

Uniformed Search (cont.)

Computer Science cpssc322, Lecture 6

(Textbook finish 3.5)

January, 15, 2010



Lecture Overview

- **Recap DFS vs BFS**
- Uninformed Iterative Deepening (IDS)
- Search with Costs

Recap: Graph Search Algorithm

Input: a graph, a start node, Boolean procedure $goal(n)$ that tests if n is a goal node

$frontier := [\langle s \rangle : s \text{ is a start node}]$;

While $frontier$ is not empty:

select and remove path $\langle n_0, \dots, n_k \rangle$ from $frontier$;

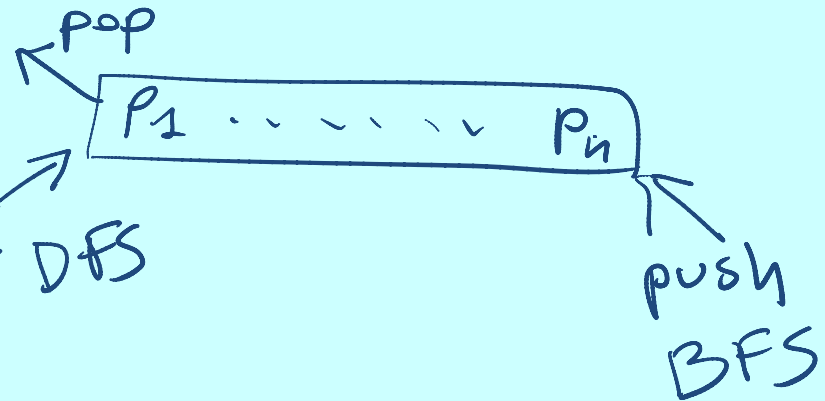
If $goal(n_k)$

return $\langle n_0, \dots, n_k \rangle$;

For every neighbor n of n_k

add $\langle n_0, \dots, n_k, n \rangle$ to $frontier$;

end



In what aspects DFS and BFS differ when we look at the generic graph search algorithm?

Recap: Comparison of DFS and BFS

	Complete	Optimal	Time	Space
DFS	- N (Y if no cycles)	Y N	+/- $O(b^m)$	+ $O(b^m)$
BFS	Y	Y	$O(b^m)$	$O(b^m)$

Lecture Overview

- Recap DFS vs BFS
- Uninformed Iterative Deepening (IDS)
- Search with Costs

Iterative Deepening (sec 3.6.3)



How can we achieve an acceptable (linear) space complexity maintaining completeness and optimality?

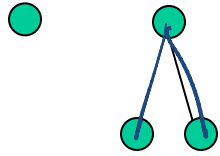
	Complete	Optimal	Time	Space
DFS	N	N	b^m	$m b$
BFS	Y	Y	b^m	b^m
\hat{I} IDS	Y	Y	b^m	$m b$

Key Idea: let's re-compute elements of the frontier rather than saving them.

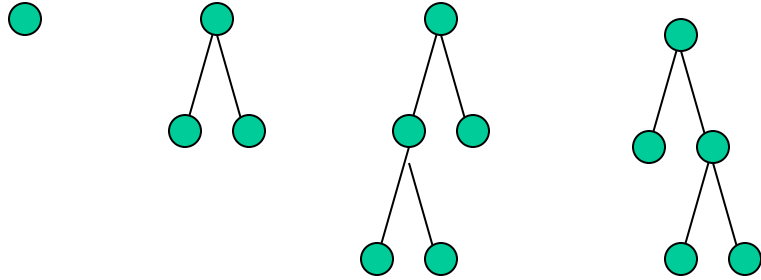
Iterative Deepening in Essence

- Look with **DFS** for solutions at depth 1, then 2, then 3, etc.
- If a solution cannot be found at depth D , look for a solution at depth $D + 1$.
- You need a **depth-bounded depth-first searcher**.
- Given a bound B you simply assume that paths of length B cannot be expanded....

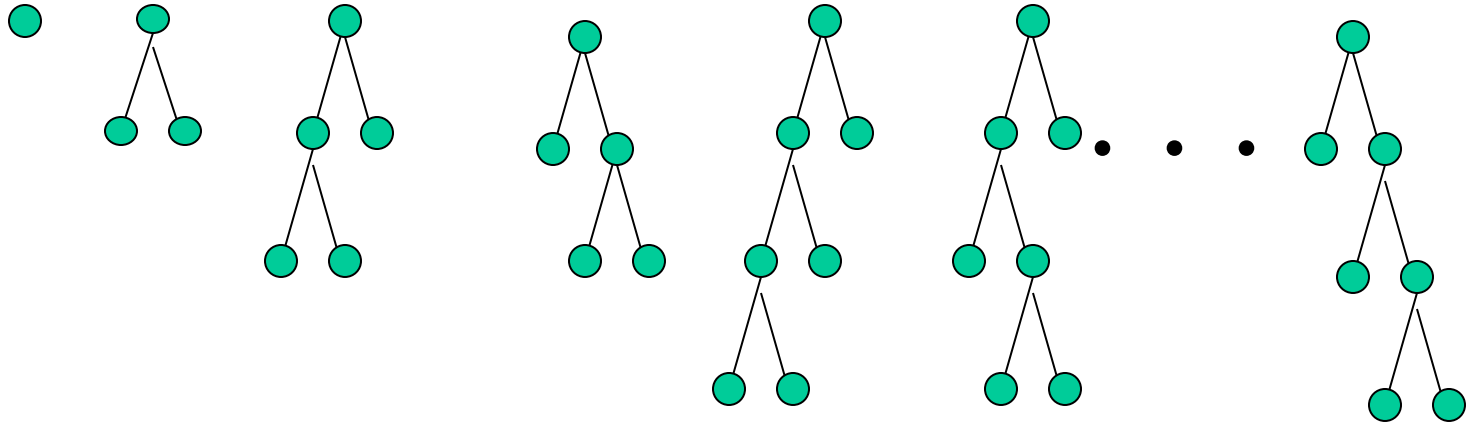
depth = 1



depth = 2



depth = 3



(Time) Complexity of Iterative Deepening

Complexity of solution at depth m with branching factor b

Total # of paths generated

$$b^m + 2b^{m-1} + 3b^{m-2} + \dots + mb = A$$

$A < B$

$$b^m (1 + 2b^{-1} + 3b^{-2} + \dots + mb^{1-m}) \leq$$

$$b^m \left(\sum_{i=0}^{\infty} \frac{b^i}{b^i} \right) = b^m \left(\frac{b}{b-1} \right) = O(b^m)$$

$$b = 2$$

$$\boxed{4}$$

$$b = 3$$

$$\frac{9}{4} = \boxed{2.25}$$

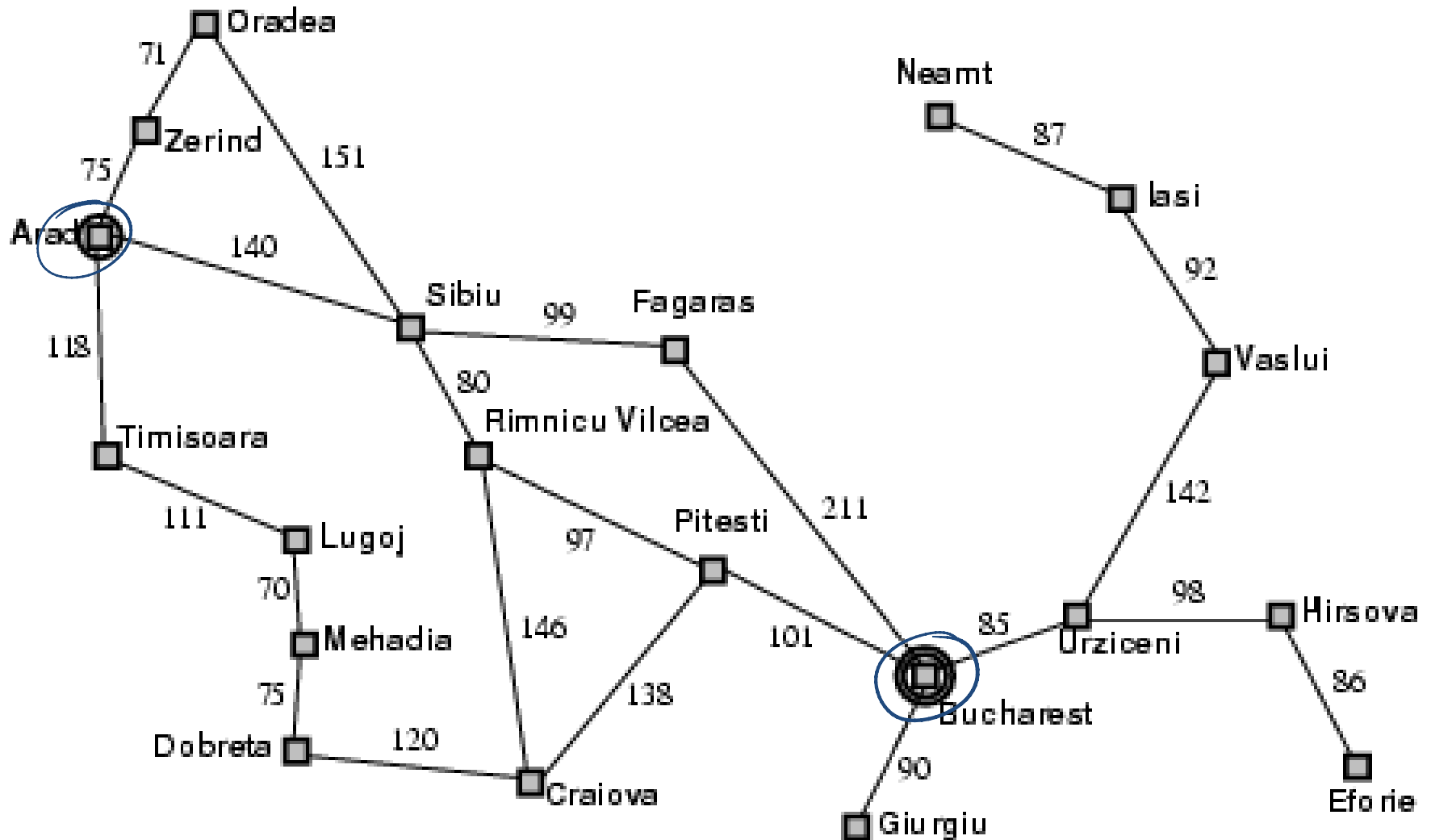
$$b = 4$$

$$\boxed{\frac{16}{9} < 2}$$

Lecture Overview

- Recap DFS vs BFS
- Uninformed Iterative Deepening (IDS)
- **Search with Costs**

Example: Romania



Search with Costs

Sometimes there are **costs** associated with arcs.

Definition (cost of a path)

The cost of a path is the sum of the costs of its arcs:

$$\text{cost}(\langle n_0, \dots, n_k \rangle) = \sum_{i=0}^{k-1} \text{st}(\langle n_i, n_{i+1} \rangle)$$

In this setting we often don't just want to find just any solution

- we usually want to find the solution that **minimizes cost**

Definition (optimal algorithm)

A search algorithm is **optimal** if it is complete, and only returns cost-minimizing solutions.

Lowest-Cost-First Search

- At each stage, lowest-cost-first search selects a path on the frontier with **lowest cost**.
 - The frontier is a priority queue ordered by path cost
 - We say "a path" because there may be ties
- When all arc costs are equal, LCFS is equivalent to ? BFS ?
- **Example of one step for LCFS:**
 - the frontier is $\langle p_2, 5 \rangle, \langle p_3, 7 \rangle, \langle p_1, 11 \rangle$
 - p_2 is the lowest-cost node in the frontier
 - "neighbors" of p_2 are $\langle p_9, 10 \rangle, \langle p_{10}, 15 \rangle$
- What happens?
 - p_2 is selected, and tested for being a goal.
 - Neighbors of p_2 are inserted into the frontier
 - Thus, the frontier is now $\langle p_3, 7 \rangle, \langle p_9, 10 \rangle, \langle p_1, 11 \rangle, \langle p_{10}, 15 \rangle$.
 - ? p_3 ? is selected next.
 - Etc. etc.

Analysis of Lowest-Cost Search (1)

- Is LCFS **complete**?
 - not in general: a cycle with zero or negative arc costs could be followed forever.
 - yes, as long as arc costs are strictly positive

AI space

cost ≥ 0



- Is LCFS **optimal**?
 - Not in general. Why not?
 - Arc costs could be negative: a path that initially looks high-cost could end up getting a "refund".
 - However, LCFS *is* optimal if arc costs are guaranteed to be non-negative.

cost ≥ 0

Analysis of Lowest-Cost Search

- What is the **time complexity**, if the maximum path length is m and the maximum branching factor is b ?
 - The time complexity is $O(b^m)$: must examine every node in the tree.
 - Knowing costs doesn't help here.
- What is the **space complexity**?
 - Space complexity is $O(b^m)$: we must store the whole frontier in memory.

Learning Goals for Search (up to today)

- Apply basic properties of search algorithms: completeness, optimality, time and space complexity of search algorithms.

	Complete	Optimal	Time	Space
DFS	N	N	b^m	$b \cdot m$
BFS	Y	Y	n	b^m
ID3	Y	Y	1	$b \cdot m$
LEFS	N Y if $c > 0$	N Y if $c > 0$	b^m b	b^m b

Learning Goals for Search (cont') (up to today)

- Select the most appropriate search algorithms for specific problems.

• BFS vs DFS vs IDS vs BidirS-

• LCFS vs. BFS -

• ~~A* vs. B&B vs IDA* vs MBA*~~

uninformed

informed

- Define/read/write/trace/debug different search algorithms
 - With / Without cost
 - ~~Informed~~ / Uninformed

Beyond uninformed search....

- So far the selection of the next path to examine (and possibly expand) is based on
....

Next Class

- **Start Heuristic Search**
(textbook.: start 3.6)