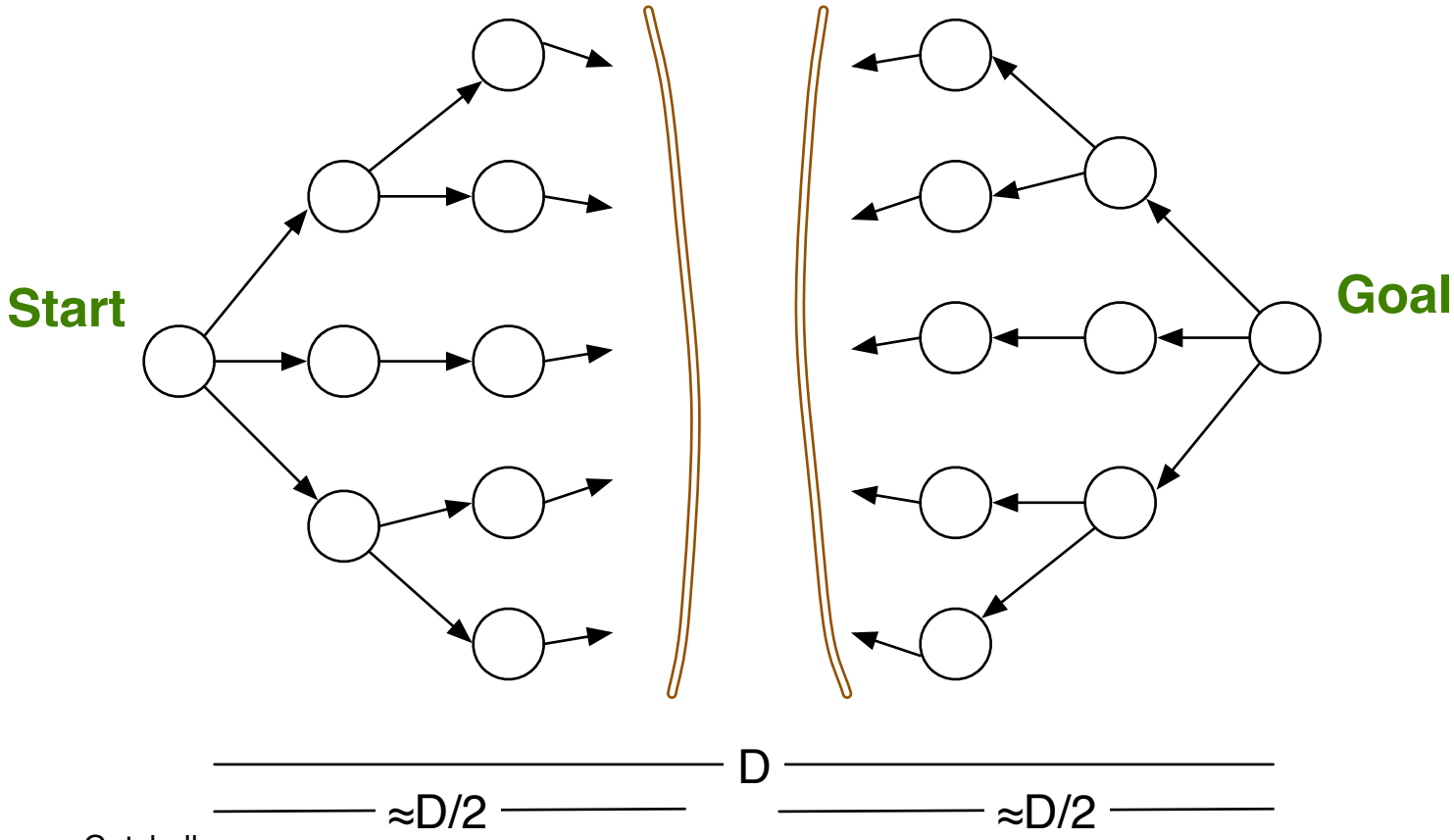
Bidirectional search

- **What is bidirectional search?**

© Gunnar Gotshalks

Bidirectional search – 2

- Do breadth-first search from the Start to the Goal
- Simultaneously do a breadth-first search from the Goal to the Start



© Gunnar Gotshalks

Bidirectional search properties

- Find shortest solution? (Y, N)
- Time complexity
  - Consider B (branching factor) and D (depth of search)
- Space complexity
  - Consider B (branching factor) and D (depth of search)

© Gunnar Gotshalks

Bidirectional search properties – 2

- Shortest solution guaranteed

© Gunnar Gotshalks

Bidirectional search properties – 3

- Shortest solution guaranteed
- Infinite loops possible in cyclic graphs
  - **May need to break cycles**

© Gunnar Gotshalks

Bidirectional search properties – 4

- Shortest solution guaranteed
- Infinite loops possible in cyclic graphs
  - **May need to break cycles**
- Time complexity is $O(B^{D/2})$
  - **About half of breadth-first depth is searched in each direction**

© Gunnar Gotshalks

Bidirectional search properties – 5

- Shortest solution guaranteed
- Infinite loops possible in cyclic graphs
- Time complexity is $O(B^{D/2})$
  - **About half of breadth-first depth is searched in each direction**
- Space complexity is $O(B^{D/2})$
  - **About half of breadth-first paths are kept in each direction**

© Gunnar Gotshalks

Bidirectional search conditions

- When can you use bidirectional search

© Gunnar Gotshalks

Bidirectional search conditions– 2

- When can you use bidirectional search
  - Need to know the goal state

© Gunnar Gotshalks

Bidirectional search conditions

- When can you use bidirectional search
  - Need to know the goal state
  - Need to be able to have inverse of successor function

© Gunnar Gotshalks

Bidirectional alias

- What alias do you know for bidirectional search?

© Gunnar Gotshalks

Bidirectional alias – 2

- What alias do you know for bidirectional search?
  - Combination of
    - Forward chaining

      and

    - Backward chaining

© Gunnar Gotshalks

Time & space complexity summary

- Breadth-first and iterative deepening guarantee shortest solution

© Gunnar Gotshalks

Time & space complexity summary – 2

- Breadth-first and iterative deepening guarantee shortest solution

- Breadth-first has high space complexity

© Gunnar Gotshalks

Time & space complexity summary – 3

- Breadth-first and iterative deepening guarantee shortest solution
- Breadth-first has high space complexity
- Depth-first has low space complexity
  May search far below goal state depth

© Gunnar Gotshalks

Time & space complexity summary – 4

- Breadth-first and iterative deepening guarantee shortest solution
- Breadth-first has high space complexity
- Depth-first has low space complexity
  May search far below goal state depth
- Iterative deepening has best performance in terms of orders of complexity

© Gunnar Gotshalks

Problems with basic search

- What are the problems with basic search?

© Gunnar Gotshalks

Problems with basic search – 2

- What are the problems with basic search?
  - Too simplistic

© Gunnar Gotshalks

Problems with basic search – 3

- What are the problems with basic search?
  - Too simplistic
  - Wastes time/resources exploring “obviously” poor paths.

© Gunnar Gotshalks

Problems with basic search – 4

- What are the problems with basic search?
  - Too simplistic
  - Wastes time/resources exploring “obviously” poor paths.

- What can we do about it?

© Gunnar Gotshalks

Problems with basic search – 5

- What are the problems with basic search?
  - Too simplistic
  - Wastes time/resources exploring “obviously” poor paths.

- What can we do about it?
  - Use information / knowledge of the state space to make the search more efficient

© Gunnar Gotshalks

Problems with basic search – 6

- What are the problems with basic search?
  - Too simplistic
  - Wastes time/resources exploring “obviously” poor paths.

- What can we do about it?
  - Use information / knowledge of the state space to make the search more efficient

  - Use heuristics to guide us

© Gunnar Gotshalks

Heuristic searches

- Best-first search – A* algorithm
- IDA* algorithm
  - **Iterative deepening A***
- RBFS algorithm
  - **Recursive Best First Search**
- Hill climbing, steepest descent, greedy search
  - **Special case of A* when successor with best F is retained; no backtracking**
- Beam search
  - **Special case of A* where only some limited number, W (beam width), of best evaluated open nodes are kept**

We will look at
- **A***
- **IDA***
- **RBFS**

© Gunnar Gotshalks