Uniformed Search (cont.)

Computer Science cpsc322, Lecture 6

(Textbook finish 3.4)

January, 16, 2009

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:= [$<s>$: *s* is a start node];

While *frontier* is not empty:

  select and **remove** path $<n_o, \ldots, n_k>$ from *frontier;*

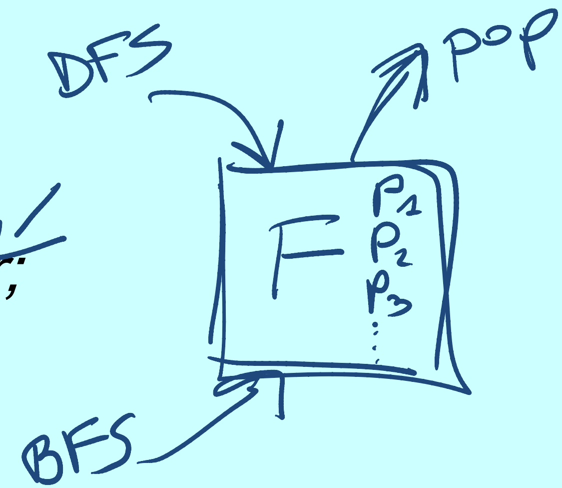
  If *goal*(n_k)

    return $<n_o, \ldots, n_k>$;

  For every neighbor *n* of n_k

    add $<n_o, \ldots, n_k, n>$ to *frontier;*

end



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

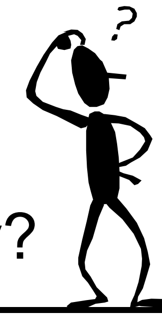
Recap: Comparison of DFS and BFS

	Complete	Optimal	Time	Space
DFS	N	N	$O(b^m)$	bm
BFS	Y	Y	$O(b^m)$	b^m

Lecture Overview

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

Iterative Deepening (sec 3.6.3)



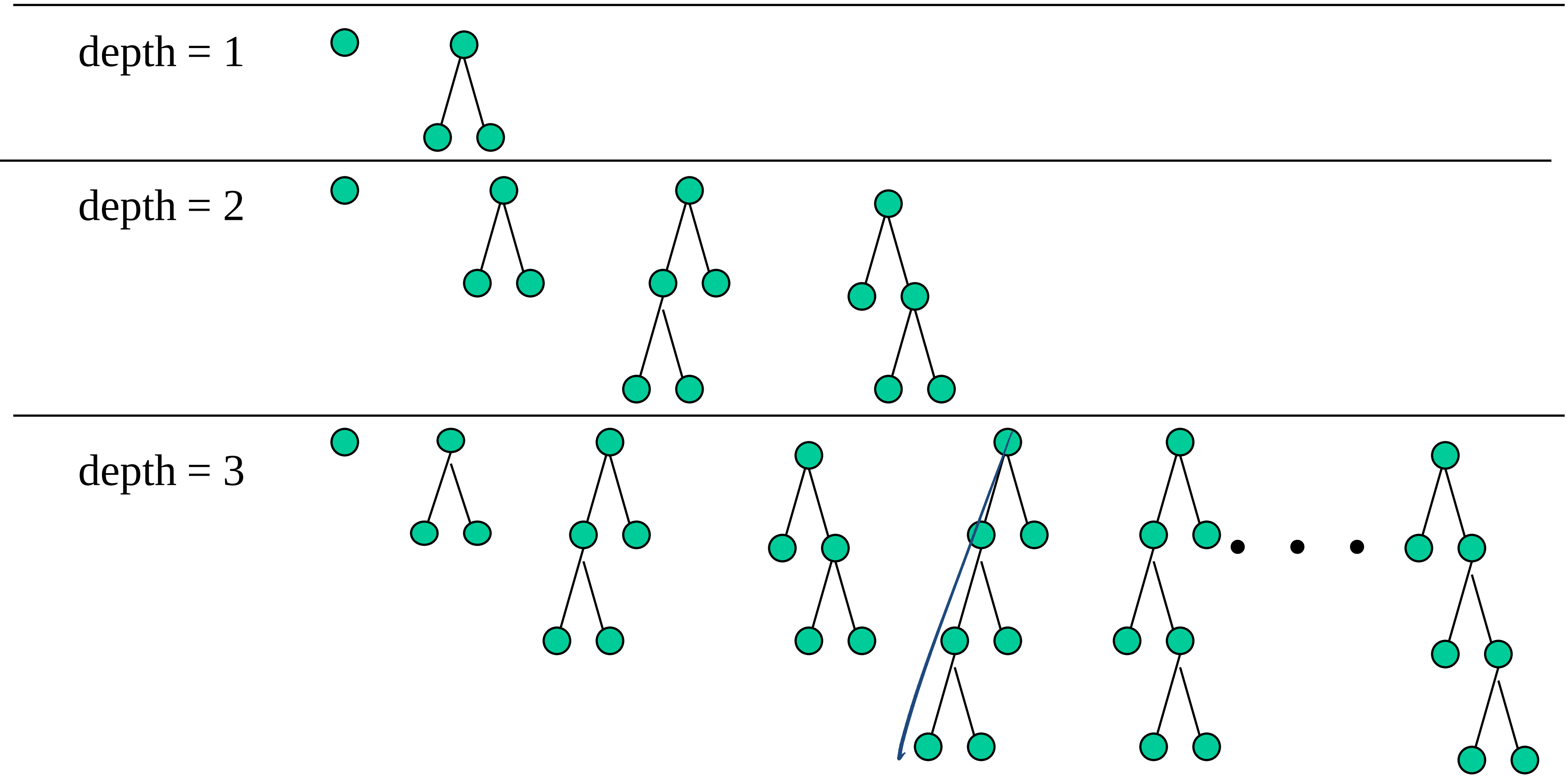
How can we achieve an acceptable (linear) space complexity maintaining completeness and optimality?

	Complete	Optimal	Time	Space
DFS	N	N	b^m	mb
BFS	Y	Y	b^m	b^m
	Y	Y	b^m	mb

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

Iterative Deepening in Essence

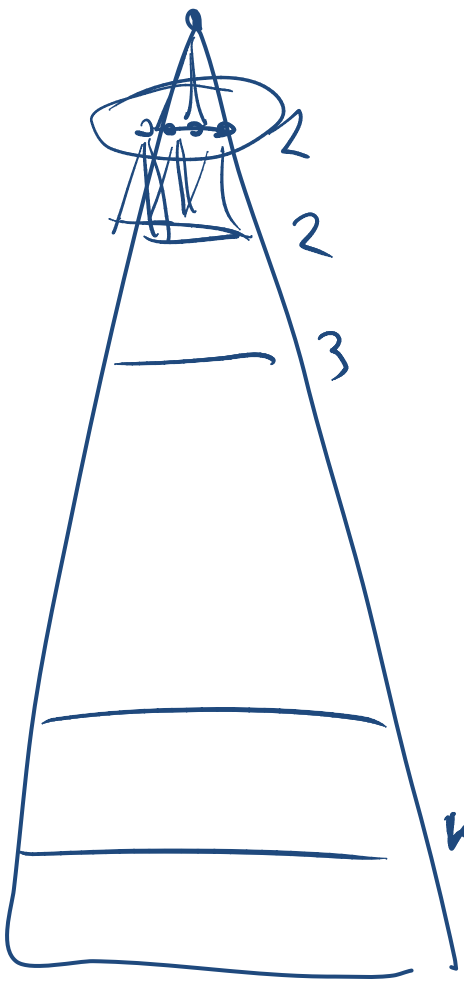
- Look with DFS for solutions at ~~depth~~ 0, then 1, then 2, then 3, etc.
- If a solution cannot be found at depth *D*, look for a solution at depth *D* + 1.
- You need a depth-bounded depth-first searcher.
- Given a bound B you simply assume that paths of length B cannot be expanded….


depth = 1
depth = 2
depth = 3

(Time) Complexity of Iterative Deepening

Complexity of solution at depth m with branching factor b

total # of paths created by IDS is sum of this column



	Total # of paths at that level	#times created by BFS (or DFS)	#times created by IDS	Σ
	b	1	m	mb
	b^2	1	$m-1$	$(m-1)b^2$
	b^3	1		
		1		
$m-1$	b^{m-1}	1	2	$2b^{m-1}$
m	b^m	1	1	b^m

(Time) Complexity of Iterative Deepening

Complexity of solution at depth m with branching factor *b*

Total # of paths generated

$$\boxed{b^m + 2\,b^{m-1} + 3\,b^{m-2} + .. + mb} =$$

$$b^m\,(1 + 2\,b^{-1} + 3\,b^{-2} + .. + m\,b^{1-m}) \leq$$

$$b^m\boxed{\left(\sum_{i=1}^{\infty} i b^{1-i}\right)} = b^m \boxed{\left(\frac{b}{b-1}\right)^2} = O(b^m)$$

$b=2$ 4

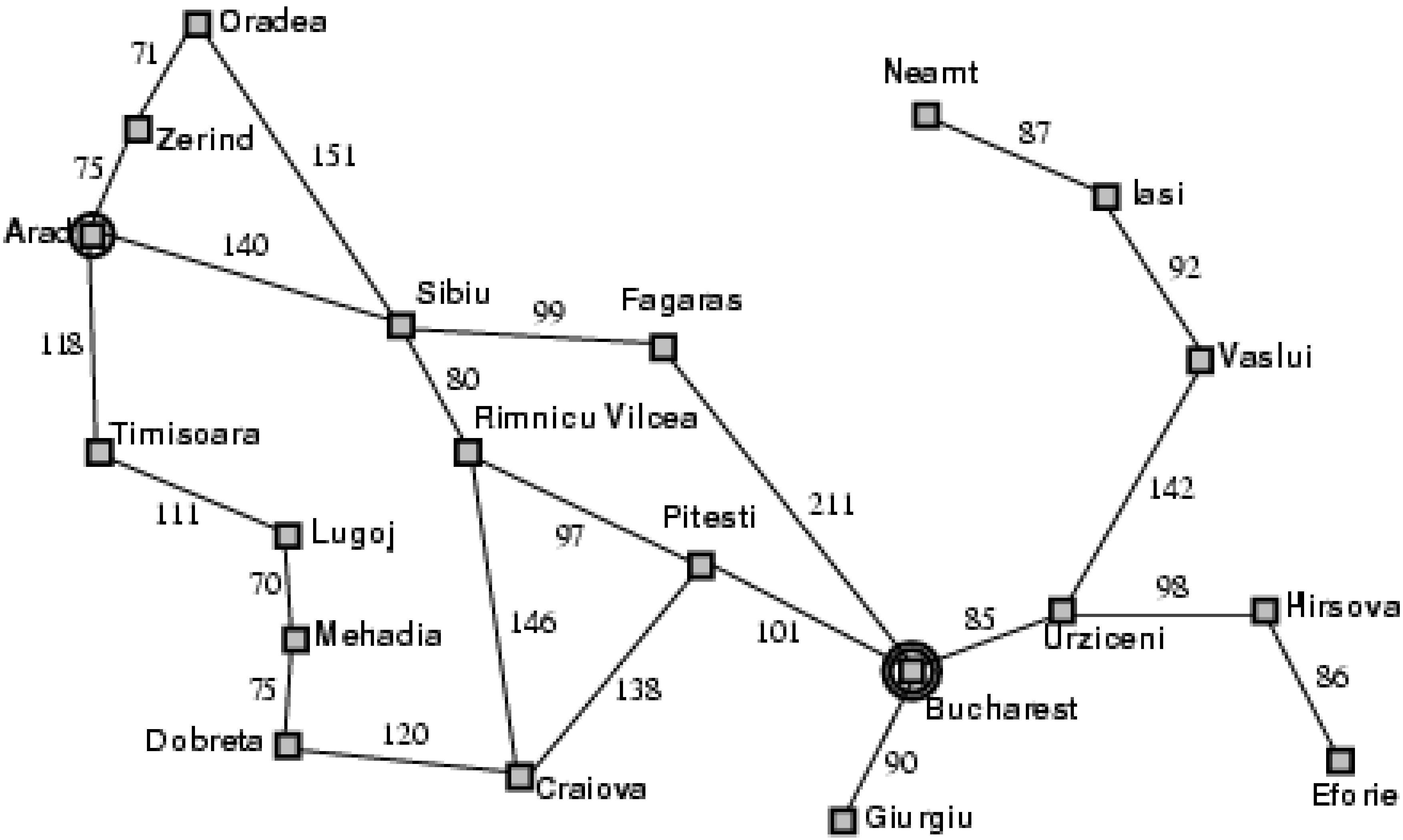
$b=3$ $\frac{9}{4} = 2.25$

$b=4$ $\frac{16}{9} < 2$

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:

$$\text{cost}\left(\langle n_0, \ldots, n_k \rangle\right) = \sum_{i=1}^{k} \text{cost}(\langle n_{i-1}, n_i \rangle)$$

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.

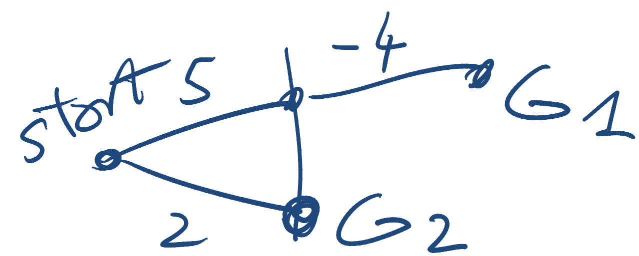
Lowest-Cost-First Search

- 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:
  - 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 $C > 0$

- 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. $C \geq 0$

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	bm
BFS	Y	Y	"	b^m
IDS	Y	Y	"	bm
LCFS	N / Y $c>0$	N / Y $c \geq 0$	"	b^m

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

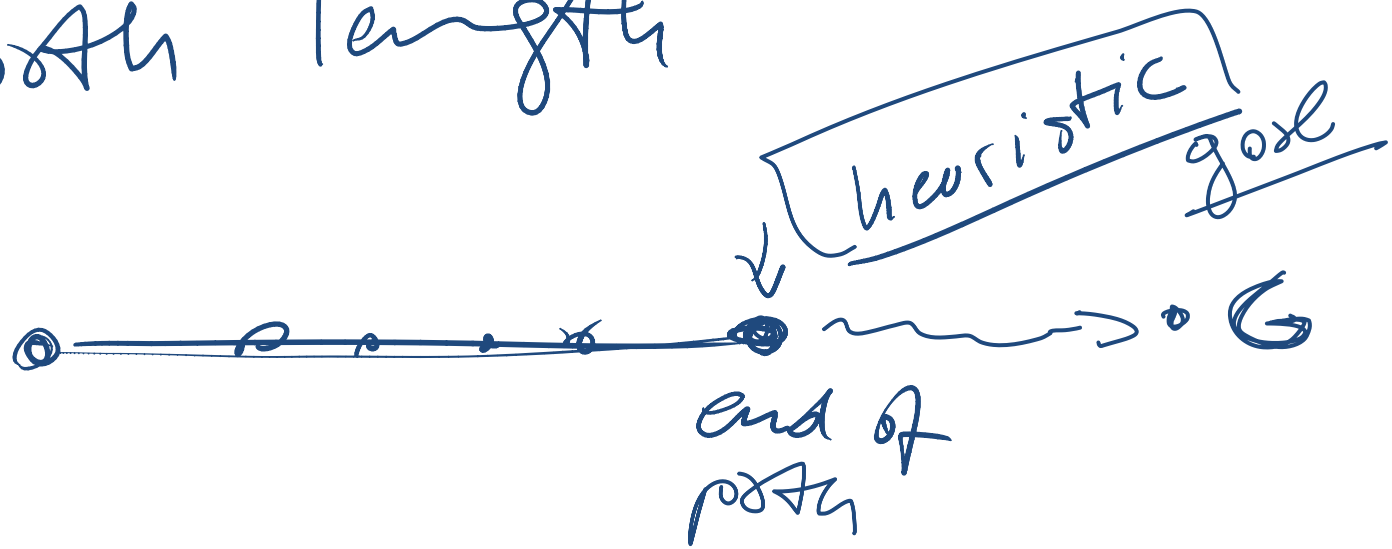
- Select the most appropriate search algorithms for specific problems.
  - BFS vs DFS vs IDS vs ~~BidirS-~~
  - LCFS vs. ~~BFS –~~
  - ~~A* vs. B&B vs IDA* vs MBA*~~

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

Why uninformed search?

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

path length



Next Class

- Start **Heuristic Search**

(textbook.: start 3.5)