CSCI 5582
Artificial Intelligence

Lecture 3

Jim Martin

Today: 9/5

- Achieving goals as searching
- Some simple uninformed algorithms
- Issues and analysis
- Better uninformed methods

Review

- What's a goal-based agent?

Goal-based Agents

- What should a goal-based agent do when none of the actions it can currently perform results in a goal state?
- Choose an action that at least leads to a state that is closer to a goal than the current one is.

Goal-based Agents

Making that work can be tricky:

- What if one or more of the choices you make turn out not to lead to a goal?
- What if you're concerned with the best way to achieve some goal?
- What if you're under some kind of resource constraint?

Problem Solving as Search

One way to address these issues in a uniform framework is to view goal-attainment as problem solving, and viewing that as a search through the space of possible solutions.

Problem Solving

A problem is characterized as:

- An initial state
- A set of actions (functions that map states to other states)
- A goal test
- A cost function (optional)

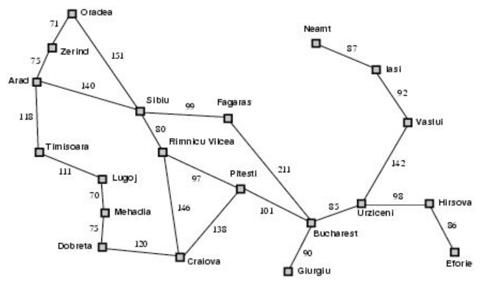
What is a Solution?

- A sequence of actions that when performed will transform the initial state into a goal state
  - Or sometimes just the goal state itself

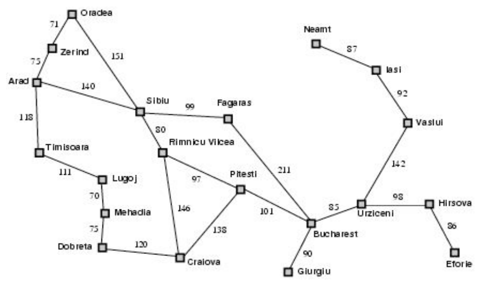
Framework

- We’re going to cover three kinds of search in the next few weeks:
  - Backtracking state-space search
  - Optimization search
  - Constraint-based search

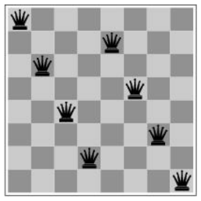
Backtracking State-Space Search



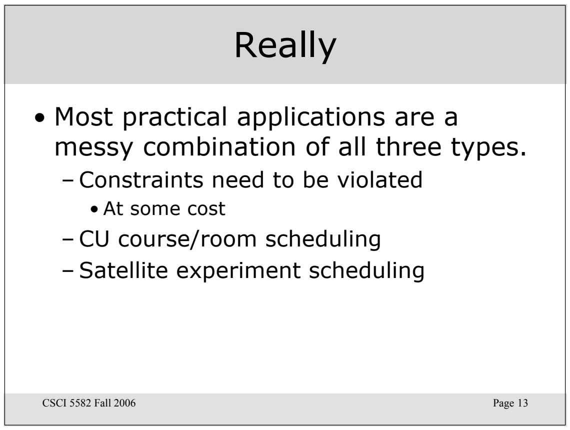
Optimization Search



Constraint Satisfaction Search

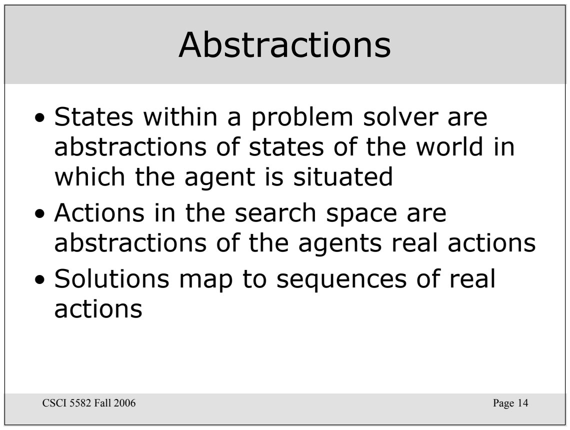


- Place N queens down on a chess board such that
  - No queen attacks any other queen
  - The goal state is the answer (the solution)
  - The action sequence is irrelevant



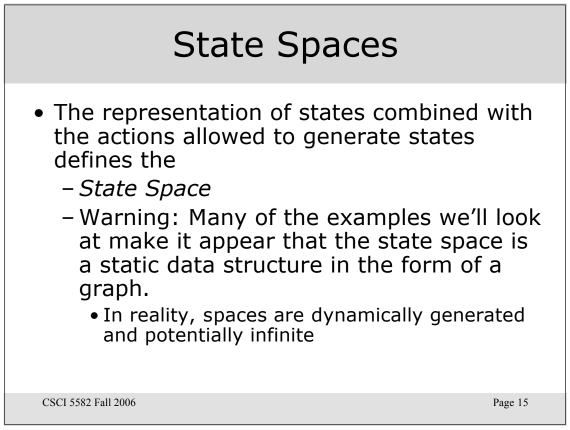
Really

- Most practical applications are a messy combination of all three types.
  - Constraints need to be violated
    - At some cost
  - CU course/room scheduling
  - Satellite experiment scheduling



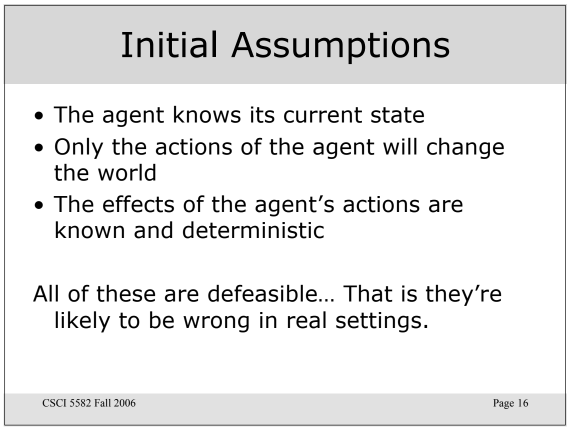
Abstractions

- States within a problem solver are abstractions of states of the world in which the agent is situated
- Actions in the search space are abstractions of the agents real actions
- Solutions map to sequences of real actions



State Spaces

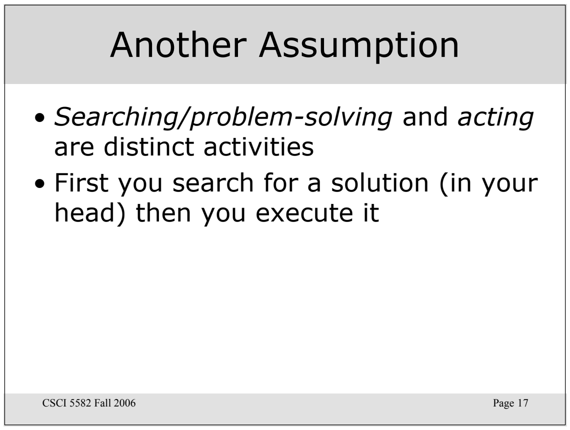
- The representation of states combined with the actions allowed to generate states defines the
  - *State Space*
  - Warning: Many of the examples we’ll look at make it appear that the state space is a static data structure in the form of a graph.
    - In reality, spaces are dynamically generated and potentially infinite



Initial Assumptions

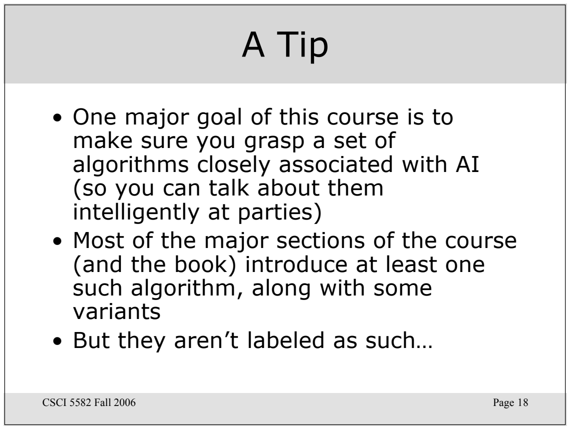
- The agent knows its current state
- Only the actions of the agent will change the world
- The effects of the agent’s actions are known and deterministic

All of these are defeasible… That is they’re likely to be wrong in real settings.



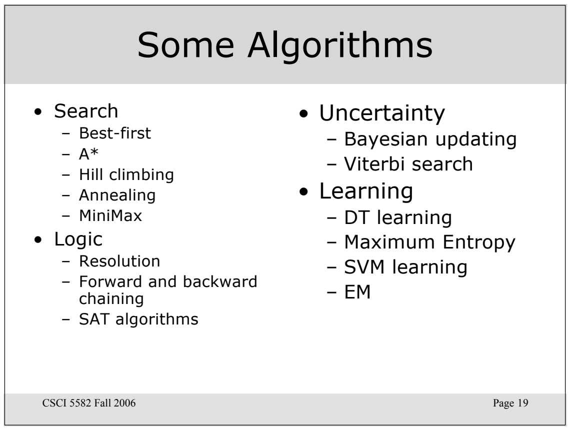
Another Assumption

- *Searching/problem-solving* and *acting* are distinct activities
- First you search for a solution (in your head) then you execute it



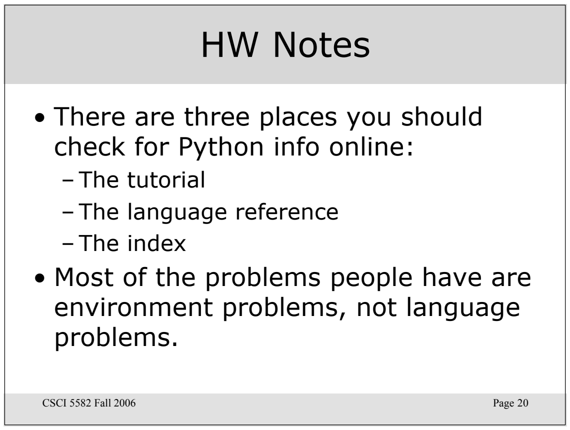
A Tip

- One major goal of this course is to make sure you grasp a set of algorithms closely associated with AI (so you can talk about them intelligently at parties)
- Most of the major sections of the course (and the book) introduce at least one such algorithm, along with some variants
- But they aren’t labeled as such…



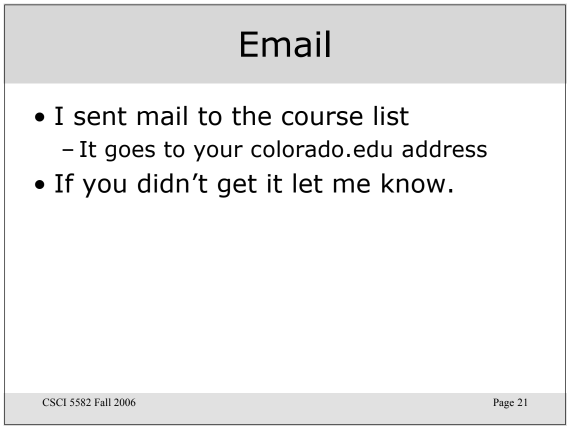
Some Algorithms

- Search
  - Best-first
  - A*
  - Hill climbing
  - Annealing
  - MiniMax
- Logic
  - Resolution
  - Forward and backward chaining
  - SAT algorithms
- Uncertainty
  - Bayesian updating
  - Viterbi search
- Learning
  - DT learning
  - Maximum Entropy
  - SVM learning
  - EM



HW Notes

- There are three places you should check for Python info online:
  - The tutorial
  - The language reference
  - The index
- Most of the problems people have are environment problems, not language problems.



Email

- I sent mail to the course list
  - It goes to your colorado.edu address
- If you didn’t get it let me know.

CAETE Students

- Hardcopy is not required for remote CAETE students
- Participation points will be based on email/phone communication
- Assignments/Quizzes are due 1 week after the in-class due date

Generalized (Tree) Search

Start by adding the initial state to an Agenda

Loop

```
  If there are no states left then fail
  Otherwise choose a state to examine
  If it is a goal state return it
  Otherwise expand it and add the resulting states to the agenda
```

Uninformed Techniques

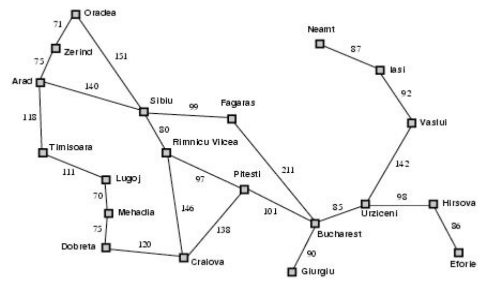
- Breadth First Search
- Uniform Cost Search
- Depth First Search

- Depth-limiting searches

Differences

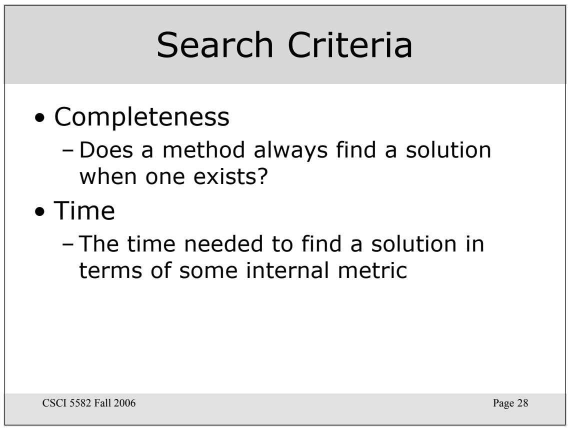
- The only difference among BFS, DFS, and Uniform Cost searches is in the the management of the agenda
  - The method for inserting elements into a queue
  - But the method has huge implications in terms of performance

Example Problem



Example Problem

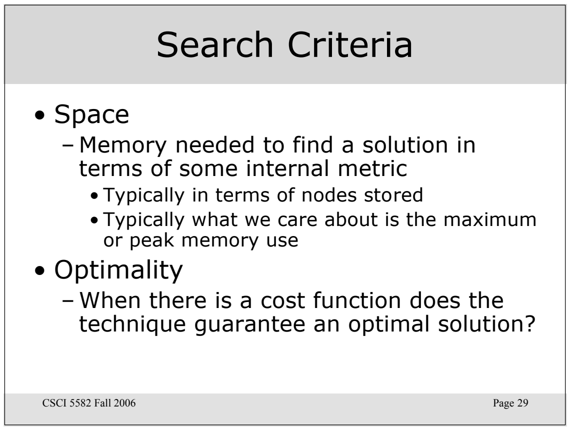
- You’re in Arad (initial state)
- You want to be in Bucharest (goal)
- You can drive to adjacent cities (actions)
- Sequence of cities is the solution (where Arad is the first and Bucharest is the last)



Search Criteria

- Completeness
  - Does a method always find a solution when one exists?
- Time
  - The time needed to find a solution in terms of some internal metric

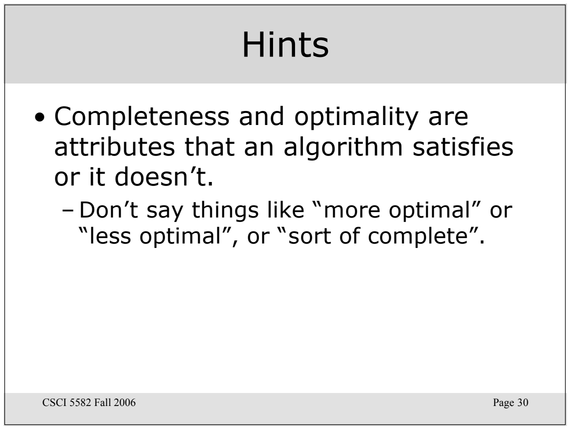
CSCI 5582 Fall 2006 Page 28



Search Criteria

- Space
  - Memory needed to find a solution in terms of some internal metric
    - Typically in terms of nodes stored
    - Typically what we care about is the maximum or peak memory use
- Optimality
  - When there is a cost function does the technique guarantee an optimal solution?

CSCI 5582 Fall 2006 Page 29



Hints

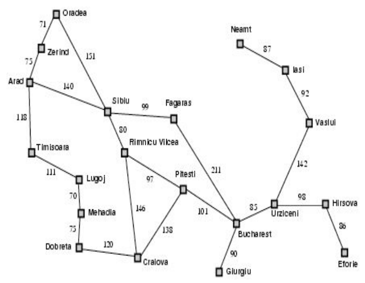
- Completeness and optimality are attributes that an algorithm satisfies or it doesn't.
  - Don't say things like "more optimal" or "less optimal", or "sort of complete".

CSCI 5582 Fall 2006 Page 30

Breadth First Search

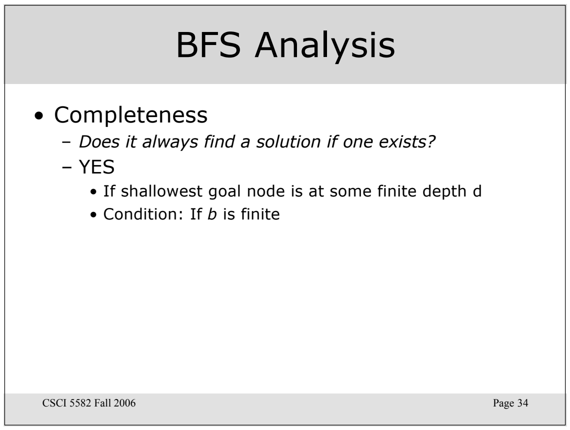
- Expand the shallowest unexpanded state
  - That means older states are expanded before younger states
  - I.e. A FIFO queue

BFS Bucharest



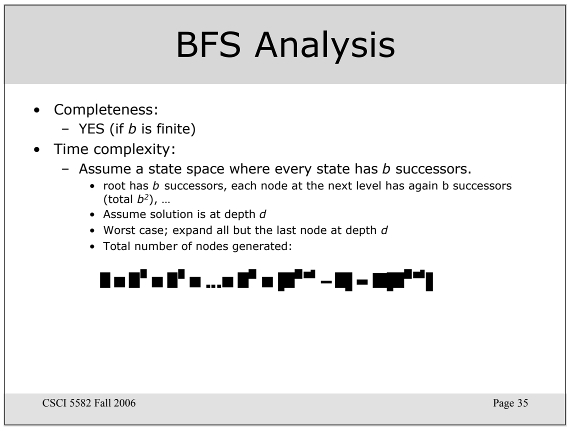
Terminology

- Branching factor (b)
  - Average number of options at any given point in time
- Depth (d)
  - (Partial) solution/path length



BFS Analysis

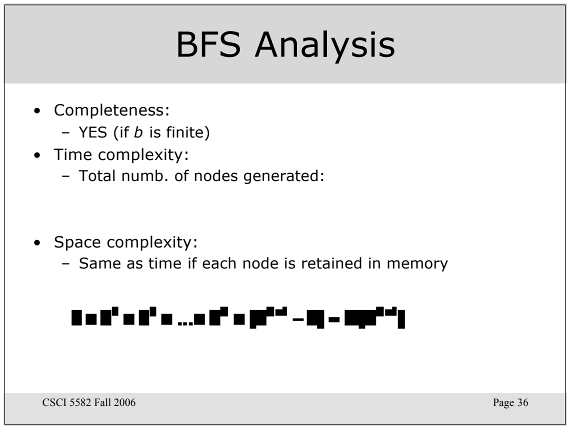
- Completeness
  - *Does it always find a solution if one exists?*
  - YES
    - If shallowest goal node is at some finite depth d
    - Condition: If *b* is finite



BFS Analysis

- Completeness:
  - YES (if *b* is finite)
- Time complexity:
  - Assume a state space where every state has *b* successors.
    - root has *b* successors, each node at the next level has again b successors (total b^2), …
    - Assume solution is at depth *d*
    - Worst case; expand all but the last node at depth *d*
    - Total number of nodes generated:

[illegible]



BFS Analysis

- Completeness:
  - YES (if *b* is finite)
- Time complexity:
  - Total numb. of nodes generated:
- Space complexity:
  - Same as time if each node is retained in memory

[illegible]

BFS Analysis

- Completeness
  - YES (if *b* is finite)
- Time complexity
  - Total numb. of nodes generated:
- Space complexity
  - Same if each node is retained in memory
- Optimality
  - *Does it always find the least-cost solution?*
    - Only if all actions have same cost

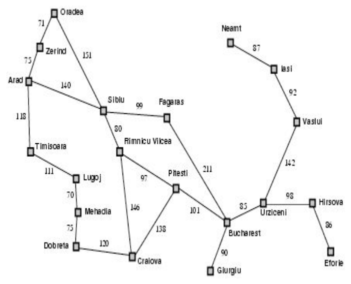
CSCI 5582 Fall 2006 Page 37

Uniform Cost Search

- How can we find the best path when we have actions with differing costs
  - Expand nodes based on minimum cost options
  - Maintain agenda as a priority queue based on cost

CSCI 5582 Fall 2006 Page 38

Uniform-Cost Bucharest

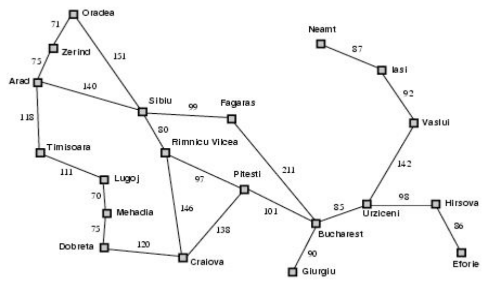


CSCI 5582 Fall 2006 Page 39

DFS

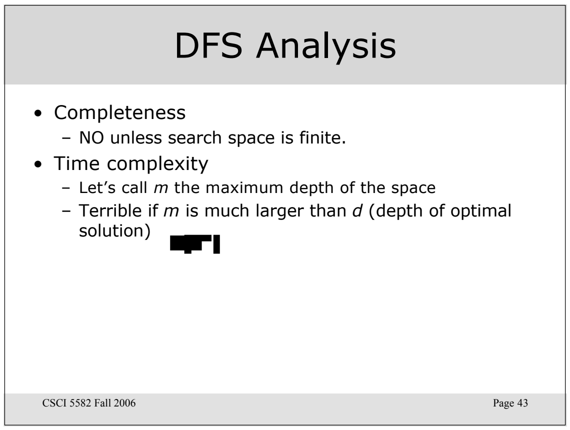
- Examine deeper nodes first
  - That means nodes that have been more recently generated
  - Manage queue with a LIFO strategy

DFS Bucharest



DFS Analysis

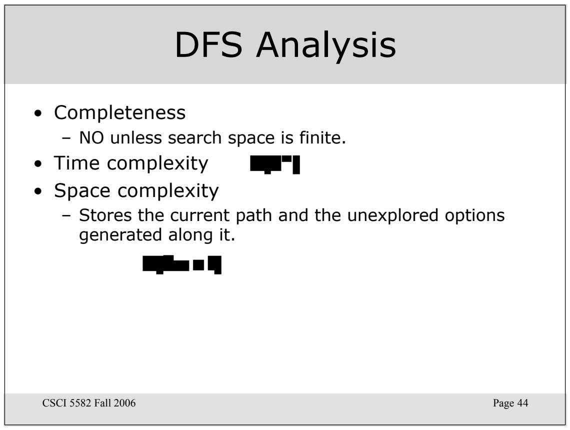
- Completeness;
  - *Does it always find a solution if one exists?*
  - NO
    - *unless* search space is finite and no loops are possible



DFS Analysis

- Completeness
  - NO unless search space is finite.
- Time complexity
  - Let’s call *m* the maximum depth of the space
  - Terrible if *m* is much larger than *d* (depth of optimal solution) [redacted]

CSCI 5582 Fall 2006 Page 43

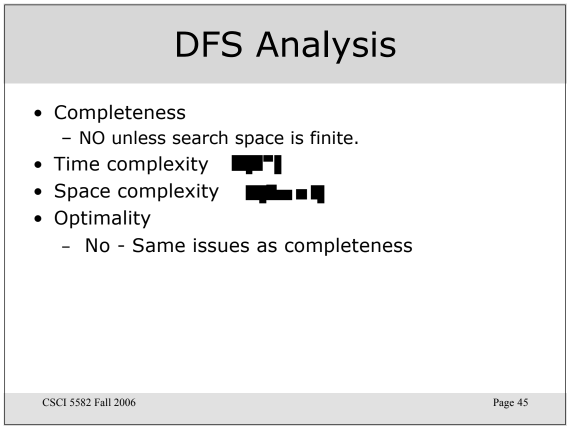


DFS Analysis

- Completeness
  - NO unless search space is finite.
- Time complexity [redacted]
- Space complexity
  - Stores the current path and the unexplored options generated along it.

    [redacted]

CSCI 5582 Fall 2006 Page 44



DFS Analysis

- Completeness
  - NO unless search space is finite.
- Time complexity [redacted]
- Space complexity [redacted]
- Optimality
  - No - Same issues as completeness

CSCI 5582 Fall 2006 Page 45

Depth Limiting Methods

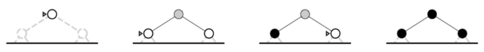
- Best of both DFS and BFS
- BFS is complete but has bad memory usage; DFS has nice memory behavior but doesn’t guarantee completeness. So…
  - Start with some depth limit (say 0)
  - Search for a solution using DFS
  - If none found increment depth limit
  - Search again…

ID-search, example

- Limit=0

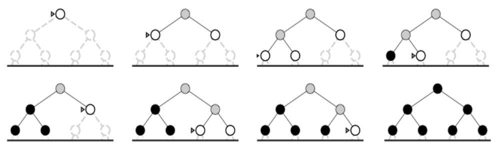
ID-search, example

- Limit=1



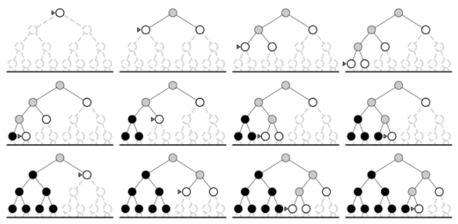
ID-search, example

- Limit=2



ID-search, example

- Limit=3



Iterative Deepening Analysis

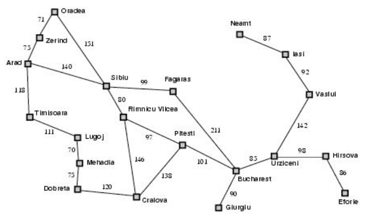
- Looks bad… Does lots of work at a given level and then throws it all away and starts over.
- Is it really a problem?
- The work done in then end (the iteration where a solution is found) is the SUM of the work done on all proceeding levels.
- But how does the work change from level to level?

Iterative Deepening

- If you
  - Don’t know the depth of likely solutions
  - And the search space is large
  - And you’re uninformed
- Then an iterative deepening method is the way to go

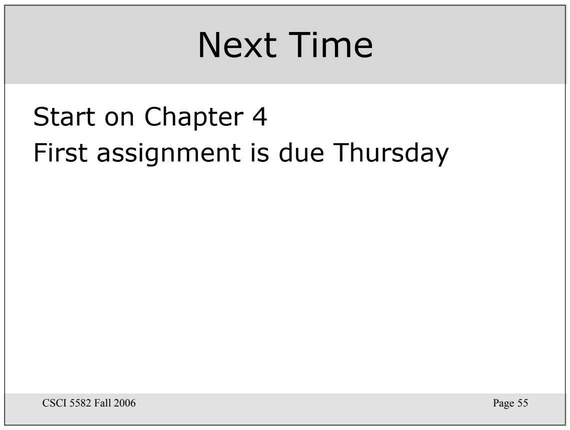
Uninformed?

- What is it that uninformed methods are uninformed about?



Review

- Attaining goals involves reasoning about how to get to hypothetical states
- This can be formalized as a search
- All searches can be viewed as variations on a theme
- In practical applications, memory becomes a problem long before time does



Next Time

Start on Chapter 4

First assignment is due Thursday

CSCI 5582 Fall 2006 Page 55