ROB 537:
Learning Based Control

Week 8, Lecture 1
Planning

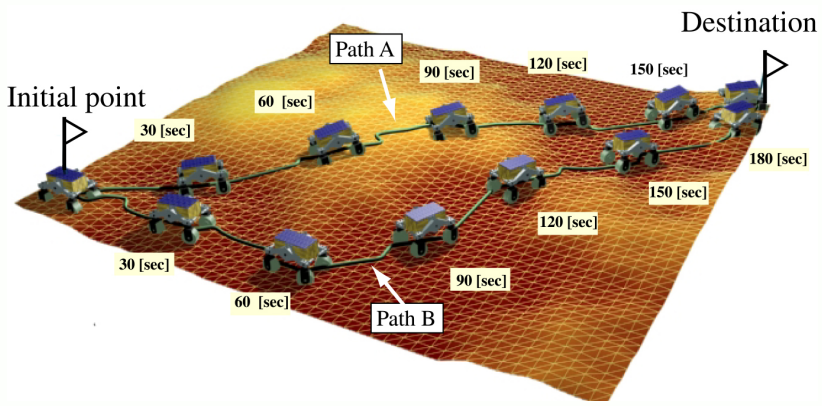
Announcements:

Preliminary Paper due TODAY
Presentations: Monday 11/27 at 8:15 AM, Rogers 226
Final papers: Due Monday 12/4 at 11:59 PM

No class on 11/22

Oregon State University

What is Planning?



Oregon State University

Planning

- Given start point S, end point G:
  - Find controls to take robot from S to G.
- What are challenges:
  - Obstacles?
  - Dynamics?
  - Feasibility?

Oregon State University

Planning

- Necessary when near-term choices of actions can enable, or prevent, later action choices required to achieve goals.
- Possible when agent possesses a sufficiently detailed and correct model of the environment, and of how actions affect the environment.
- Challenging because the space of possible plans grows exponentially with the plan duration.

Oregon State University

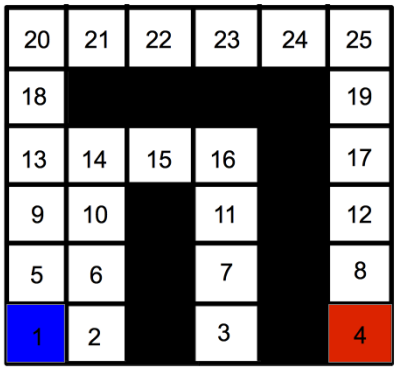
Planning

- Path planning: detailing a motion task into discrete motions
  - Robot must get from starting location to defined waypoint
  - Robot must not run into walls, fall down stairs, etc.
  - Ideally, path taken is most efficient (shortest) path
- Inputs to algorithm:
  - Description of task (starting and ending locations)
  - Constraints (robot's physical limitations, environmental obstacles, etc.)
  - Uncertainty about robot and environment
- Outputs of algorithm:
  - Speed and turning commands sent to robot's wheels at each time step

Oregon State University

Simple Planning Example

- We discretize an environment into a grid
  - The robot starts at the blue square (cell 1)
  - Want the robot to move to the red square (cell 4)
- How do we plan a path from cell 1 to cell 4?



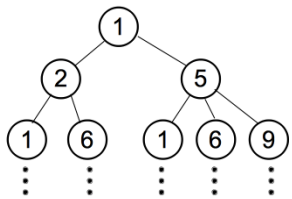
Oregon State University

Simple Planning Example

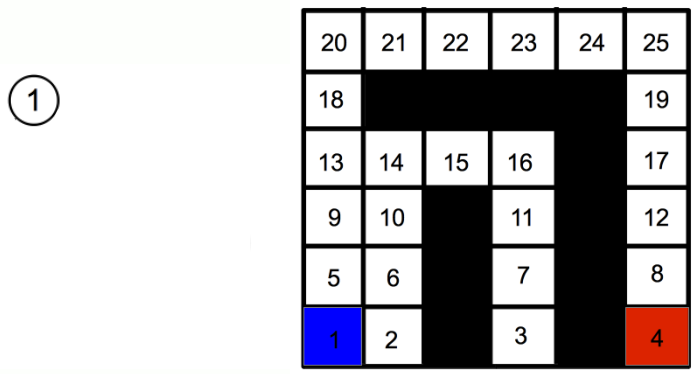
- We construct a "tree" through which we can search for optimal paths through the environment
- Each node of the tree corresponds to a position in the grid as well as the path taken to get to that position

Search Tree (Planning Tree)

- A search tree:
  - This is a "what if" tree of plans and outcomes
  - Start state at the root node
  - Children correspond to successors
  - Nodes contain states, correspond to PLANS to those states
  - For most problems, we can never actually build the whole tree



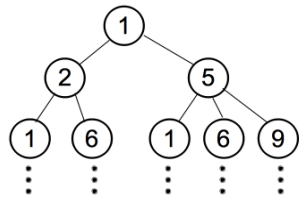
Search Tree for Planning



Oregon State University

Search Tree Basics

- A tree of depth *n* gives us all possible paths of length *n* or less which the robot can take
- Traverse the tree to find optimal path
- If we know that the optimal path is of length *n* or less, then the tree of depth *n* contains the optimal path for the robot
- How do we find the optimal path in this tree?
  - Search algorithms



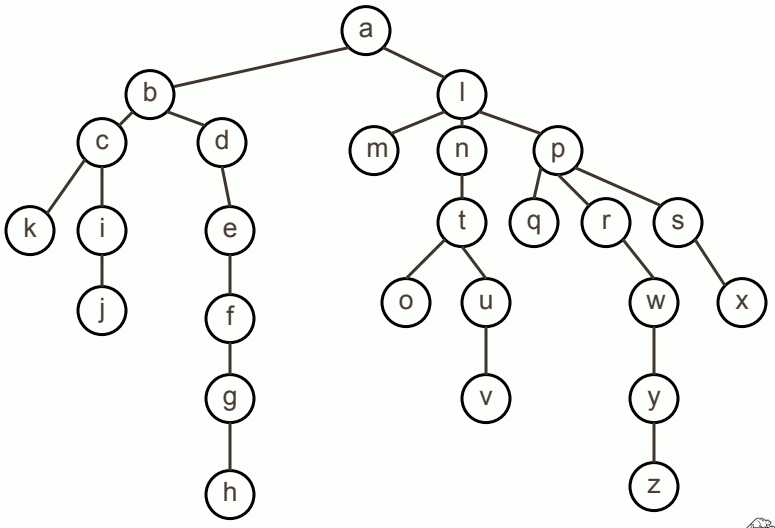
Oregon State University

Basics of Search

- Expand out possible plans
- Maintain a fringe of unexpanded plans
- Try to expand as few tree nodes as possible

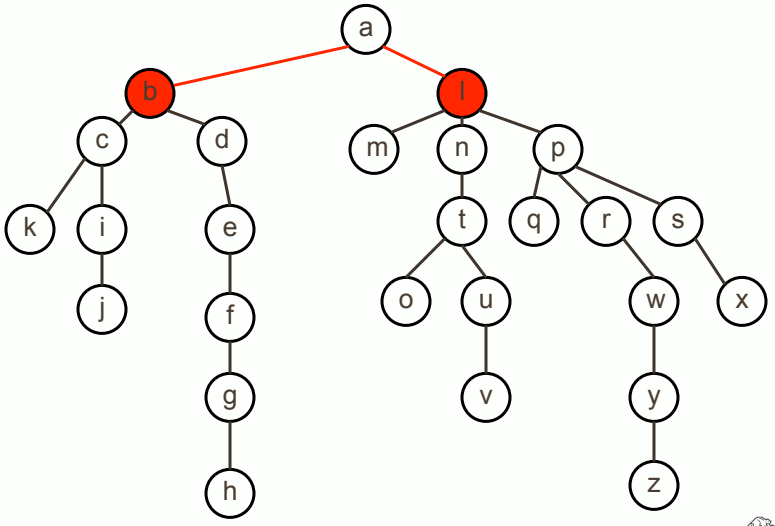
Oregon State University

Tree Search Example



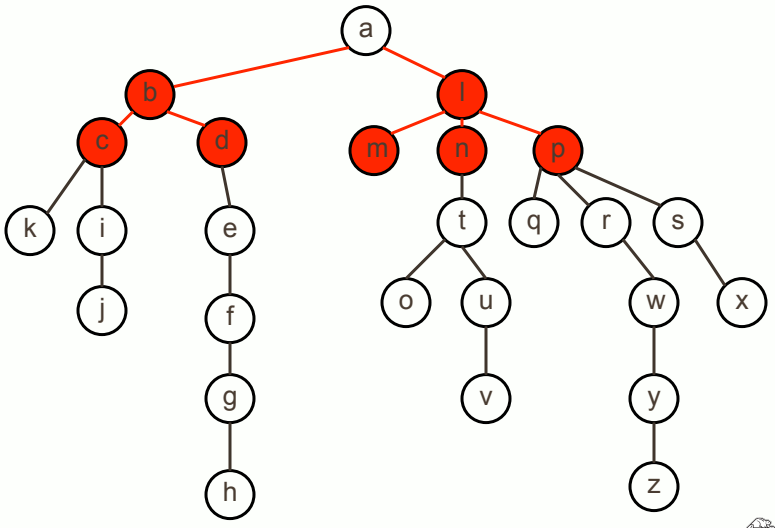
Oregon State University

Breadth First Search



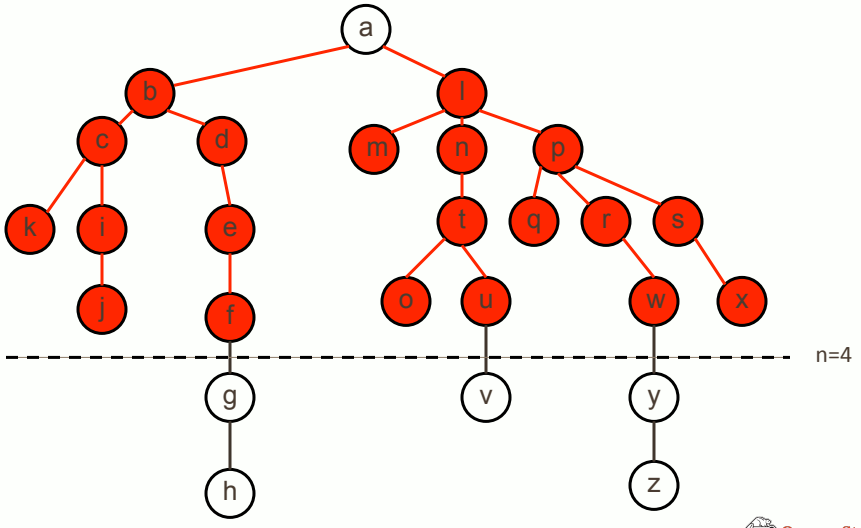
Oregon State University

Breadth First Search



Oregon State University

Breadth First Search



Oregon State University

Breadth First Search

- Breadth First Search (BFS) is one of the simplest search algorithms
- Two main operations in BFS:
  - Visit and inspect a node of a graph
  - Gain access to visit the nodes that neighbor the currently visited node
- BFS begins at a root node and inspects all neighboring nodes.
- For each neighbor node in turn, BFS inspects the neighbor nodes which were unvisited, and so on.
- BFS is guaranteed to find the optimal path through the search tree
  - First solution found is the optimal path
- Unfortunately, BFS is computationally inefficient

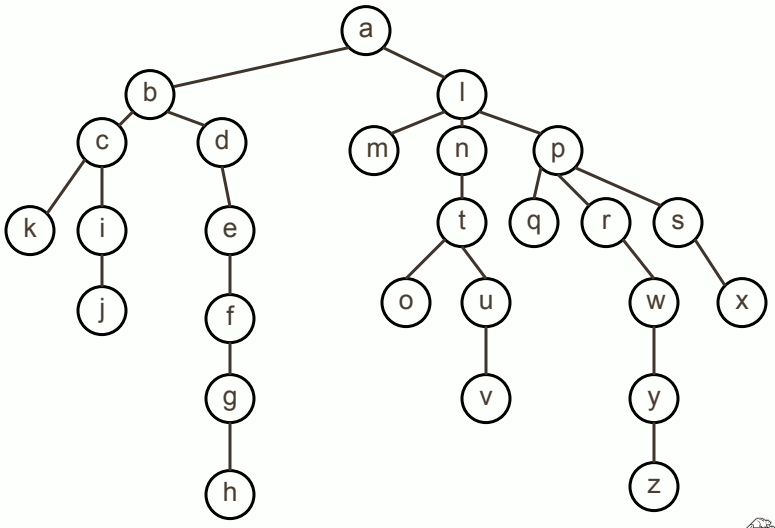
Oregon State University

Breadth First Search Overview

- Time complexity of O(V+E)
  - V is the number of vertices
  - E is the number of edges
  - Worst case is that each node is visited: computationally expensive!
- Consider another approach: Depth first search

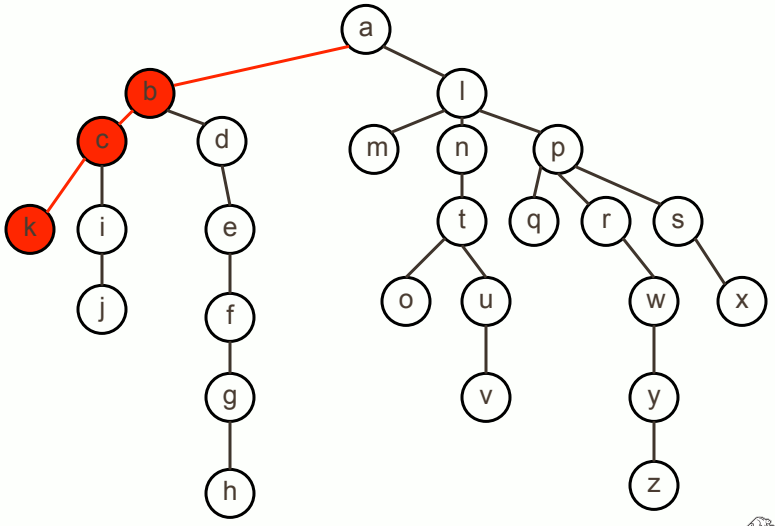
Oregon State University

Depth First Search



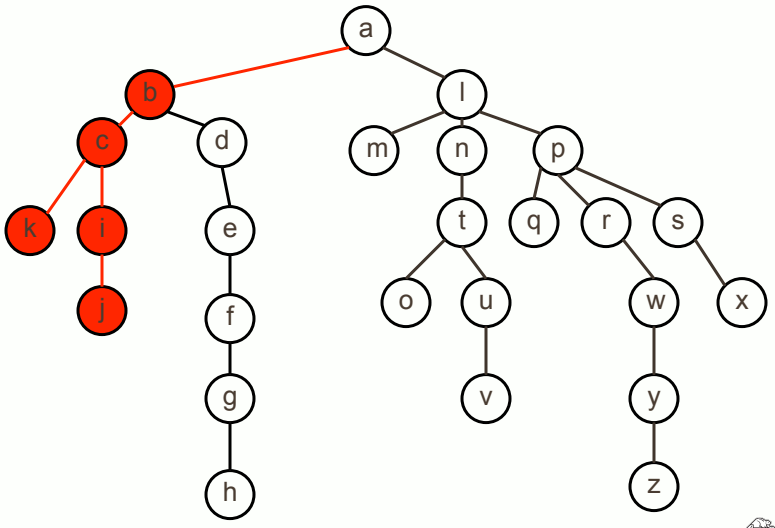
Oregon State University

Depth First Search



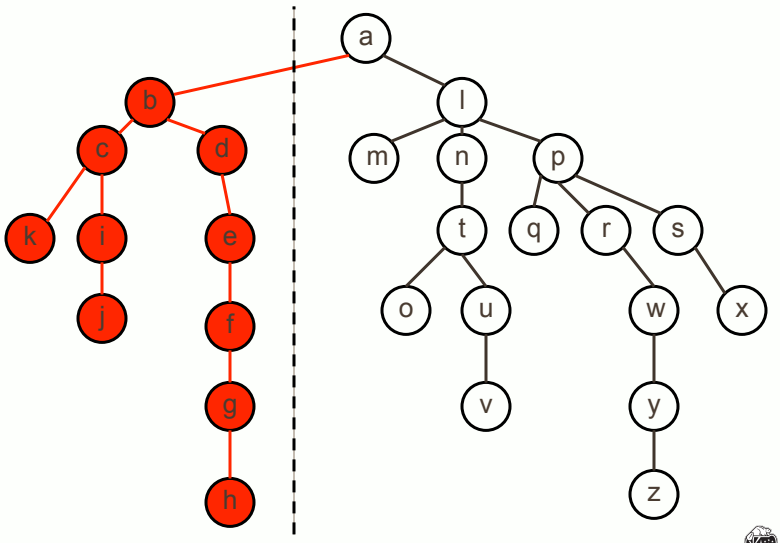
Oregon State University

Depth First Search



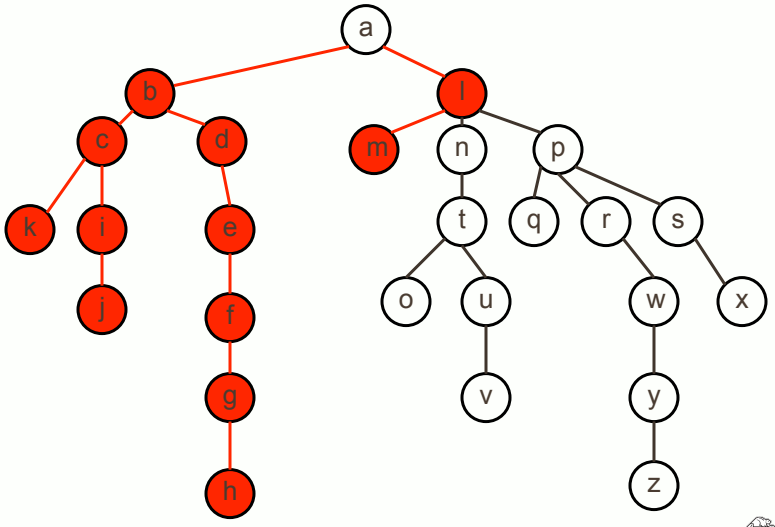
Oregon State University

Depth First Search



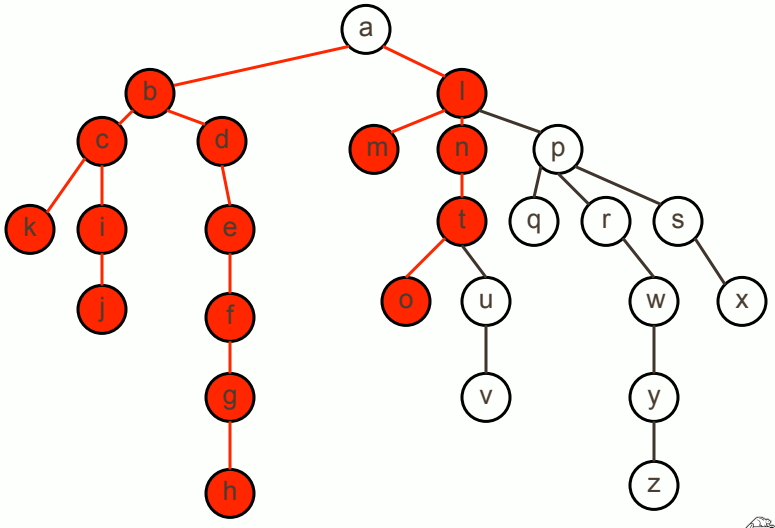
Oregon State University

Depth First Search



Oregon State University

Depth First Search



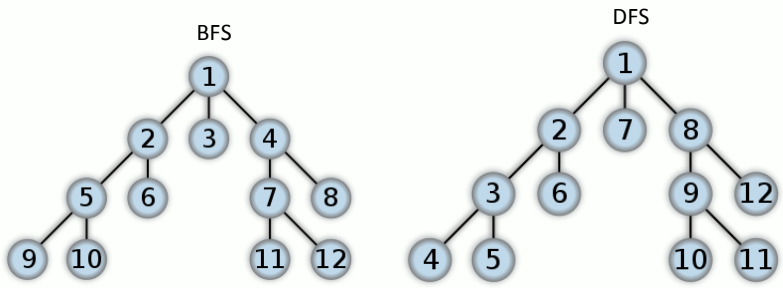
Oregon State University

Depth First Search

- Depth First Search (DFS) is a search algorithm where the search starts at the root node and explores as far as possible along each branch before backtracking
- Like BFS, DFS is guaranteed to find the optimal path in the tree
- Problem with DFS: graph is often too large to visit in its entirety, and DFS may not terminate when path lengths are infinite (looping through grid)
  - Therefore we typically only search to limited depth

Oregon State University

BFS vs. DFS



- Key difference between BFS and DFS is the order in which nodes are visited:
  - BFS searches the tree one **layer** at a time
  - DFS searches the tree one **path** at a time

Oregon State University

BFS vs. DFS

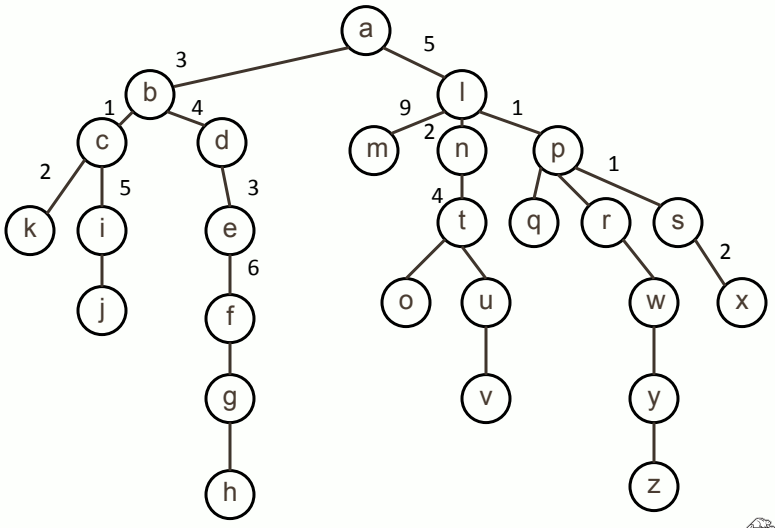
- Time bounds of DFS and BFS are equivalent: in the worst case, each algorithm searches the entire tree
- Space bounds of DFS are better than BFS: DFS takes less memory than BFS, since it isn't necessary to store all children at each level of the tree search
- Which algorithm is "better" is more dependent on the domain than computational complexity of the algorithms:
  - If solution is higher up the tree (closer to the root node), then BFS will perform better
  - If solution is farther down the tree (farther from the root node), then DFS will perform better
- Both BFS and DFS are simple to implement, however are computationally inefficient. More complex algorithms are faster, but more difficult to implement

Oregon State University

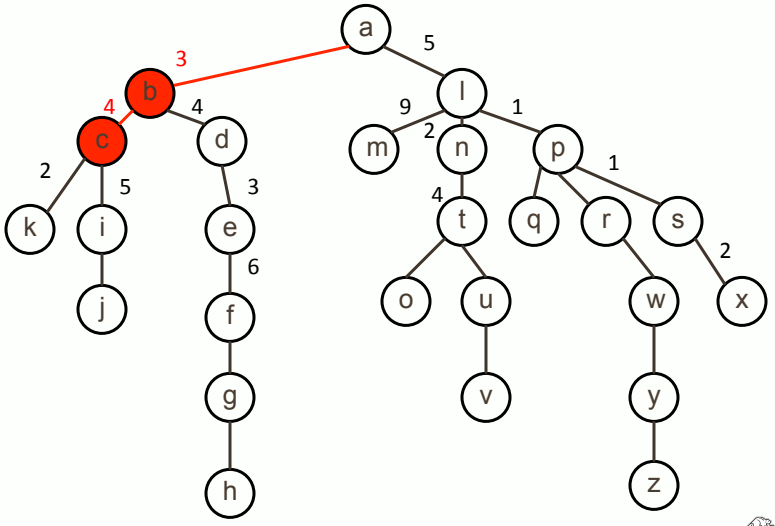
Costs on Actions

- Both BFS and DFS find the shortest path in terms of number of transitions.
  - What about costs?
- Uniform Cost Search

Tree Search Example

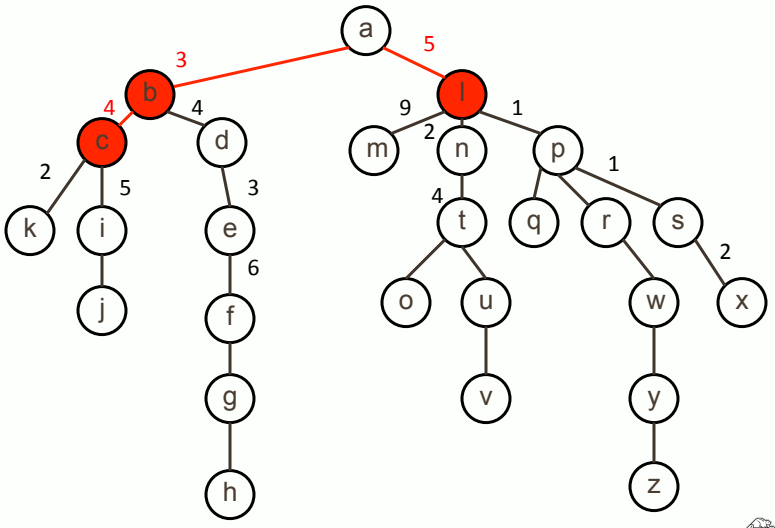


Uniform Cost Search



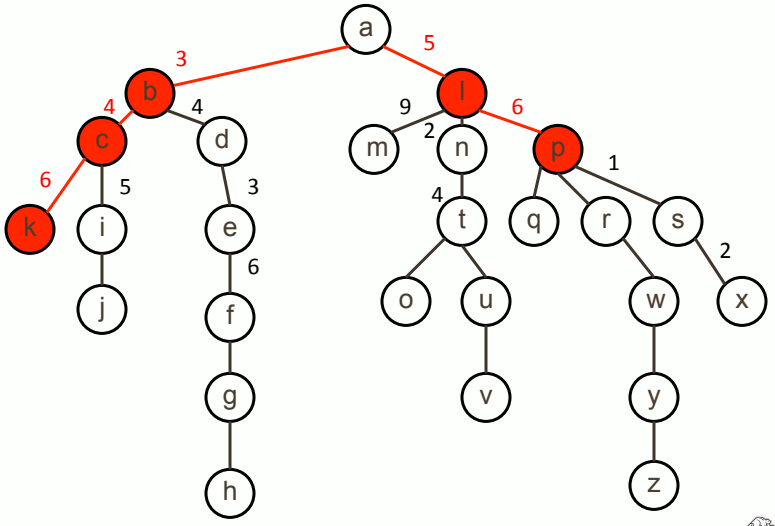
Oregon State University

Uniform Cost Search



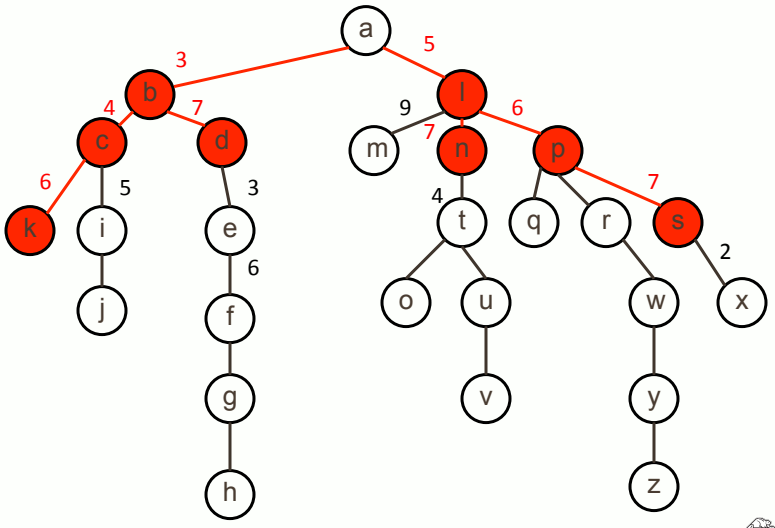
Oregon State University

Uniform Cost Search



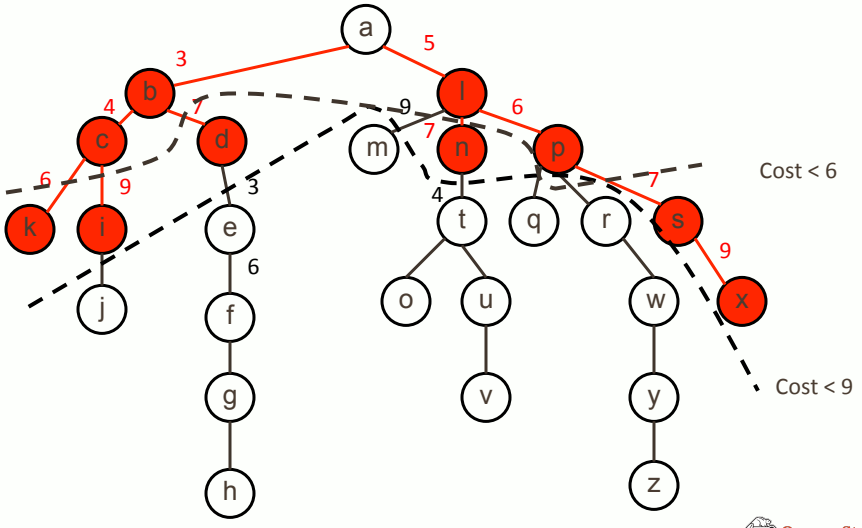
Oregon State University

Uniform Cost Search



Oregon State University

Uniform Cost Search



Oregon State University

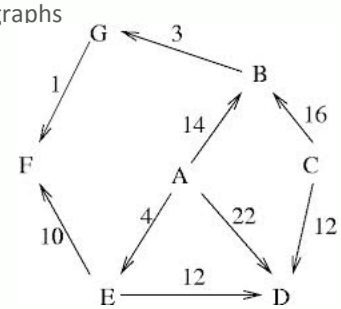
Uniform Cost Search

- Explore along cost "contour"
- UCS will find optimal path
- Problem: Search isn't directional
  - Searches in every direction
  - What about information about the goal state?

Oregon State University

Dijkstra's Algorithm

- Developed by Edsger Dijkstra in 1959
- Solves the single-source shortest path problem, producing a shortest path tree
- One of the most commonly used routing algorithms in graph traversal problems
- Asymptotically the fastest known single-source shortest path algorithm for arbitrary directed graphs



Oregon State University

Dijkstra's Algorithm Overview

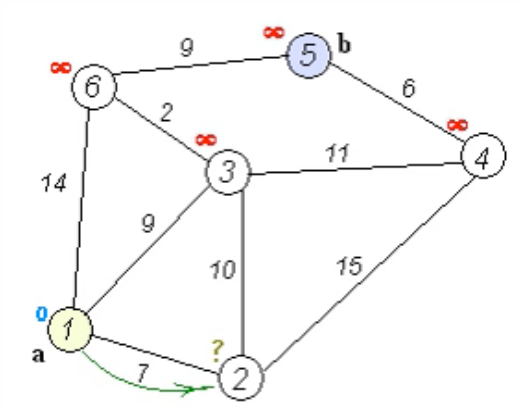
- Example: want to find shortest distance between two points in a city
- Initially, mark the distance to every intersection on the map with infinity

- Select an intersection: update the distance to every unvisited intersection that is directly connected to it: If this distance is less than previously recorded distance, update distance of unvisited intersection.
- Once each neighbor is evaluated, mark the current intersection as visited
- Select intersection with smallest distance to starting point

- Continue this process until the destination is reached. Once the destination is reached, follow the path cost tree in reverse to obtain shortest path.

Oregon State University

Dijkstra's Algorithm Overview



Courtesy wikipedia (http://en.wikipedia.org/wiki/Dijkstra%27s_algorithm)

Heuristic Search

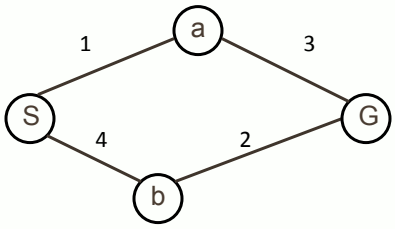
- Heuristic:
  - Any *estimate* of how close a state is to a goal
  - Designed for a particular search problem
  - Examples: Manhattan distance, Euclidean distance
- Combine UCS and a heuristic
- A* : Cost is f(n) = g(n) + h(n)

A* Search Algorithm

- A* is an extension of Dijkstra's algorithm, and achieves faster performance by using heuristics
- Uses best-first search: A* traverses a graph following a path of lowest expected total cost or distance
- Uses a knowledge plus heuristic cost function to determine the order in which the search visits nodes in the tree. The cost function is a sum of two functions:
  - Past path-cost function, which is known distance from the starting node to the current node
  - Future path-cost function, which is a "heuristic estimate" of the distance from the current node to the goal
- A* is equivalent to running Dijkstra's algorithm with reduced computational cost if certain assumptions about the heuristic are met:
  - the estimate of the future path-cost must not overestimate the distance to the goal

Oregon State University

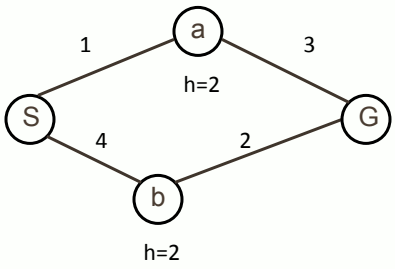
A* Algorithm



- Start node S
- Goal node G
- What is best path?

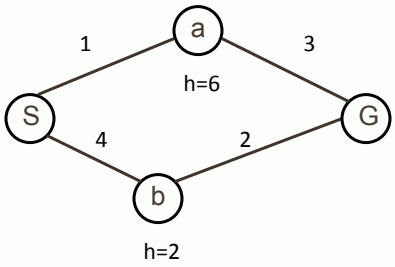
Oregon State University

A* Algorithm



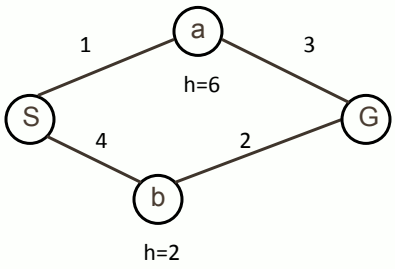
- From S:
  - Cost of a is 3= 1+2 (heuristic)
  - Cost of b is 6= 4+2
- Use a to reach goal
  - Total cost turns out to be 4
- Done!
  - Cost from b was 4 + 2 =6

A* Algorithm



- From S:
  - Cost of a is 7= 1+6 (heuristic)
  - Cost of b is 6= 4+2
- Use b to reach goal
  - Total cost turns out to be 6
- Done!
  - Cost from b was 4 + 2 =6
- Wait! What happened?

Optimality of A*



- Heuristic overestimated cost
- Key:
  - Heuristic must NOT overestimate actual cost

Oregon State University

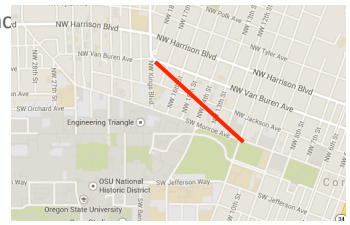
Admissible Heuristic

- A heuristic *h* is admissible if:

$$h(n) \leq h^*(n)$$

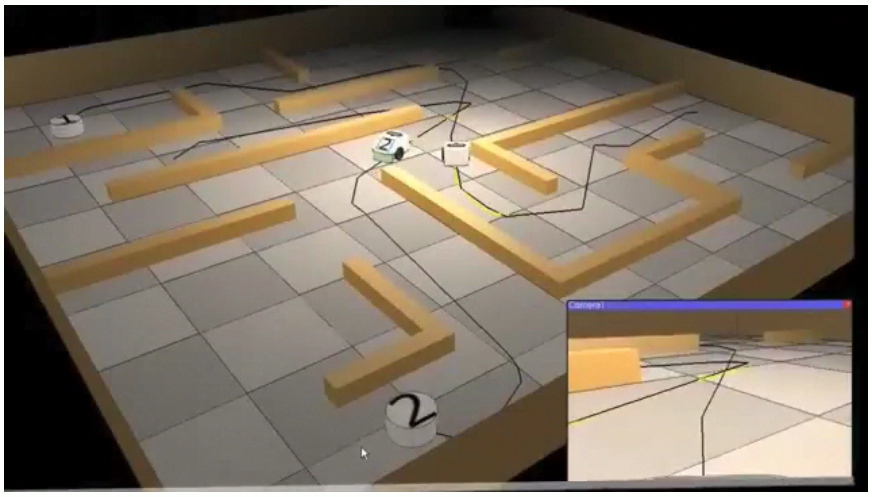
If h*(n) is true cost to goal

- Example:
  - On maps: direct line: guaranteed to not overestimate distance
  - On grid: Manhattan distanc



Oregon State University

Planning Example



Oregon State University

Difficulties in Planning

- Dynamic environments
- Uncertainty

When the environment changes or we don't know everything about the robot/environment, planned paths may need to be altered.

Uncertainty always complicates robotics!

Let's look at these problems in detail

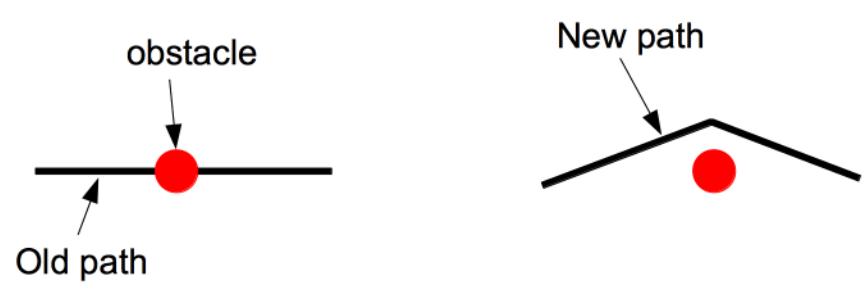
Oregon State University

Planning in Non-Static Environments

- What if the environment isn't static?
  - People (obstacles) moving around in the world
  - Doors opening and closing (paths which used to exist may no longer exist, and vice versa)
  - In disaster exploration missions, parts of buildings may collapse, closing off previously existing paths
- How do we account for environment changing after we have already found an optimal path?
- Two main ways to deal with dynamic maps:
  - Keep original plan, and deviate from plan as little as possible
  - Replan a new path based on new environmental conditions

Oregon State University

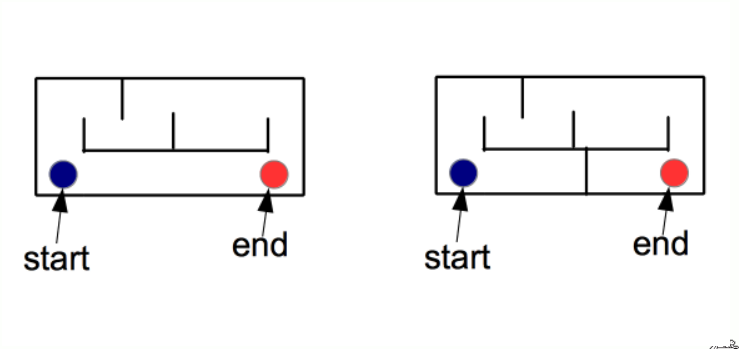
Keep Original Plan with Small Deviation



Oregon State University

Keep Original Path With Small Deviation

- This works fine if a person gets in the way and the robot simply needs to move around them
- What if a door is closed, or part of a hallway collapses?
  - The "small deviation" in the path is no longer well defined



Oregon State University

Replan a New Map

- If the environment changes, construct a new search tree and re-run the search algorithm to find the new best path based on changing environmental conditions
- What are the problems with this approach?
  - Search algorithms are slow, often difficult to compute in real-time
  - If we have to replan many times, this could slow down the mission substantially
- Path replanning algorithms exist, such as Delayed D*, to deal with these types of problems

Oregon State University

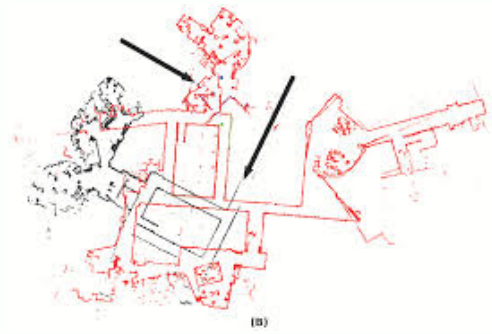
Which to use?

- So, do we deviate slightly from paths or plan new paths?
  - There is no canonical answer to this question.
  - It depends on nature of how the environment changes
- As a general rule of thumb:
  - If dynamics are minor and we can get away with small deviation, do it
  - If major change in environment occurs and it isn't clear what a "small deviation" means, then replan
- Dynamic environments are very difficult to deal with
  - If environment is static, then we can easily find and execute a solution
  - If environment is dynamic, then we often have to adjust solution during execution to deal with new environmental conditions

Oregon State University

Planning with Uncertainty

- There are many types of uncertainty in planning
  - Motion uncertainty
  - Missing information about environment
- How do these different types of uncertainty affect planning?



Oregon State University

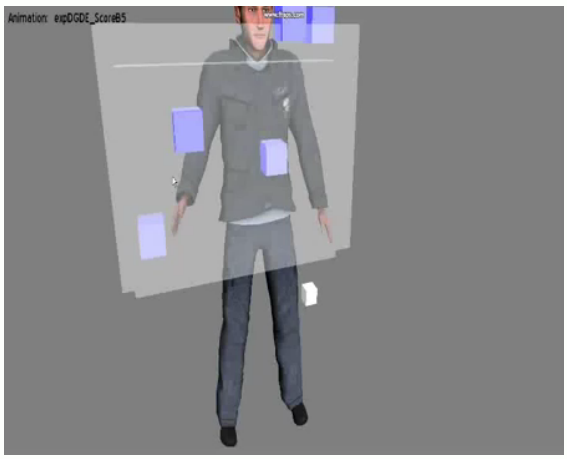
Missing Information About Environment

- Very commonly, we do not have all the information about an environment
  - A robot exploring a building after an earthquake to assess damage
  - Robot exploring a previously unknown environment, such as Mars
- Plan based on the limited knowledge; modify during execution
  - As more information about the environment is discovered, replan a path
- Missing and incorrect information has similar effects when compared to dynamic environments. An unexpected aspect of the environment is discovered, and the path is changed to account for this new information.

Oregon State University

Other Applications

- Path planning is not limited to a mobile robot moving around



Oregon State University

Hierarchical Planning

- A plan-space planning approach:
  - Incrementally creates a plan by refining more abstract plan steps, expanding them into more detailed subplans.
  - Exploits knowledge captured in the form of a library of subplans: rather than constructing a plan directly as a search through primitive executable actions, retrieve and combine subplans that have been prebuilt to achieve typical (sub)goals.
  - The planning process is complete when the plan is refined down to the level of executable actions.

Oregon State University

Summary

- Path planning: given an environment, a starting location, and an ending location, find the optimal path
- Process:
  - Enumerate all possible plans in a tree
  - Search through the tree to find the best path
- Algorithms: BFS, DFS, Dijkstra's algorithm, and A* search
- Many other search algorithms exist: B*, Best-First, Recursive Best First, D*, Depth-limited, IDA*, Hill-climbing, evolutionary algorithms, …
- When it isn't possible to follow previously planned path:
  - Take a slight deviation around obstacle and return to planned path, or
  - Plan a new path

Oregon State University